PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA

CONTROLADOR NEURAL PREDITIVO MULTIVARIÁVEL APLICADO A UMA PLANTA PILOTO DE NEUTRALIZAÇÃO DE pH

Silas de Miranda Prottes

Dissertação submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia do Centro Universitário do Leste de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Mestre em Engenharia Industrial.
Área de Concentração: Processos Industriais

Orientador: Prof. Dr. Roselito de Albuquerque Teixeira

Coronel Fabriciano, Julho de 2009

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA

CONTROLADOR NEURAL PREDITIVO MULTIVARIÁVEL APLICADO A UMA PLANTA PILOTO DE NEUTRALIZAÇÃO DE pH

Silas de Miranda Prottes

Banca:

Prof. Dr. Roselito de Albuquerque Teixeira - PPGE/Unileste-MG - Orientador.
Prof. Dr. Marcelo Vieira Corrêa - PPGE / UnilesteMG.
Prof. Dr. Dair José de Oliveira - PPGE / UnilesteMG.
Prof. Dr. Marcelo Azevedo Costa - UFMG.

Aos meus pais.

Agradecimentos

Agradeço a Deus por fazer o ser humano dotado de capacidade lógica/intelectual e pensamento crítico.

Agradeço à minha mãe por ser um exemplo de cuidado e ao meu pai pelas incontáveis horas de trabalho em prol do sustento de nossa família.

Agradeço ao orientador Prof. Dr. Roselito de Albuquerque Teixeira pelo grande incentivo, orientando com sugestões, e às muitas importantes perguntas e pontos de vista que me fizeram pensar de forma mais crítica e analista.

Agradeço aos demais professores do Unileste-MG por passarem o conhecimento de forma objetiva e necessária no decorrer do Mestrado.

Agradeço ao Unileste-MG e à FAPEMIG pelos recursos materiais e financeiros, de imensa ajuda durante o desenvolvimento deste trabalho.

Agradeço à Sandra e demais funcionários do Laboratório de Pesquisas Ambientais do Unileste-MG pela preparação das inúmeras soluções químicas para a realização dos testes na planta piloto de neutralização de *pH*.

Agradeço aos colegas de Mestrado pelas discussões pertinentes e ao colega Silvano pela disponibilização das rotinas de comunicação utilizadas no controle da planta piloto de neutralização de *pH*.

Agradeço também aos Doutores Rodrigo Iván Goytia Mejía - UFSC e Leizer Schnitman - UFBA, pela disponibilização de materiais que serviram de ajuda para o desenvolvimento de algumas partes desta pesquisa.

Espero não ter esquecido o nome de ninguém, se assim ocorrido peço sinceras desculpas e agradeço pelo apoio.

"O maior guerreiro de todos é aquele
que vence a si mesmo."

Bruce Lee

Resumo

Este trabalho apresenta a implementação de um controlador neural preditivo para controle de um sistema de neutralização de *pH* multivariável. Atualmente, com o aumento da exigência do mercado quanto à qualidade dos produtos, a conservação ambiental, a otimização na produção e outros fatores, levam a um aumento no número de informações utilizadas no controle de sistemas. Este crescente número de informações exige que os sistemas de controle estejam preparados cada vez mais para lidarem com processos reais. Sistemas de controle baseados em linearização podem não corresponder a um padrão de qualidade elevado, pois as não-linearidades trazem informações cruciais, que normalmente estão diretamente relacionadas com a elevação da qualidade do produto final. Estratégias de controle baseadas em modelos não-lineares são, portanto, uma saída adequada. Dentro da classe de controladores baseados em modelo, a estratégia de controle preditivo generalizado utilizando um modelo neural não-linear explícito do processo permitiu o controle do processo de forma satisfatória. A utilização das Redes Neurais Artificiais garante uma incorporação satisfatória de dinâmicas do processo a ser controlado. Com a utilização da estratégia de controle preditivo neural generalizado aliada ao método de otimização de Newton-Raphson, obtém-se resultados em rastreabilidade e outros testes, tanto para um sistema simulado de uma planta de neutralização de *pH* como para os resultados obtidos na planta piloto de neutralização de *pH*. Resultados mostram que a mesclagem das características da classe de controladores baseados em modelo, redes neurais artificiais e Newton-Raphson possibilitam o alcance de resultados satisfatórios.

Palavras-chave: Redes Neurais. Newton-Raphson. Controle preditivo generalizado.

Abstract

This paper presents the implementation of a neural predictive controller for control of a multivariable pH neutralization system. Currently, with the increasing market demand for product quality, environmental conservation, optimization of production and other factors lead to an increase in information to be used in control systems. This growing number of information requires that control systems be increasingly prepared to deal with real processes. Control systems based on linearization may not be a high quality standard, since the nonlinearities bring crucial information, which are usually directly related to raising the quality of the final product. Control strategies based on nonlinear models are therefore appropriate solutions. Within the class of model-based controllers, the strategy of generalized predictive control using an explicit nonlinear neural model of the process assures the control satisfactorily. The use of artificial neural networks provides a satisfactory incorporation of process dynamics to be controlled. By using this strategy of neural generalized predictive control combined with the Newton-Raphson optimization method, results on traceability and other tests are obtained, both for a simulated system of a pH neutralization plant and the results obtained in pilot plant for pH neutralization. Results show that the merging of the characteristics of the class of model-based controllers, artificial neural networks and Newton-Raphson enable the achievement of satisfactory results.

Key Words: Neural Networks. Newton-Raphson. Generalized predictive control.

Sumário

Lista de Figuras

Lista de Símbolos

k	Iteração, instante de tempo.
$u(k)$	Ação de controle no instante k
N	Identificação para neurônio
$y(k)$	Saída do processo no instante k
$y_e(k+1)$	Saída estimada do processo para o instante k + 1
$r(k+1)$	Sinal de referencia para o instante k + 1
$e(k)$	Erro no instante de tempo k
$e(k+1)$	Erro do controlador estimado para o instante de tempo k + 1
$\mathbf{W_1}$	Matriz de pesos dos neurônios da camada oculta
$\mathbf{w_2}$	Vetor de pesos dos neurônios da camada de saída
$\mathbf{b}$	Vetor dos termos de polarização dos neurônios da camada escondida
b_s	Termo de polarização do neurônio da camada de saída
$d(k)$	Distúrbio no instante k
Q	Vazão
λ	Variável de ponderação ao esforço de controle
m	Memória de atraso referente a variável y
n	Memória de atraso referente a variável u_1
o	Memória de atraso referente a variável u_2
z^{-1}	Operador de atraso
ϑ	Derivada parcial
x	Sinal de entrada do neurônio
p	Entrada do neurônio
w_{Np}	Peso sináptico da entrada p do N-ésimo neurônio
$\sum$	Somatório
$\mathcal{F}$	Função não-linear
$\varphi(.)$	Função de ativação do neurônio
$\varphi'(.)$	Derivada da função de ativação do neurônio

y_n	Saída do N-ésimo neurônio
υ_N	Nível de ativação interna ou campo local induzido do N-ésimo neurônio
υ	Nível de ativação interna ou campo local induzido
η	Taxa de aprendizagem
$\Delta w_{Np}\left(k\right)$	Correção do peso do N-ésimo neurônio na iteração k
$\delta_N\left(k\right)$	Gradiente local do N-ésimo neurônio na iteração k

Siglas e Abreviações

NARX	Modelo não-linear auto-regressivo com entradas exógenas (*Nonlinear Auto Regressive model with eXogenous inputs*)
RNA	Rede Neural Artificial
IC	Inteligência Computacional
IA	Inteligência Artificial
MLP	Redes neurais multi-camadas (*Multi Layer Perceptron*)
MSE	Média do somatório dos erros quadráticos (*Mean Squared Errors*)
RBF	Redes de funções de base radiais (*Radial Basis Function*)
PID	Proporcional Integral Derivativo
MIMO	Múltiplas Entradas e Múltiplas Saídas (*Multi-Input Multi-Output*)
SISO	Uma Entrada e Uma Saída (*Single-Input Single-Output*)
USB	Barramento Serial Universal (*Universal Serial Bus*)
GPC	Controle preditivo generalizado (*Generalized Predictive Control*)
NGPC	Controle Preditivo Neural Generalizado (*Neural Generalized Predictive Control*)
MRC	Controle por modelo de referência (*Model Reference Control*)
IMC	Controle por modelo interno (*Internal Model Control*)
MPC	Classe de controladores baseados em modelo (*Model Predictive Control*)

Capítulo 1

Introdução

Com as exigências de qualidade fabril e a otimização da produção, tende-se a fazer menor uso de sistemas de controle baseados no uso da linearização, devido ao fato de que o uso destes sistemas de controle baseados em linearização não podem dar garantia de resultados satisfatórios (te Braake et al., 1998).

Quando há evidência de existência de relações não-lineares fortes no controle de sistemas, o uso de controladores convencionais (e.g., Controlador Proporcional Integral Derivativo - PID) pode também não ser adequado. Nestas situações o uso de estratégias de controle avançado oferecem melhor desempenho (Seborg, 1999). Isto se torna um incentivo para a utilização de métodos de controle avançado, reservando o seu uso para problemas de difícil controle onde podem ser obtidos desempenhos significantemente superiores sobre os controladores convencionais.

O processo de neutralização de *pH* é um processo que apresenta alta não-linearidade e grande variação do comportamento no tempo e não pode ser devidamente controlado por controladores convencionais (Henson e Seborg, 1994; Böling et al., 2007). Uma tentativa de minimizar este problema, seria o uso de estratégias de controle adaptativo referidas como *switching control* ou *multi-model adaptative control* em conjunto com PID (Böling et al., 2007). Uma outra abordagem seria a utilização dos Algoritmos Genéticos - AG juntamente com *Strength Pareto Evolutionary Algorithm* - SPEA2 (Popov et al., 2005; Zitzler et al., 2001), reconhecida como uma das técnicas multi-objetivo mais poderosas (Zitzler et al., 2000).

O controle convencional PID tem sido usado em processos de *pH* por muitos anos. Há uma exigência no ajuste adequado dos parâmetros do controlador para satisfazer o ponto de operação do processo. O ajuste destes parâmetros são tradicionalmente ajustados empiricamente ou

através dos métodos *Ziegler-Nichols* e *Cohen Coon* (Aström e Hägglund, 2001; Schei, 1994). Estes ajustes são realizados para um ponto de operação onde o modelo é considerado linear e não para uma ampla faixa.

O uso de estratégias de controle baseadas em modelo interno levam também a resultados superiores aos obtidos com o uso de controladores convencionais. A superioridade sobre controladores convencionais aplicados em um processo de neutralização de *pH* já foi também comprovada (Nahas et al., 1992).

Com a necessidade de evolução industrial, a partir da última década percebe-se um crescimento no número de aplicações industriais envolvendo estratégias de controle avançado. A classe de controladores MPC (*Model Predictive Control*) passa a assumir papel importante dentro do controle avançado de processos (Richalet, 1993). Isso pode ser devido às suas vantagens sobre controladores convencionais (Garcia et al., 1989). Assim, no final da década de 1980, MPC se estabeleceu como o dispositivo de controle padrão especialmente em indústrias petroquímicas e de refinaria (Qin e Badgwell, 1997).

Como a utilização de modelos lineares é mais antiga e os processos de hoje precisam operar dentro de especificações mais altas, o uso de MPC baseado em modelos não-lineares se torna necessário. Controle preditivo baseado em modelos não-lineares (*Nonlinear Model Predictive Control* - NMPC) é uma forma de satisfazer estas demandas com a utilização de modelos não-lineares do processo e a utilização de restrições no controlador. Segundo Findeisen e Allgower (2002), a abordagem NMPC parece ser uma abordagem satisfatória para se tratar as demandas do mundo moderno.

Mesmo já assumindo papel importante, segundo (Nikolaou, 2001) um número crescente de pesquisas podem tornar a estratégia de controle MPC uma ferramenta mais poderosa. Mesmo ainda em constante desenvolvimento, MPC está sendo amplamente utilizado na indústria como uma maneira efetiva de lidar com problemas de controle multivariável restrito (Zhang et al., 2008).

Na implementação do MPC, Richalet (1993) afirma que as dificuldades práticas encontradas aparecem mais na seleção do modelo e formulação de especificações do que estritamente o projeto do controlador. Uma característica bastante útil, e que facilita a implementação do MPC, é a possibilidade das restrições serem incorporadas diretamente na função de custo, o que faz desta estratégia de controle uma boa opção para processos em tempo real (Camacho e Bordons, 1997).

É notado portanto que a aplicação de métodos de controle MPC tem crescido e várias formas de abordagem e formas de implementação tem sido estudadas. van Overloop et al.

(2008) usa uma abordagem probabilística numa função de custo a ser minimizada no controle de nível de reservatórios de captação hídrica. O processo em funcionamento, realiza cálculos computacionais onde os dados são enviados para um operador com experiência a fim de serem aplicados aos equipamentos finais. O uso da probabilidade e estatística é também válida e é demonstrada pelo autor.

Zhang et al. (2008) utilizam MPC com o objetivo de aumentar a vida útil de membranas de troca de *proton* em células de combustível (*Proton Exchange Membrane Fuel Cells* - PEMFC). PEMFC são células que têm o objetivo de transformar energia química liberada durante uma reação eletroquímica de hidrogênio e oxigênio em energia elétrica. O controle da quantidade de água na membrana não somente melhora o desempenho e eficiência da célula mas também previne sua degradação irreversível. O autor utiliza a estratégia de controle MPC baseada em redes *feedback* ou recorrentes.

Dougherty e Cooper (2003) faz uso do MPC com uma estratégia de controle adaptativa baseada em múltiplos modelos. A idéia é manter o desempenho do controlador linear sobre uma ampla faixa de níveis de operação. Uma estratégia de controle baseada em múltiplos modelos para o uso em controle por matriz dinâmica (*Dynamic Matrix Control* - DMC) é utilizada. Vários controladores lineares DMC são utilizados. O autor salienta que o desenvolvimento desta estratégia para sistemas MIMO é de certa forma crítica, devido à interdependência das variáveis do processo.

Um dos algoritmos mais utilizados da classe de controladores MPC é o *Generalized Predictive Control* - GPC. GPC tem sido implementado em aplicações industriais mostrando seu bom desempenho (Clarke, 1988; Pam et al., 1999). A idéia básica do GPC é calcular uma seqüencia de ações de controle futuras a fim de minimizar uma função definida sob um horizonte de custo. O índice a ser otimizado é a esperança de uma função quadrática que avalia a distancia entre a saída predita e uma sequência de referência, ainda levando em conta o esforço de controle (Clarke et al., 1987).

De acordo com Huang e Lewis (2003), a habilidade do algoritmo GPC em fazer predições pode ser melhorada com a utilização de RNAs para aprenderem as dinâmicas da planta, ao invés de utilizar técnicas de modelagem não-linear padrão. Assim o algoritmo GPC passa a ser denominado controle preditivo neural generalizado (*Neural Generalized Predictive Control* - NGPC), devido ao uso de RNAs.

Quando se fala de controle de sistemas, a utilização de Redes Neurais Artificiais - RNAs apresenta vantagens confirmadas quando utilizadas na identificação e controle de sistemas dinâmicos não-lineares (Narendra e Parthasarathy, 1990). Segundo Narendra e Parthasarathy (1990),

para sistemas dinâmicos não-lineares, onde modelos matemáticos não são conhecidos a princípio, o uso das RNAs na identificação e controle pode ser realizado com resultados satisfatórios.

As RNA têm assumido então um papel importante, já que nos processos industriais a quantidade de dados é quase sempre grande favorecendo o uso destas. Modelos neurais não são tão complicados de serem obtidos, principalmente quando a quantidade de dados favorecem isso. Assim, a utilização de RNAs têm crescido a cada dia. A sua aplicação tem se difundindo em várias áreas como no campo de reconhecimento de caracteres, detecção de defeitos em partes metálicas (Oukhellou e Aknin, 1997), detecção de corrosão em oleodutos e gasodutos (Ma e Liu, 2007), aplicações em mineração de dados (Yang e Hamer, 2007; Misra et al., 2007), aplicações em robótica (Itoa et al., 2006; Barakat, 2006) e etc.

As RNAs possuem também suas aplicações no campo de controle. Ou e Rhinehart (2003) demonstram experimentalmente o uso de RNA como sub modelos onde cada modelo é responsável por uma função no controle. Cada sub modelo prediz uma saída do processo, sendo independentes e funcionando em paralelo. O controle é implementado num processo de destilação. Já Pam e Don (1997) apresentam uma implementação demonstrando o controle preditivo de um modelo não-linear. Para isso utilizam de derivações internas das próprias RNA, a fim de conduzirem a parcela de modificação na ação de controle até à saída do modelo neural e usar sua capacidade de predição para prever comportamentos futuros do processo. Utilizam o método de Newton-Raphson como minimizador da função de custo.

Schnitman e Fontes (1999) apresentam uma técnica de controle preditivo usando RNA. São desenvolvidas as equações de primeira ordem da RNA, que são usadas na regra de atualização das ações de controle. O algoritmo de controle preditivo baseado em redes neurais é então desenvolvido e usado para controlar uma planta específica. O autor propõe um controle adaptativo baseado no pólo dominante da equação do modelo do sistema controlado.

Chen e Narendra (2001) demonstram dentro do controle adaptativo, como uma lei de seleção de modelos pode melhorar o desempenho geral do controlador. Fazem uso de um controlador adaptativo linear robusto e um controlador adaptativo não-linear baseado em RNA. Ambos, se usados sozinhos, produzem desempenho não satisfatório, mas quando usados juntos e através de uma lei de seleção de modelos, produzem melhor desempenho e estabilidade.

Quando se fala da arquitetura das RNAs, uma rede MLP composta de uma camada oculta tem a capacidade de permitir a aproximação de qualquer função contínua e com o uso de duas camadas permite a aproximação de qualquer função (Cybenco, 1989).

Dentro das estratégias de controle envolvendo RNAs (controladores que fazem uso das RNAs), a RNA quanto à sua disposição dentro da arquitetura de controle pode ser classificada

em como sendo o próprio controlador ou como sendo um modelo do processo Agarwal (1997).

Em processos de *pH*, Bharathi et al. (2007) abordam o uso de um neuro-controlador e um controlador *fuzzy* em um processo de neutralização de *pH*. O autor faz testes com um controlador PI, mas este não demonstra desempenhos satisfatórios. Testes são refeitos utilizando um modelo inverso do processo baseado em RNA e um controlador *fuzzy*. O controlador *fuzzy* utiliza três entradas para gerar uma saída de controle, sendo uma delas o *setpoint* que dá informação sobre as diferentes regiões não-lineares do processo. Mesmo assim, o desempenho do neuro-controlador é melhor.

De acordo com Camacho e Bordons (2004a), o controle preditivo baseado em modelo se apresenta como a forma mais geral de formular um problema de controle no domínio do tempo e tem a capacidade de integrar controle ótimo, controle de processos com tempo morto, processos multivariáveis e a utilização de referências futuras quando disponíveis. Permitindo quando da utilização de um horizonte de controle, a consideração de restrições e processos não-lineares. Segundo Camacho e Bordons (2004a), a utilização de redes neurais formando os neuro-controladores é uma forma prática de implementar os controladores preditivos.

De acordo com Gustafsson et al. (1995), o que torna difícil o controle de *pH* é o fato de que a resposta obtida com uma mesma adição de ácido ou base difere muito de sistema pra sistema e de *pH* para *pH*: uma pequena quantidade adicionada pode mudar o *pH* muito ou muito pouco.

Devido ao fato de apresentar bons desafios para a avaliação de controladores, o processo de neutralização de *pH* continua sendo alvo de pesquisas. A planta piloto de neutralização de *pH*, portanto, tem tomado destaque nos últimos anos e se tornado um dos sistemas mais populares para avaliação de técnicas de controle. É um sistema consideravelmente simples, na forma de ser obtido, possui ampla utilização na indústria e apresenta desafio em ser controlado devido sua alta não-linearidade (Gustafsson et al., 1995). Peymani et al. (2008) apresentam uma comparação entre um controlador adaptativo convencional e um controlador adaptativo com múltiplos modelos. A comparação é realizada em termos do ponto de vista de rejeição de distúrbios. Os testes são realizados em uma planta piloto de neutralização de *pH*. A abordagem utilizando múltiplos modelos levou vantagem nos testes.

Yu e Yu (2007) apresentam uma abordagem a múltiplos modelos utilizando três modelos MISO (*Multi-Input Single-Output*) baseados em redes neurais. Cada modelo com taxa de amostragem diferente, sendo um modelo para cada saída do reator. O autor compara os resultados obtidos com três controladores PID. Testes são realizados em um processo de neutralização de *pH*. A técnica utilizando múltiplos modelos em redes neurais é a que apresenta menor erro.

Yu et al. (2002) fazem uso do controle preditivo baseado em RNA com característica adap-

tativa, em um reator multivariável utilizado para a neutralização de *pH*. Demonstram que em malha fechada a adaptabilidade inserida garante melhor desempenho no controle. Os autores fazem comparação de desempenho utilizando redes denominadas *pseudo-linear radial basis function* (PLRBF) com adaptação *on-line* com redes *radial basis function* (RBF) treinadas *off-line*. O resultado obtido é que a PLRBF faz uso de menor número de nós na camada oculta e ainda prediz com melhor precisão.

Outra forma de abordagem é a utilização de múltiplos modelos lineares (Galán et al., 2004). Já Böling et al. (2007) aborda dentro dos controladores PID, uma comparação através de algumas simulações em um processo de neutralização de *pH* de controladores PID multimodelo com controladores PID convencionais. Na abordagem a modelos múltiplos, um conjunto de modelos candidatos e/ou controladores são especificados à priori. O controlador supervisor seleciona o modelo mais apropriado e/ou o controlador para a condição atual. A abordagem em múltiplos modelos foi mais efetiva durante mudanças no processo, mas respondeu mais lentamente para distúrbios repentinos.

Noor et al. (2004) implementam um sistema de controle *fuzzy* baseado no modelo do sistema. O foco é a neutralização do *pH* em uma unidade de tratamento de água. O autor salienta a facilidade da lógica *fuzzy* em ser utilizada no controle de sistemas devido sua flexibilidade. Resultados apontam que não há diferenças entre o sistema *Sugeno* e o sistema *Mamdani*, se preferindo o sistema *Mamdani* devido à facilidade de computação.

Valarmathi et al. (2009) utiliza algoritmos genéticos (AG) e redes neurais artificiais em um processo de *pH*. O autor faz uso dos AGs para a atualização dos parâmetros do controlador PID nas mudanças das características dinâmicas do processo. O controlador convencional por si só apresentou muitas oscilações em rastreabilidade enquanto que a resposta utilizando AG obteve melhoras significativas.

Neste trabalho a estratégia de controle NGPC é utilizada, fazendo uso do algoritmo GPC com um modelo neural do processo. O controle preditivo multivariável é realizado com a utilização do método de otimização de Newton-Raphson. De acordo com Pam e Don (1997) o custo principal do algoritmo Newton-Raphson é a computação da Hessiana, mas ainda assim o baixo número de iterações realizadas, faz deste método um algoritmo bastante adequado para controle em tempo real. A velocidade de convergência aliada ao desafio que gira em torno de uma possível não-convergência faz do método de otimização de Newton-Raphson um desafio a ser aplicado (Edgar e Himmelblau, 2001; Luenberger, 2008).

1.1 Objetivos

O objetivo geral é pesquisar sobre controle neural preditivo e sua utilização dentro de estratégias de controle, com foco no desenvolvimento de um controlador e a realização de testes no controle de uma planta piloto de neutralização de *pH*. Pode-se definir como objetivos específicos:

- Comprovar a eficácia da rede neural aplicada em controle de processos não-lineares;
- Implementação do controle preditivo baseado em redes neurais no controle multivariável do processo de neutralização do *pH*;
- Realização de testes com o controlador proposto e verificar seu comportamento.

1.2 Organização do trabalho

O presente trabalho está organizado da seguinte forma:

- **Capítulo 2: Inteligência Artificial** - Apresenta as redes neurais artificiais, técnica recentemente utilizada na construção de controladores. Aborda suas propriedades a fim de fornecer subsídio para sua utilização em sistemas de controle. Outros conceitos ligados às RNAs são apresentados, como algoritmo de treinamento, arquitetura, generalização, capacidade de incorporar dinâmicas, entre outros.
- **Capítulo 3: Neuro-controladores** - Aborda o controle neural de sistemas, apresentando controladores baseados em modelo. Aborda estratégias de controle que fazem uso das RNAs como modelo do processo. Quanto a problemas de otimização, mostra conceitos e alguns passos a serem levados na geração de uma função de custo. Aborda ainda o método de otimização de Newton-Raphson que é utilizado neste trabalho de pesquisa.
- **Capítulo 4: Planta piloto e neuro-controlador utilizado** - Aborda a planta piloto de neutralização de *pH* e o neuro-controlador utilizado no controle do *pH*. Mostra os passos utilizados na obtenção do modelo neural e aplicação deste modelo na arquitetura do neuro-controlador. Apresenta ainda as equações utilizadas pelo neuro-controlador.
- **Capítulo 5: Estudo de casos, resultados e discussões** - Apresenta os testes realizados com o neuro-controlador proposto tanto no sistema simulado da planta piloto como na planta piloto de neutralização de *pH* e apresenta os resultados obtidos.

- **Capítulo 6: Conclusões e sugestões para trabalho futuros** - Apresenta conclusões sobre a utilização do neuro-controlador proposto e também sugestões para possíveis trabalhos futuros.

Capítulo 2

Inteligência Artificial

2.1 Introdução

De uma vasta lista de definições a respeito de Inteligência Artificial - IA[1], pode-se resumidamente dizer que IA é a parte da ciência que busca continuamente a implementação em computadores, de tarefas onde os seres humanos são melhores. O motivo de tal pensamento pode estar no interesse de desenvolver máquinas que operem incansavelmente em contínua produção a fim de aumentar produtividade e qualidade, evitando erros, ou ainda em automatizar tarefas e ampliar a velocidade de produção. Os motivos na realidade são inerentes ao interesse de cada área.

No lado filosófico da IA, as perguntas sempre incentivaram o avanço das pesquisas: Pode uma máquina agir inteligentemente? Pode uma máquina resolver qualquer problema que o ser humano poderia resolver? Pode a máquina sentir e ter consciência? O cérebro humano pode ser realmente representado em sua totalidade apenas por conexões?

Nesta filosofia acredita-se que o ser humano sempre esteve inserido. Sempre pensou a respeito de IA e da sua real possibilidade e do que aconteceria se acaso chegasse a um patamar de igualdade entre homem e máquina. Neste pensamento, pode-se observar abaixo um trecho extraído do livro *The Book of Lieh-tzu: A Classic of the Tao*, do século 4 A.C.:

> "*Quem é o homem que te acompanha? perguntou o rei. Este, senhor, respondeu Yen Shih, é minha obra prima. Ele pode cantar e atuar. O rei encarou a figura em*

[1] Inteligência Artificial é o nome mais comumente citado quando se quer falar de Inteligência Computacional (IC).

> *espanto... não poderia crer que não era real. Quando a apresentação estava se dirigindo ao fim, o robô piscou seu olho e encarou as donzelas com atenção, daí o rei se enfureceu..., instantâneamente deixou o robô em peças para ver o que ele realmente era.*"

Na realidade, este pensamento sempre esteve intrínseco à natureza criadora do homem. O sucesso de pesquisas hoje se deve ao fato de que o homem é um ser com características de criador. Algo intrínseco ao homem o leva a pesquisar, criar e desenvolver. Baseado nesta capacidade criadora, pode-se afirmar que o homem sempre pensou em IA, mesmo talvez não sabendo o que isto muitas vezes significava.

Com o passar dos anos, com as descobertas e avanços em IA, começou-se a desenvolver novas tecnologias e estratégias que fizessem uso de IA para cumprir a meta do desenvolvimento e criação de seres inteligentes (agentes com capacidade de interagir com o meio de forma inteligente), apresentando capacidade de auto adequação.

Em meados do século passado, mais precisamente em 1950, Turing (1950) definiu o problema de inteligência em uma conversação simples, a qual deu o nome de *Imitation Game*. Ele sugeriu que se uma máquina pode responder a quaisquer questões apresentadas a ela, fazendo para isso uso das mesmas palavras que uma pessoa faria, então esta máquina poderia ser considerada uma máquina inteligente. É o pensamento de que, se uma máquina age com a mesma inteligência que um ser humano, então ela é tão inteligente quanto. O teste de Turing gera controvérsias até hoje (Shieber, 2007; Rapaport, 2006).

Alguns anos depois apareceu o *Logic Theorist* de Newell et al. (1957) como uma tentativa inicial de provar teoremas matemáticos usando heurística como o homem faz. Mais tarde, Samuel (1959), fez um trabalho de IA que envolvia um jogo de damas. Ele escreveu um programa de jogo de damas que não apenas jogava com oponentes, mas que segundo ele, fazia-se uso da experiência adquirida para melhorar posteriormente seu desempenho.

A IA teve grande crescimento com a criação da linguagem de programação LISP (*List Processing Language*)[2]. Pode-se dizer que passos importantes na IA foram dados com conceitos introduzidos por Minsky (1963).

Focando-se dentro da IA para o lado físico, uma máquina considerada inteligente interage com o meio onde está inserida recebendo a nomenclatura de *agente inteligente* (AI). Um AI é um sistema ou processo que atua inteligentemente, ou seja, que faz o apropriado para realizar suas metas sob variadas circunstâncias, sendo flexível à mudanças no ambiente e flexível para

[2]Linguagem estruturada em 1958, sendo a segunda linguagem de programação mais antiga, depois do Fortran.

mudar as suas próprias metas, aprendendo com a experiência e fazendo escolhas apropriadas que são limitadas e baseadas em seu conhecimento.

A meta principal da IA é entender os princípios que fazem o comportamento inteligente ser possível, tanto em sistemas naturais como artificiais. E a engenharia por trás disso tem como meta principal o projeto de artefatos inteligentes úteis.

Com o avanço da pesquisa acerca de IA, novas técnicas para tratar de quantidades maiores de conhecimento foram desenvolvidas, com progresso considerável. Atualmente a IA tem conquistado espaço muito maior do que antes nas mais diversas áreas. Segue alguns dos problemas enquadrados dentro da IA:

- Jogos [3];
- Diagnóstico médico;
- Análise química;
- Projetos de engenharia;
- Resolução de problemas gerais e especializados;
- Percepção, Visão e Fala (compreensão de linguagem natural).

Muitas técnicas de IA podem ser empregadas na solução desses problemas. Algumas se tornaram mais específicas a determinados assuntos, outras ainda não receberam atenção ou ainda não foram pesquisadas à fundo.

Dentre os mais variados ramos da inteligência artificial, pode-se citar os seguintes:

- Sistemas Especialistas;
- Processamento de Linguagem Natural;
- Algoritmos Inteligentes;
- Algoritmos Genéticos;
- Lógica Fuzzy;
- Redes Neurais Artificiais (ramo da IA utilizada neste trabalho e abordado neste capítulo).

[3] Atualmente, praticamente todos os jogos produzidos possuem alguma programação baseada em IA.

2.2 Redes neurais artificiais

Ao falar de RNAs, pode-se indagar o quanto é útil esta tecnologia. O avanço tecnológico industrial tem tomado uso dessa ferramenta nos processos industriais. A conotação artificial é uma diferenciação às redes neurais biológicas onde esta tecnologia está baseada.

Neste capítulo seram apresentados os fundamentos e os conceitos inerentes a esta tecnologia, como também algumas caracterizações quanto à estrutura e operação, e algumas aplicações típicas.

Basicamente, uma RNA é uma cadeia interligada de neurônios artificiais com capacidade de adquirir conhecimento por meio de um aprendizado, podendo apresentar boa capacidade de generalização e excelente percepção de não-linearidades dos dados, conforme a topologia utilizada.

Segundo Braga et al. (2007):

> *"RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (neurônios artificiais) que calculam determinadas funções matemáticas (normalmente não-lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande numero de conexões, geralmente unidirecionais."*

A RNA pode ser vista como uma máquina que se adapta às condições de um meio. Esta máquina se assemelha ao cérebro em dois aspectos: a aquisição de conhecimento se dá por um processo de aprendizado e o armazenamento de conhecimento se dá através do ajuste dos pesos das conexões internas entre os elementos (neurônios). O processo de aprendizado é também chamado de treinamento e o valor das conexões entre os neurônios é chamado de peso sináptico.

Uma rede neural artificial pode ser estruturada a partir de componentes eletrônicos ou simulada através de *software* e, para funcionar eficientemente, faz uso de intensa interconexão entre suas unidades de processamento denominadas neurônios. Portanto, falar de conexões quando se trata de redes neurais, não é dizer que devam existir fisicamente. Em um vasto número de aplicações as conexões não existem fisicamente, são meras terminologias. As aplicações são basicamente feitas por meio de *softwares* através de diferentes linguagens de programação.

Dentre algumas propriedades e capacidades das redes neurais, pode-se citar:

- **Não-Linearidade** - Um neurônio é basicamente um dispositivo não-linear. E sendo um conjunto de neurônios, uma rede neural é essencialmente um sistema não-linear. Esta

propriedade é extremamente importante, pois o mundo real é predominantemente não-linear;

- **Adaptatividade** - As redes neurais têm a capacidade intrínseca de se adaptar para representar mudanças no ambiente que as envolve. Essencialmente, uma rede treinada em um ambiente específico, pode ser retreinada para ser mais sensível às mudanças nesse ambiente. Ou ainda, uma rede neural pode ser estruturada de forma a mudar os seus próprios pesos sinápticos a fim de se adaptar automaticamente às mudanças no sistema em que está acoplada;

- **Mapeamento Entrada-Saída** - O aprendizado supervisionado envolve a modificação dos pesos sinápticos através da aplicação de um conjunto de dados. Esses dados consistem de pares de sinais, um sinal de entrada e um sinal de saída desejado para a rede. A partir disso, a rede entra num processo de aprendizado que visa minimizar a diferença entre a saída da rede e a saída desejada. Assim a rede aprende dos exemplos;

- **Uniformidade para analisar e projetar** - Neurônios são comuns a todas as redes neurais. Redes neurais são facilmente montadas a partir da integração de módulos. Essas características são importantes quando se quer montar uma rede para atacar um problema específico, pois as diversas arquiteturas de redes neurais possíveis só diferem na maneira com que os seus neurônios estão agrupados;

- **Tolerância à falhas** - Uma rede neural é tolerante a falhas. Desde que esta sofra algum dano em algum neurônio (como mudança do valor do peso sináptico), devido a sua natureza de ter as informações paralelamente distribuídas, esse dano acarreta uma degradação suave do desempenho;

- **Resposta a evidências** - Quanto à classificação de padrões, uma rede neural pode ser treinada para não apenas selecionar um padrão específico, mas também para fornecer uma confiança pela seleção daquele padrão;

- **Sistemas multivariáveis** - Por natureza a RNA é um sistema que processa múltiplas entradas e múltiplas saídas, sendo por isso muito aplicada com eficácia em sistemas MIMO;

- **Analogia Neurobiológica** - O projeto de uma RNA é motivado pela analogia com o cérebro, prova viva de que o processamento paralelo tolerante a falhas pode ser implementado. Assim, os engenheiros procuram na neurobiologia novas idéias para resolver problemas de maior complexidade que aqueles baseados em técnicas convencionais.

2.2.1 Modelo de um neurônio artificial

Os neurônios artificiais são unidades de processamento que tentam simular a estrutura e a função dos neurônios biológicos. Na Figura 2.1 é apresentado o modelo de um neurônio artificial.

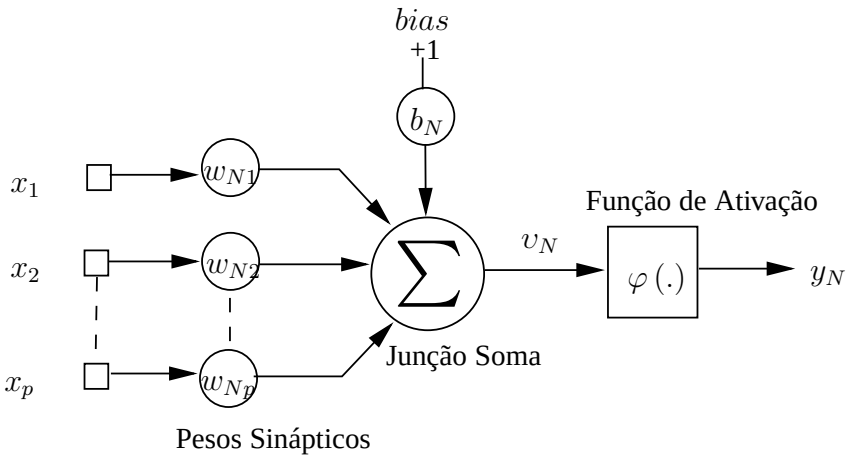


Figura 2.1: Um modelo não-linear de um neurônio.

Pode-se identificar neste modelo os seguintes elementos:

- Os vários sinais de entrada x_p;
- Um conjunto de sinapses que são caracterizadas cada uma por um peso w_{Np};
- Um somador que realiza o somatório do produto do sinal de entrada pelo peso da sinapse correspondente (combinador linear);
- Uma função de ativação $\varphi(.)$ (geralmente não-linear) que limita a amplitude do sinal de saída do neurônio;
- O *bias* b_N que aumenta ou diminui a entrada liquida da função de ativação, dependendo se o mesmo seja positivo ou negativo. O uso do *bias* b_N aplica uma transformação à saída do combinador linear;
- O sinal de saída y_N.

Um sinal de entrada x_p ao chegar na respectiva sinapse p do neurônio N recebe um peso w_{Np} e então segue ao somador onde ocorre uma combinação linear dos valores de todas as entradas pelos seus respectivos pesos tendo como valor de saída

$$v_N = b_N + \sum_{j=1}^{p} w_{Nj} x_j \tag{2.1}$$

Esse novo sinal combinado passa então pela função de ativação tendo como resposta a saída

$$y_N = \varphi\left(v_N\right) \tag{2.2}$$

O primeiro modelo de neurônio artificial foi criado por McCulloch e Pitts (1943) e conhecido como o modelo de McCulloch-Pitts onde o ponto principal era a modelagem de um neurônio biológico e o uso da capacidade computacional para executar funções booleanas. O modelo substituía os dendritos por entradas e estabelecia que as ligações destas entradas com o corpo celular artificial seriam realizadas através de elementos chamados de pesos, que multiplicados às respectivas entradas simulariam sinapses.

Deste modo, uma sinapse de maior eficiência seria representada pela associação de um peso de maior valor à sua entrada. Além disso, os estímulos captados pelas entradas seriam processados em uma função de soma, sendo que o resultado seria alimentado à uma função de ativação estabelecendo uma saída.

2.2.2 Funções de ativação

Em termos básicos, quando se fala em função de ativação, a diferenciabilidade desta é uma única exigência a ser satisfeita. De acordo com Haykin (1999) para que haja essa diferenciabilidade, a função deve ser contínua.

A função de ativação $\varphi\left(.\right)$ define a saída do neurônio com base no seu nível de ativação interna ou campo local induzido v_N. Dentre os tipos de funções de ativação, as duas mais comumente utilizadas são:

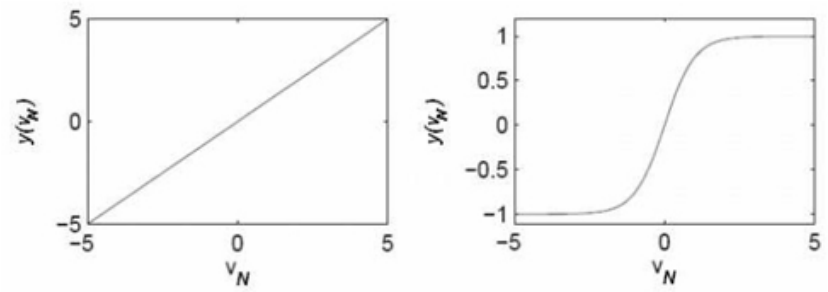
1. **Função linear** - vista na Figura 2.2(a), a saída do neurônio é dada pela Equação 2.3

$$\varphi(v) = \alpha v, \tag{2.3}$$

sendo α um número real que define a saída linear para os valores de v.

2. **Função sigmóide tangente hiperbólica** - vista na Figura 2.2(b), a saída do neurônio é dada pela Equação 2.4

$$\varphi(v) = \frac{1 - e^{-2v}}{1 + e^{-2v}}. \tag{2.4}$$



(a) (b)

Figura 2.2: Funções de ativação: (a) função linear, (b) função sigmóide tangente hiperbólica.

Para a função de ativação tangente hiperbólica, foi considerada a faixa de variação entre $\pm$ 1, no entanto pode ser desejável para algumas situações que a variação esteja definida entre 0 e 1.

2.3 Principais Arquiteturas das RNAs

De forma a padronizar as terminologias referentes às camadas da rede neural, neste trabalho serão adotadas as seguintes terminologias, a mesmas utilizadas em Haykin (1999):

- **Camada de entrada** - ou também "camada de entrada de nós de fonte". É assim chamada a camada de entrada da rede pela qual é fornecida o vetor de entrada de dados, que constituem os sinais de entrada aplicados aos neurônios (também conhecidos como "nós computacionais");

- **Número de camadas** - o número de camadas será definido pelo número de camadas de neurônios ou nós computacionais. Sendo a camada de entrada não contada, devido na mesma não ser realizada nenhuma computação (ao mencionar "uma rede *feedforward* de uma camada", subentende-se que é uma rede com uma camada de neurônios);

- **Camada Oculta** - camada ou camadas situadas antes da camada de saída e depois da camada de entrada de nós de fonte (camada de entrada);
- **Camada de Saída** - como o número de saídas é o número de neurônios da última camada, então a camada de saída deverá ser subentendida como "camada de neurônios da saída da rede".

A representação artificial surgiu de uma inspiração biológica entre as décadas de 1940 e 1970, com o trabalho pioneiro de McCulloch e Pitts (1943), até a demonstração da impossibilidade do *perceptron* de uma camada (*perceptron* simples como ficou conhecido), de resolver problemas não-linearmente separáveis (Minsky e Papert, 1969).

As RNAs normalmente são divididas em duas classes: redes *feedforward* e redes *feedback*.

2.4 Redes *Feedforward*

Uma rede neural *feedforward* é entendida como um conjunto de neurônios interligados onde o fluxo da informação segue somente no sentido direto, das entradas para as saídas. Os neurônios que realizam o cálculo final, cujas saídas são as saídas da própria rede, são chamados de neurônios de saída, ou camada de saída, e os neurônios da camada oculta são chamados neurônios ocultos. Na Figura 2.3 pode ser vista uma rede neural com uma única camada.

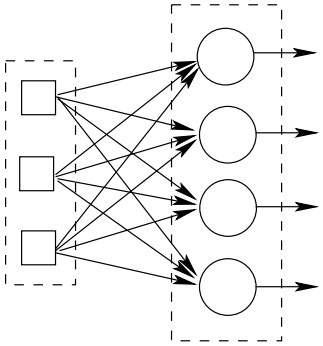


Figura 2.3: Rede alimentada adiante com uma única camada.

As redes de múltiplas camadas obtêm vantagem sobre as redes neurais de uma camada na resolução de problemas[4], isso porque os neurônios das camadas ocultas possuem a capacidade de extrair dos dados as informações de ordem mais elevada: cada neurônio oculto é responsável por produzir um hiperplano particular e as fronteiras de decisão destes hiperplanos são

[4]Em quase sua totalidade os problemas cotidianos apresentam não-linearidades

combinadas linearmente pelos neurônios da camada de saída (Haykin, 1999). Isso possibilita a extração de características não-lineares intrínsecas que podem passar despercebidas por outras técnicas de modelagem. Exemplo de uma rede de múltiplas camadas pode ser visto na Figura 2.4.

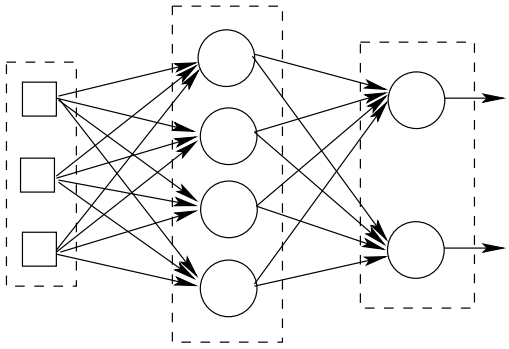


Figura 2.4: Rede *feedforward* com uma camada oculta - exemplo de uma rede de múltiplas camadas.

Em algumas publicações literárias a camada de entrada da rede é contada na estrutura da mesma. Isso pode ser um tanto confuso, já que esta não contém neurônios e portanto não há nenhum processamento dos dados. Conforme Haykin (1999) não se conta a camada de entrada, definida camada de entrada de neurônios de fonte, pelo motivo de não ser realizada qualquer computação na mesma.

Redes *feedforward* são estáticas; se as entradas são constantes as saídas assim o são. Assim redes neurais *feedforward* são denominadas redes estáticas em contraste com redes neurais *feedback* ou dinâmicas. Redes neurais *feedforward* com funções não-lineares sigmóides são denominadas redes perceptron multicamadas (*Multilayer Perceptron* - MLP).

2.5 Redes *Feedback* (Recorrentes)

Redes *feedback* ou *Recurrent Networks*, independente do termo utilizado, são redes neurais onde a saída de um neurônio na camada seguinte é utilizada como entrada para o neurônio na camada anterior, ou mesmo para o próprio neurônio.

De acordo com Elman (1990), para um RNA ser considerada dinâmica é preciso que possua memória:

> "*Existe uma outra possibilidade muito diferente, que é permitir o tempo ser representado pelo efeito que ele tem no processo. Isto significa dar ao processamento*

> *da dinâmica do sistema propriedades que são responsivas a seqüências temporais. Em outras palavras, para a rede deve ser dada memória.*"

De acordo com Braga et al. (2007), as formas de prover memória para uma RNA são a introdução de um operador de atraso ou o uso de redes recorrentes, como *back-propagation Through Time, Real-Time Recurrent Learning, Cascade Correlation Recorrente*, redes de Elman e redes de Jordan.

- **Modelo recorrente de entrada-saída**

Para lidar com variações temporais, estabelece-se uma RNA que utiliza atrasos de tempo. Basicamente a técnica consiste de uma rede *feedfoward* onde as entradas e saídas são atrasadas no tempo e realimentam a rede neural.

Ao se considerar valores passados de entrada/saída, a capacidade de aprendizado da RNA é aumentada, permitindo a captura das dinâmicas de um sistema. A utilização do modelo recorrente entrada-saída é uma ferramenta indicada para aplicações práticas em controle de sistemas dinâmicos. De acordo com Haykin (1999), a arquitetura de uma rede recorrente genérica pode ser vista na Figura 2.5.

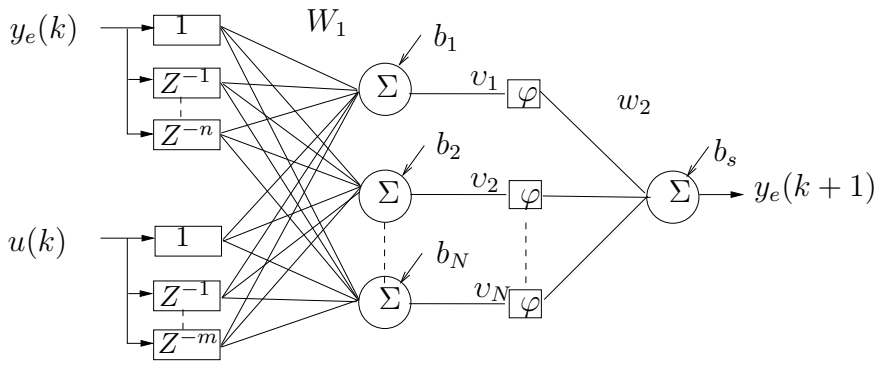


Figura 2.5: Arquitetura de uma rede recorrente genérica.

Na figura acima o número de neurônios da camada intermediária são representados por N, as polarizações de cada neurônio desta camada definidos por b_i, o bloco Σ representa o combinador linear e φ denota a função de ativação. O b_s denota a polarização do neurônio de saída da rede. Matrizes W_1 e w_2 representam os pesos da camada oculta e saída respectivamente.

2.6 Redes *perceptron* multicamadas

As não-linearidades estão presentes na maioria dos problemas reais o que faz ser recomendado o uso de estruturas com características não-lineares para a resolução destes problemas.

Através das funções de ativação dos neurônios e das sucessivas camadas da rede é que são incorporadas essas não-linearidades. Uma rede neural de múltiplas camadas composta por neurônios com funções de ativação sigmóides nas camadas ocultas recebe o nome de *Multilayer Perceptron* - MLP. Uma rede MLP típica pode ser vista na Figura 2.6.

Uma rede MLP composta de uma camada oculta tem a capacidade de permitir a aproximação de qualquer função contínua e com o uso de duas camadas permite a aproximação de qualquer função (Cybenco, 1989).

No treinamento de redes de múltiplas camadas, o procedimento de cálculo de erro aplicado a redes de uma única camada pode ser aplicado somente à camada de saída por não possuir saídas desejadas para as camadas ocultas. Para solucionar este problema de treinamento de MLPs, surge no meio da década de 1980 a descrição do algoritmo de retro-propagação de erros, ou *back-propagation* (Rumelhart et al., 1986).

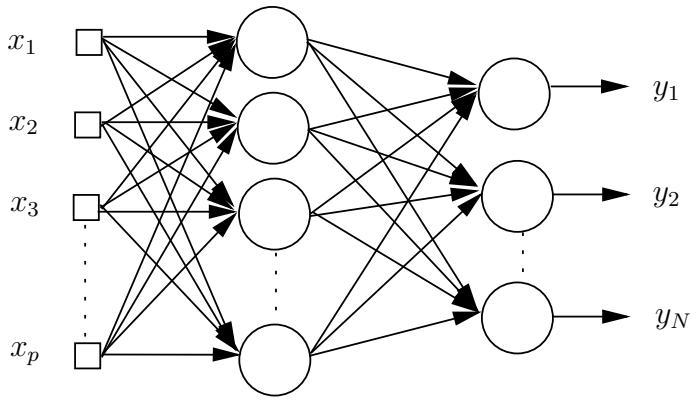


Figura 2.6: Uma típica rede MLP com uma camada oculta.

2.6.1 Algoritmo *back-propagation*

O algoritmo *back-propagation* ou algoritmo de retropropagação inicialmente proposto por Rumelhart et al. (1986) é o algoritmo mais utilizado no treinamento de RNA multicamadas do tipo MLP, independentemente do número de camadas escondidas. Ele utiliza pares entrada-saída desejada para ajustar os pesos da rede neural, por meio de um mecanismo de correção de

erro.

O ajuste de pesos é baseado na regra delta proposta por Widrow e Hoff (Widrow e Hoff, 1960). A generalização da regra delta para a aplicação em redes de múltiplas camadas é conhecida como regra delta generalizada (Rumelhart et al., 1986) ou *back-propagation*:

$$\Delta w_{Np}(k) = \eta \delta_N(k) x_p(k) + \alpha \Delta w_{Np}(k-1) \tag{2.5}$$

sendo,

$\Delta w_{Np}(k)$, a correção no peso do neurônio N na iteração k;

η, a taxa de aprendizagem;

$\delta_N(k)$, o gradiente local do N-ésimo neurônio na iteração k;

$x_p(k)$, o sinal da entrada x do N-ésimo neurônio na iteração k;

α, a constante de momento que pode tirar a rede de mínimos locais.

Como na regra delta, a função de custo a ser minimizada é uma função de erro ou também chamada de função de energia e é definida como a soma dos erros quadráticos.

O treinamento pelo algoritmo *back-propagation* ocorre por meio das etapas (Rumelhart et al., 1986):

- **Etapa de propagação** - nesta etapa é definida a saída da rede para um dado padrão de entrada, aqui os pesos não são alterados. O fluxo é no sentido direto, entrada-saída;
- **Etapa de retropropagação** - nesta etapa utiliza-se a saída desejada e a saída calculada pela rede na etapa anterior para o ajuste dos pesos sinápticos. O sentido do fluxo do sinal de erro é o contrário da etapa anterior.

A seguir algumas relações que tipificam o ajuste dos pesos da rede realizado pelo algoritmo de retropropagação do erro:

1. Cálculo da correção dos pesos da rede, vindo pela regra delta (Widrow e Hoff, 1960):

$$\Delta w_{Np}(k) = \eta \delta_N(k) x_p(k), \tag{2.6}$$

sendo:

$\Delta w_{Np}(k)$	Correção no peso do neurônio N na iteração k;
η	Taxa de aprendizagem;
$\delta_N(k)$	Gradiente local do neurônio N na iteração k;
$x_p(k)$	Sinal da entrada p do neurônio N na iteração k.

2. Cálculo do gradiente local:

   - Para um neurônio N na camada de saída:

   $$\delta_N(k) = e_N(k)\,\varphi'_N(v_N(k)), \tag{2.7}$$

   sendo $e_N(k)$ o erro da diferença entre a saída do neurônio N e a saída desejada na iteração k, $\varphi'_N(v_N(k))$ a derivada da função de ativação do neurônio N em relação à saída linear do neurônio, $v_N(k)$, na iteração k.

   - Para um neurônio N na camada oculta:

   $$\delta_N(k) = \varphi'_N(v_N(k)) \sum_p \delta_p(k)\, w_{pN}(k), \tag{2.8}$$

   sendo $\varphi'_N(v_N(k))$ a derivada da função de ativação do neurônio N em relação à saída linear do neurônio na iteração k e $\sum_p \delta_p(k)\, w_{pN}(k)$ a soma ponderada dos gradientes locais da camada seguinte na iteração k.

O aparecimento do termo $\varphi'_N(v_N(k))$ nas Equações 2.7 e 2.8 do cálculo do gradiente local, implica em que as funções de ativação utilizadas em uma rede MLP devam ser diferenciáveis. Para o desenvolvimento completo do algoritmo *back-propagation*, ver Haykin (1999) e Braga et al. (2007).

O algoritmo *back-propagation* tem sido utilizado e demonstrado bons resultados em muitas aplicações. No entanto ele apresenta alguns problemas como taxa de convergência lenta e possibilidade em ficar estagnado em um mínimo local. Um outro problema do algoritmo *back-propagation* convencional é que quando a saída desejada for 0 ou 1, e a saída da rede estando próxima destes valores, a derivada estará próxima de 0, fazendo com que os pesos recebam um ajuste praticamente zero. O problema da derivada próxima de zero pode ser eliminado pelo algoritmo *Resilient back-propagation* - RPROP (Riedmiller e Braun, 1993), uma variação do *back-propagation*.

2.6.2 Melhorando o desempenho do algoritmo *back-propagation*

Segundo Haykin (1999), mesmo sendo a experiência de cada projetista um ponto forte para melhoras de desempenho no *back-propagation*, este pode ser melhorado significantemente utilizando para isto, alguns métodos:

- **Abrangência da informação** - todo conjunto de treinamento apresentado ao algoritmo deve ser escolhido de forma a abranger a maior informação do problema estudado. Pode-se para isto buscar um exemplo que resulte no maior erro de treinamento, e usar um exemplo que seja radicalmente diferente de todos os outros usados anteriormente;

- **Quanto ao tipo de atualização** - o modo seqüencial de aprendizagem é computacionalmente mais rápido que o modo de aprendizagem por lote, ainda mais quando o conjunto de dados de treinamento é significantemente grande;

- **Função de ativação** - Com o algoritmo *back-propagation*, uma rede MLP pode aprender mais rápido quando a função de ativação sigmóide do neurônio for anti-simétrica do que quando ela for não-simétrica (LeCun et al., 1990);

- **Normalização** - A escolha de média zero e variância unitária não é tão irreal já que as entradas para uma MLP são frequêntemente normalizadas para ter média zero e variância unitária (Lawrence et al., 1998);

- **Resposta desejada** - é importante que a resposta desejada (valor-alvo) seja definida com base no intervalo da função de ativação sigmóide localizada na camada de saída. Se caso contrário, o algoritmo de retropropagação tende a levar os parâmetros livres da rede para infinito, reduzindo a velocidade de treinamento e levando a uma estagnação do treinamento.

2.7 Treinamento

A utilização de uma RNA para um determinado propósito está diretamente ligada à eficiência com que esta é capaz de fornecer respostas próximas o suficiente dos dados de saída reais. Portanto, o neurônio artificial deve ser capaz de aprender uma determinada tarefa.

Na fase de aprendizagem, a rede extrai informações relevantes de padrões de informação apresentados criando uma representação própria do problema. Nesta etapa há o ajuste dos

parâmetros da rede, que são representados pelos pesos das conexões entre as unidades de processamento, e o *bias*. Assim, ao final do treinamento a rede terá adquirido o conhecimento sobre o ambiente em que está operando.

Existem diversos métodos para realizar o treinamento de RNA, mas quando estas são aplicadas na modelagem de processos utiliza-se o aprendizado supervisionado (Figura 2.7) ou também chamado de aprendizagem com um professor. O objetivo deste método de aprendizagem é ajustar os parâmetros da rede de forma a estabelecer uma ligação entre os pares de entrada e saída fornecidos por um supervisor externo. O supervisor é responsável por dar a direção no processo de treinamento. A resposta obtida pela rede é comparada com a resposta desejada. A diferença existente entre estes dois valores representa o erro gerado pelo cálculo da rede, o qual deve ser minimizado através do ajuste dos pesos das conexões. A minimização do erro é incremental, pois a cada etapa de treinamento são efetuados pequenos ajustes nos pesos das conexões.

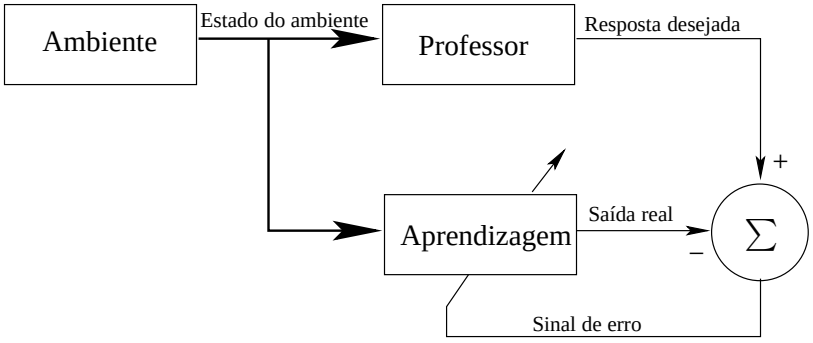


Figura 2.7: Método de aprendizagem supervisionada com professor.

2.8 Generalização da RNA

Por generalização se entende como a capacidade de uma RNA devidamente treinada de responder de forma coerente a padrões desconhecidos. Tais padrões podem ser entendidos como um conjunto de dados extraídos da mesma população utilizada no treinamento, ou seja, possuindo as mesmas características estatísticas. Esta capacidade de generalização não é uma propriedade implícita às RNAs. A capacidade de uma RNA em responder de forma satisfatória a padrões desconhecidos indica que esta possui capacidade de generalização elevada. Um trabalho baseado em um método Multi-Objetivo que garante a capacidade de generalização de uma RNA pode ser visto em Teixeira (2001).

Alguns fatores devem ser observados a fim de se obter uma RNA com elevada capacidade de generalização. Basicamente, alguns fatores influenciam na capacidade de generalização das RNAs, como o tamanho e representatividade estatística do conjunto de dados de treinamento, a arquitetura da rede neural, o número de épocas de treinamento e o número de parâmetros livres.

Não existem regras para a escolha do tamanho do conjunto de treinamento. Cada problema requer uma quantidade de amostras que seja capaz de representá-lo e esta escolha não é simples, ainda mais quando não se tem conhecimento à priori do domínio do problema.

Um dos maiores desafios no estudo da capacidade de generalização de uma RNA é a escolha da arquitetura apropriada do modelo neural em relação à complexidade do problema. Arquiteturas grandes elevam a complexidade do modelo. Quando esta complexidade é maior do que o necessário para a modelagem do problema, há um super-ajuste da rede aos dados de treinamento, o que acarreta em respostas não adequadas a padrões de validação e teste. Este fenômeno é chamado de *overfitting* e reduz a capacidade de generalização da rede.

No entanto, se a complexidade do problema supera a complexidade do modelo neural, é caracterizado o fenômeno de sub-ajuste ou *underfitting*. A Figura 2.8a ilustra um exemplo de boa generalização, enquanto que a Figura 2.8b ocorre o que se chama de generalização pobre. Um excesso de treinamento faz com que a rede perca a habilidade de generalizar bem. A ocorrência de *underfitting* e *overfitting* influenciam na capacidade de generalização das RNAs.

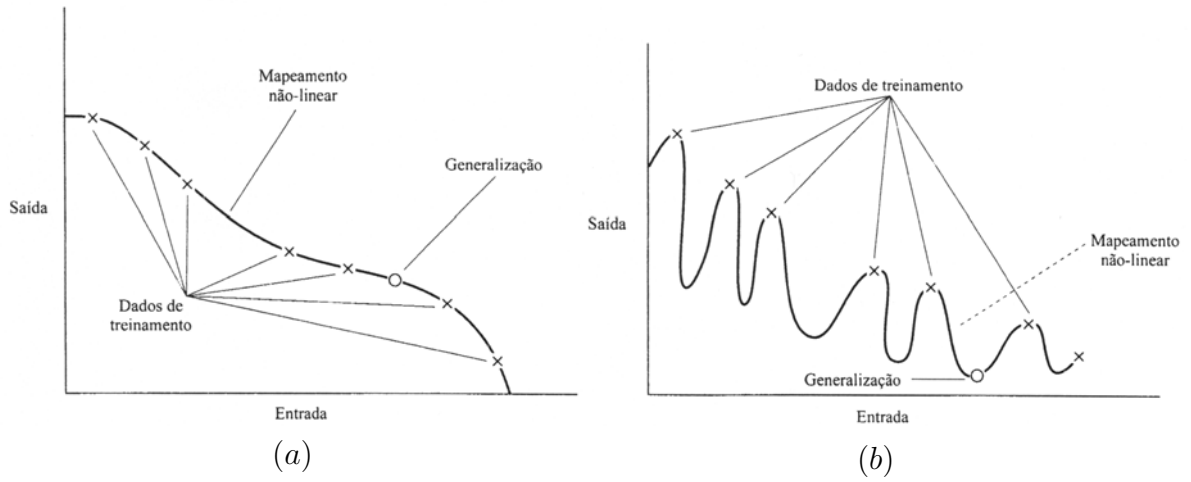


Figura 2.8: Generalização da RNA, (a) Dados adequadamente mapeados (b) Dados inadequadamente mapeados (generalização pobre). Fonte: (Haykin, 1999).

Existem algumas técnicas que têm como objetivo melhorar a generalização de uma RNA, fazendo para isso uso de algumas introduções de termos ou modificações na forma de treina-

mento. A adição de termos de penalidade na função objetivo é uma forma, e é utilizada pelo algoritmo *weight decay* (Hinton, 1989), a remoção de elementos que afetam menos a função de erro como os algoritmos *Optimal brain surgeon* (OBS) (Hassibi et al., 1996) e *Optimal brain damage* (OBD) (Le Cun et al., 1990) ou também como outras técnicas como interrupção de treinamento *Early Stopping* (Wang et al., 1994).

2.9 Comentários finais

Este capítulo teve como objetivo principal abordar as RNAs, tecnologia atualmente muito utilizada na construção de controladores. Foi abordada a suas propriedades e utilidades a fim de fornecer subsídio para sua utilização em sistemas de controle. As propriedades das RNAs, sobretudo sua capacidade de aprendizado e capacidade de se adaptar ao meio é o que motiva o seu uso em controle. Portanto alguns conceitos foram abordados, como algoritmo de treinamento, arquitetura, generalização, a capacidade de absorver dinâmicas, entre outros.

Na Seção 2.2 foi exposta uma introdução sobre as RNAs, abordando suas características. Na Seção 2.3 foi abordado as terminologias da arquitetura das RNAs e alguns conceitos básicos usados quando se fala de camadas. As Seções 2.4 e 2.5 tratam das duas classes em que as RNAs são normalmente divididas: redes *feedfoward* e redes *feedback*.

A Seção 2.6 abordou as redes MLP, que são as redes utilizadas no presente trabalho de pesquisa, escolhidas por serem aproximadoras universais de funções. A abordagem de treinamento *back-propagation* foi demonstrada por ser o algoritmo mais utilizado no treinamento de RNAs multicamadas do tipo MLP.

A abordagem do treinamento supervisionado e a generalização de RNAs foram abordados nas Seções 2.7 e 2.8, respectivamente. O treinamento supervisionado busca direcionar o processo de treinamento comparando a saída obtida com a saída desejada, e esta diferença deve ser minimizada durante o treinamento. A generalização serve como medida da RNA em responder a padrões desconhecidos de forma satisfatória.

Capítulo 3

Neuro-controladores

3.1 Introdução

Este capítulo trata das diferentes estratégias de controle baseadas em modelo, dando ênfase à utilização de RNAs no projeto de controladores.

Segundo Agarwal (1997), a RNA nestes sistemas de controle, é classificada quando à sua disposição, como sendo o próprio controlador ou como sendo o modelo do processo. Uma introdução às estratégias de controle é dada neste capítulo, a fim de situar e preparar o leitor para o neuro-controlador utilizado neste trabalho.

3.2 O controle neural de sistemas

As RNAs podem ser usadas em sistemas de controle como modelos do processo fornecendo informações como, por exemplo, a predição da saída do processo em k instantes de tempo à frente. Esta informação pode ser usada para tomar decisões antecipadamente. Quanto à sua disposição dentro do dispositivo controlador, a RNA pode ser classificada segundo Agarwal (1997) em duas situações:

- A RNA como o próprio controlador, em uma arquitetura direta. Basicamente nesta arquitetura, o controlador é do tipo simples realimentação;

- A RNA como modelo do processo, em uma arquitetura indireta.

Com o uso das RNAs em sistemas de controle, há o aparecimento dos termos Neuro-Controlador ou Controlador Neural. A RNA então tem seu comportamento definido de acordo com sua disposição na arquitetura do sistema ou dispositivo de controle.

Os neuro-controladores podem fazer uso de adaptação *on-line* ou apenas conterem um modelo neural do processo treinado *off-line*. Quanto a esta abordagem, o presente trabalho faz uso de um modelo neural com treinamento *off-line*.

Segundo Hunt et al. (1992), a habilidade das RNAs em representar mapeamento de dados e gerar modelos não-lineares é a principal característica observada ao se projetar controladores não-lineares.

Já dentro da forma de atuação dos neuro-controladores, estes podem ser classificados basicamente em controle direto ou controle indireto. É considerado que o controle é direto sempre que a rede neural atua como controlador, e controle indireto quando a rede neural é um modelo de predição de estados ou dinâmicas futuras.

No controle direto, o neuro-controlador procura imitar a ação humana. Uma ação no sinal de entrada da rede neural deve levar à entrada do processo um sinal de controle correspondente à ação humana, por isso denominado controle direto. Um exemplo simples de entender esta situação é imaginar como se quisesse diminuir o volume de áudio de um equipamento manipulando um potenciômetro. Na Figura 3.1 pode-se observar a arquitetura de um típico controle direto.

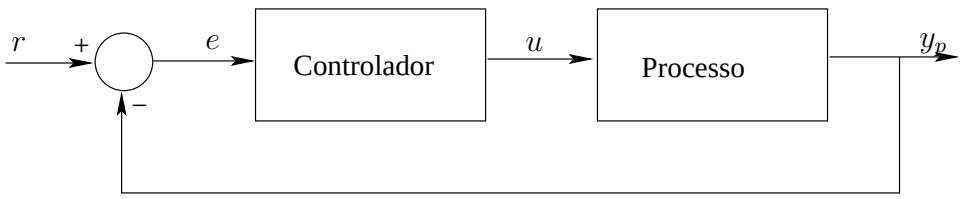


Figura 3.1: Controle direto. Realimentação convencional.

No controle indireto, há uma procura de um mapeamento direto entre a resposta desejada e a respectiva ação de controle necessária. Na Figura 3.2 pode-se observar o conceito básico de controle indireto. Para as Figuras 3.1 e 3.2, r é o sinal de referência, u a ação de controle, e o sinal de erro e y_p a saída do processo.

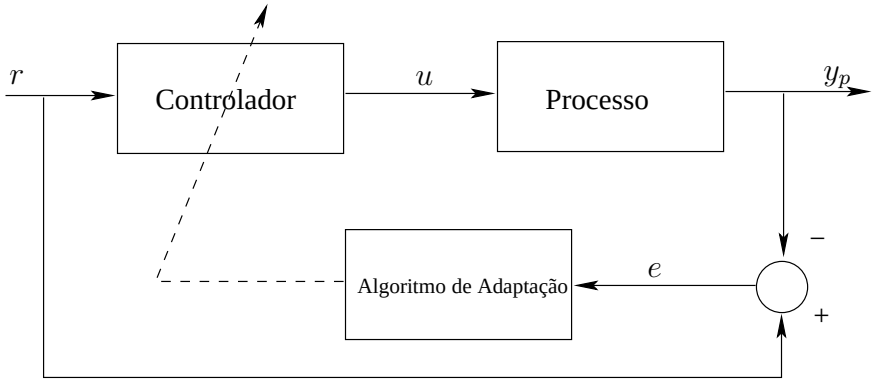


Figura 3.2: Conceito básico do controle indireto.

Existem várias estratégias de controle que são baseadas em algum tipo de modelo do processo. A maioria destas estratégias utiliza um modelo direto ou um modelo inverso do processo. Narendra e Parthasarathy (1990) afirma que muitas destas estratégias de controle baseadas em modelo podem empregar modelos a redes neurais, beneficiando-se das capacidades destas em captar dinâmicas do processo.

Os modelos neurais diretos possuem a característica de ter a informação dirigindo-se no mesmo sentido do processo, da entrada para a saída. O objetivo da modelagem direta é construir um modelo neural que, quando submetido à uma mesma entrada u aplicada ao processo, forneça uma saída y_e o mais fiel possível à saída y do processo. O modelo neural é colocado em paralelo com o processo e o treinamento da rede é realizado a fim de minimizar o erro entre a saída do processo e a saída da rede, como visto na Figura 3.3.

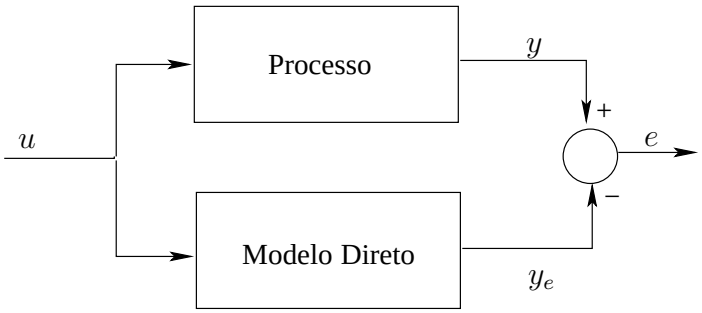


Figura 3.3: Estrutura da modelagem direta.

Diferentemente, modelos inversos possuem o fluxo da informação no sentido oposto do fluxo do processo. Um modelo inverso pode ser obtido utilizando a estrutura de treinamento

direta ou indireta. A abordagem mais simples é a aprendizagem inversa pelo método direto que pode ser vista na Figura 3.4.

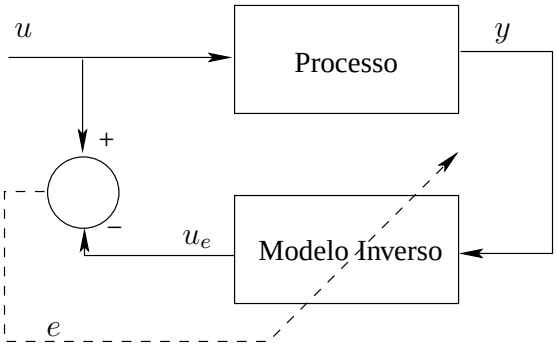


Figura 3.4: Estrutura da aprendizagem inversa pelo método direto.

A variável manipulada é aplicada ao processo e a saída do processo e valores passados da própria rede neural são utilizados como entradas da rede neural. A saída da rede u_e, como vista na Figura 3.4, é comparada com a entrada do processo produzindo um sinal de erro que é utilizado para o treinamento da rede neural. Este tipo de abordagem leva a rede neural a tentar representar o processo de forma inversa. Tem a desvantagem de não se poder manipular a saída do processo para cobrir uma faixa de operação desejada, já que a saída é função da entrada, tornando uma estratégia um pouco trabalhosa.

O problema com este tipo de estratégia de treinamento do modelo inverso pelo método direto, é que durante o treinamento o modelo tenta minimizar o erro da variável manipulada, o que não corresponde ao objetivo do controle que é minimizar o erro da saída do processo. O método de modelagem inversa especializada (Psaltis et al., 1988), sobressai neste sentido usando o erro da saída do processo. Nesta forma de abordagem o modelo neural inverso precede o sistema, e então recebe como entrada o sinal de treinamento que corresponde ao sinal de referência. Tem-se nesta estrutura a presença de um modelo neural direto do processo em configuração paralela. O sinal de erro é a diferença entre o sinal de treinamento e a saída do modelo direto. O erro pode ser retropropagado através do modelo neural direto, chegando ao modelo inverso onde os pesos são ajustados, como pode ser visto na Figura 3.5. Diferentemente do processo de treinamento direto do modelo inverso que somente aprende nas regiões definidas pelo projetista através da sequência da variável manipulada, o método de modelagem inversa especializado é bem direcionado a uma região de operação desejada.

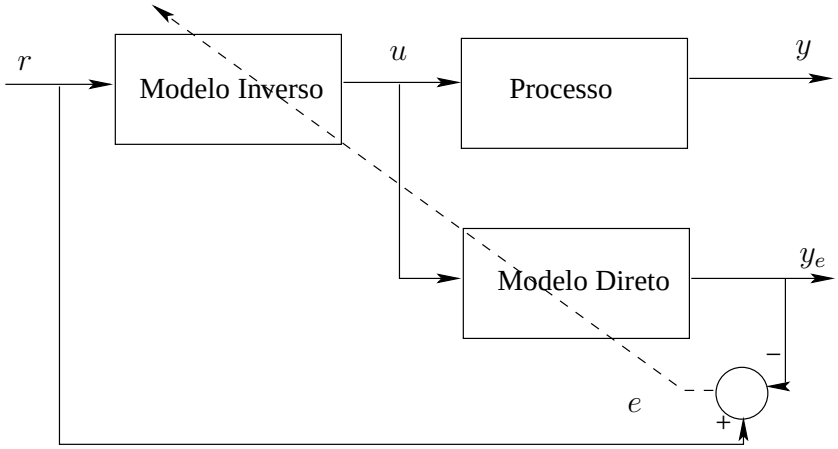


Figura 3.5: Estrutura da modelagem inversa especializada.

Existem várias estratégias utilizando redes neurais como controladores e modelos dinâmicos. Com o intuito de aplicar as redes neurais em estratégias de controle, Psaltis et al. (1988) propuseram diversas arquiteturas de aprendizagem denominadas arquitetura de aprendizagem geral, aprendizagem indireta e aprendizagem especializada. Bhat e McAvoy (1990) apresenta as redes neurais como ferramentas na modelagem dinâmica e sua utilização em controle de processos químicos, e ainda discute brevemente o uso de modelos neurais em controle e o aprendizado de dinâmicas inversas de processos.

Dentre as várias estratégias de controle que podem fazer uso das RNAs, apresenta-se a seguir as estruturas das estratégias de controle por modelo de referência (MRC), controle por modelo interno (IMC) e a classe de controladores baseados em modelo (MPC). A classe de controladores MPC será mais detalhada, pois dela se deriva o algoritmo de controle utilizado neste trabalho.

3.2.1 Controle por modelo de referência

Com a utilização de RNAs, esta estratégia de controle, também conhecida por *Model Reference Control* (MRC) incorpora um modelo neural direto e um controlador constituído de um modelo neural. O objetivo do controle é levar o processo a seguir a saída do modelo de referência. O erro e_2 entre o modelo de referência e a saída do processo é utilizado a fim de ajustar os pesos da rede controladora, da mesma forma que na estratégia de modelagem inversa especializada. O modelo direto é treinado com o erro e_1 entre a saída do processo e sua própria saída. É uma estratégia onde seu desempenho no controle depende muito do modelo de referência

escolhido. A estrutura de controle pode ser vista na Figura 3.2.1.

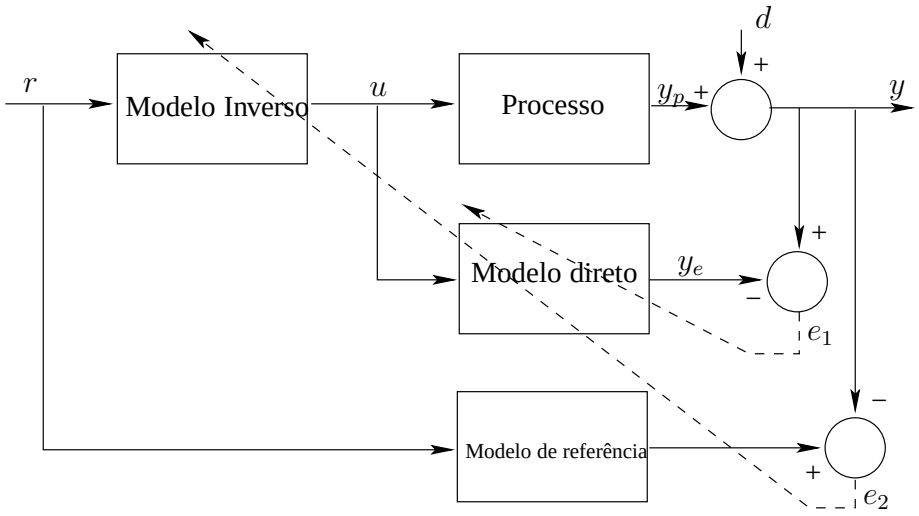


Figura 3.6: Estrutura de controle por modelo de referência

.

A utilização de RNAs com esta estratégia de controle e algumas aplicações e resultados pode ser visto em Hagan et al. (2002). A utilização de sistemas neurais *fuzzy* no projeto de um MRC para sistemas não-lineares pode ser visto em Pérez et al. (2009). Uma abordagem mais detalhada de MRC, incluindo as equações de controle pode ser vista em Chow e Li (2009).

3.2.2 Controle por modelo interno

O controle por modelo interno ou *Internal Model Control* (IMC) é uma terminologia muito confundida com outras estratégias de controle de sistemas que utilizam modelos internos, e pela confusão será adotada a terminologia IMC. Foi uma estratégia introduzida por Garcia e Morari (1982) e provada em sua robustez e estabilidade. O uso de RNA na estrutura IMC foi proposto por Econmou et al. (1986) e é mostrada na Figura 3.7.

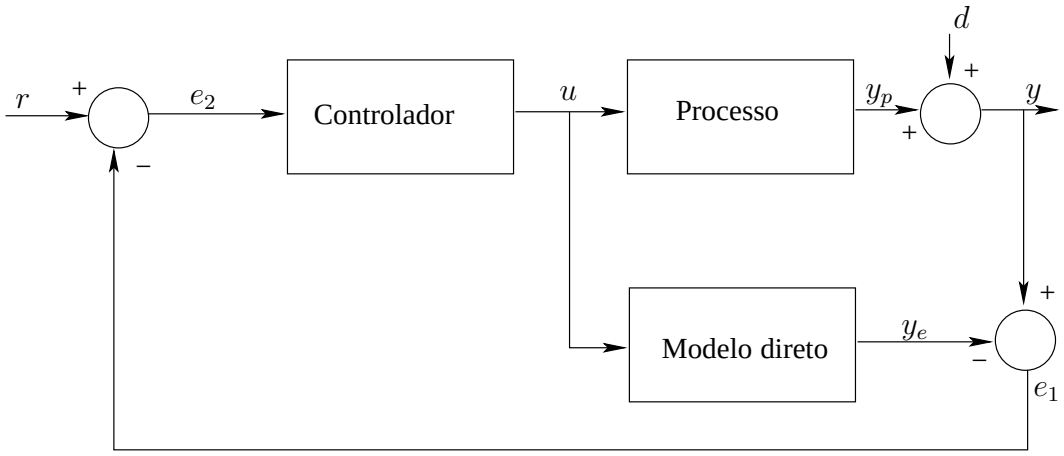


Figura 3.7: Estrutura de controle IMC.

Na Figura 3.7, d é uma perturbação desconhecida afetando o sistema. A variável u está ao mesmo tempo sendo aplicada ao processo e ao modelo do processo denominado modelo direto. O erro e_1 é resultante da comparação da saída do processo y com a saída do modelo y_e. O bloco controlador é usualmente definido como a inversa do modelo direto. Se considerar que y_e = y, pode-se mensurar a perturbação d no erro e_2. Um filtro pode ser colocado em série com o controlador e tem como propósito a redução da sensibilidade do sistema de controle em malha fechada para erros entre o processo e modelo (Hunt et al., 1992). A dificuldade com a estratégia de controle IMC é que o controlador sendo baseado na inversa do modelo do processo, necessita de um método eficiente e confiável para alcançar esta inversão (Maksumov et al., 2002).

3.3 Controle preditivo baseado em modelo

O controle preditivo baseado em modelo, definido neste trabalho como *Model Predictive Control* (MPC) não designa uma estratégia de controle específica mas uma ampla quantidade de métodos de controle. Estes métodos de controle, que podem ser chamados também por algoritmos ou métodos de controle MPC, fazem uso de um modelo do processo para obter um sinal de controle através da minimização de uma função de custo.

MPC se refere a uma classe de algoritmos que calculam uma sequência de variáveis manipuladas a fim de otimizar o comportamento futuro de um processo. Pelo tipo de modelo empregado, MPC é dividido nas categorias linear e não-linear. Logo, em esquemas MPC linear, modelos lineares são usados para predizer as dinâmicas do sistema. Uma visão geral das aplicações de MPC na indústria podem ser encontrados em Qin e Badgwell (1997, 2000). As

aplicações de MPC não-linear irão continuar a crescer nas áreas onde MPC linear têm dificuldades provadas de serem aplicadas, como em processo de polimerização (Findeisen et al., 2003).

Quando se fala em modelo, de acordo com Camacho e Bordons (1997), existem processos que apresentam não-linearidades tão severas que a representação por um modelo linear não é suficiente. Ainda, modelos não-lineares são difíceis de serem obtidos, seja pelo mapeamento de dados de entrada/saída, seja por leis de conservação de massa e energia.

A classe de controladores MPC pode também ser encontrada com outros nomes, mas designando as mesmas idéias:

- *Moving Horizon Control*;
- *Receding Horizon Control*;
- *Model Based Predictive Control*;
- *Long-Range Predictive Control*.

As idéias que aparecem em maior ou menor número em toda a classe MPC são basicamente:

- Um modelo da dinâmica do processo para predizer a saída em futuros instantes de tempo (horizonte de predição);
- Um histórico de valores de estados passados;
- Cálculo de uma sequência de controle minimizando uma função de custo. Uma função de custo J a ser minimizada sobre um horizonte de predição.
- Conceito do uso da estratégia de horizonte móvel (*"receding horizon"*), onde em cada instante de tempo o horizonte é deslocado para o futuro, o que envolve a aplicação do primeiro sinal de controle da sequência calculada em cada passo.

E os elementos comuns presentes na classe de algoritmos MPC são:

- Modelo do processo;

- Função de custo;
- Obtenção de uma lei de controle.

A Figura 3.8 mostra uma estrutura básica geral do MPC. O otimizador é usado para encontrar a ação de controle que minimiza a função de custo, que na sua forma mais simples é uma função quadrática da diferença entre a saída do processo e a referência.

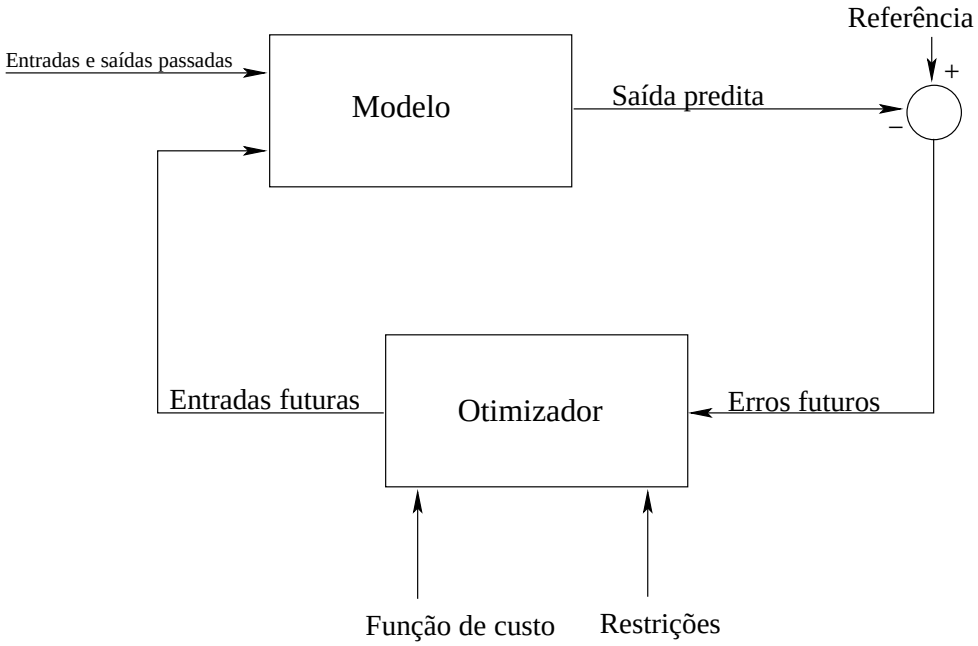


Figura 3.8: Estrutura básica geral do MPC.

De acordo com Camacho e Bordons (2004b), o MPC possui uma série de vantagens, mas também tem as suas desvantagens. Algumas das vantagens sobre outros métodos:

- Conceitos são intuitivos e ao mesmo tempo relativamente fáceis;
- Pode controlar uma grande variedade de processos, desde os com dinâmicas consideradas simples até os mais complexos, incluindo sistemas instáveis;
- A capacidade de tratar o caso multivariável com maior facilidade;
- Introduz o controle antecipado de uma forma a compensar pertubações mensuráveis;

- A forma de tratar restrições é conceitualmente simples;
- É muito útil quando referências futuras necessárias são conhecidas;
- É uma metodologia totalmente aberta que possibilita futuras extensões.

Também possui desvantagens, tais como: a derivação da lei de controle é mais complexa do que controladores convencionais; se a dinâmica do processo muda com o tempo, como no caso da utilização de um controle adaptativo, todos os cálculos precisam ser realizados em cada instante de tempo; quando se considera restrições, o esforço computacional exigido é maior, o que não deve ser um problema considerável atualmente; outra desvantagem é a necessidade de um modelo adequado do processo.

3.3.1 A estratégia MPC

Uma estratégia de controle vem caracterizar os controladores pertencentes à classe MPC. A estratégia está representada na Figura 3.9 e compreende os seguintes conceitos:

1. As saídas futuras para um horizonte de predição N_y são preditas em cada instante k usando o modelo do processo. Essas saídas preditas $y_e(k+j|k)$[1] para j= 1...N_y dependem dos valores de entradas e saídas passadas conhecidos no instante k e das ações de controle $u(k+j|k)$, j=0...N-1 aplicados ao sistema após a etapa de minimização.

2. O conjunto de ações de controle são calculados sob otimização, seguindo um critério determinado. O objetivo é manter o processo o mais próximo possível de uma trajetória definida. Normalmente o critério é uma função quadrática simples, que é a diferença entre a saída predita e a trajetória de referência. O esforço de controle também pode ser inserido na função.

3. A ação de controle $u(k|k)$ é aplicada ao processo e as outras próximas ações de controle são rejeitadas. Então é novamente realizado uma amostra do estado atual do processo, e cálculos são repetidos (passo 1) produzindo uma nova ação de controle e um novo vetor de estados preditos. O horizonte de predição sofre deslocamento caracterizando o conceito de horizonte móvel ("*receding horizon*").

[1] Esta notação é adotada neste trabalho para designar o valor da variável no instante k+j calculada no instante k.

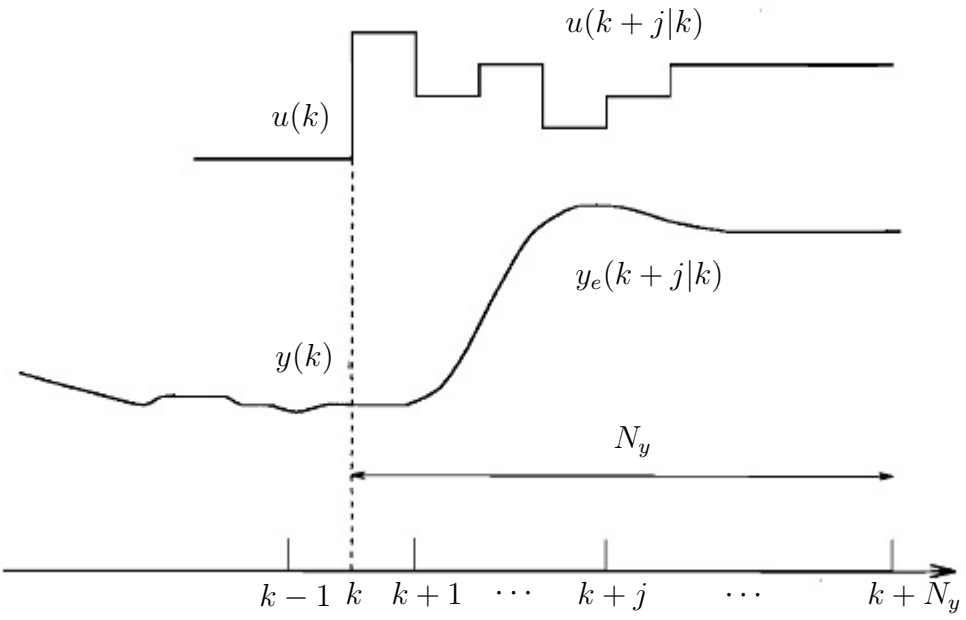


Figura 3.9: Estratégia de controle do MPC

Dentro da classe de algoritmos MPC, os métodos considerados mais representativos de acordo com Camacho e Bordons (2004b), são:

- **Controle por matriz dinâmica** - conhecido na literatura por *Dynamic Matrix Control* - DMC (Cutler e Ramaker, 1980).
- **Controle heurístico com modelo preditivo** - encontrado na literatura como *Model Predictive Heuristic Control* - MPHC ou *Model Algorithmic Control* - MAC (Rouhani e Mehra, 1982).
- **Controle funcional preditivo** - conhecido como *Predictive functional control* - PFC (Richalet et al., 1997).
- **Controle auto-adaptativo com predição extendida** - com o nome de *Extended Prediction Self-Adaptive Control* - EPSAC, foi apresentado por De Keyser e Van Cauwenberghe (1985).
- **Controle adaptativo com horizonte extendido** - conhecido por *Extended Horizon Adaptive Control* - EHAC (Ydstie, 1984).

- **Controle preditivo generalizado** - apresentado por Clarke et al. (1987), encontrado sob o nome de *Generalized Predictive Control* - GPC. A utilização de RNAs no algoritmo GPC é a abordagem empregada neste trabalho e será mais detalhada no próximo capítulo. O controle preditivo generalizado é caracterizado a seguir.

3.3.2 Controle preditivo generalizado

O controle preditivo generalizado, conhecido na literatura por GPC, foi introduzido por Clarke et al. (1987) e tornou-se um dos métodos de controle mais conhecidos dentro da classe MPC. Segundo Clarke et al. (1987), o GPC possui robustez em relação a erros de modelagem e foi originalmente desenvolvido com modelos lineares. Se um modelo não-linear for utilizado há a necessidade de um algoritmo de otimização não-linear.

Em funções de custo mais simples, o índice a ser otimizado é uma função quadrática que mede a distância entre a saída predita e uma sequência referencial definida. O algoritmo GPC tem como idéia básica o cálculo de uma sequência de futuras ações de controle de modo a minimizar a função de custo definida dentro de um horizonte de controle. A ponderação do esforço de controle é utilizada na função de otimização ou função de custo J. O índice J do GPC a ser minimizado pode ser expresso como:

$$J = \sum_{j=N_1}^{N_y} [r(k+j) - y_e(k+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(k+j)]^2 \tag{3.1}$$

sendo,

$r(k)$ a referência a ser seguida;

Δu a atualização no valor da ação de controle [$u(k+j)$ - $u(k+j-1)$];

λ a ponderação à ação de controle, sendo utilizado como limitador a variações bruscas das ações de controle, objetivando suavização nas variações destas;

N_1 o horizonte mínimo de predição;

N_u o horizonte de controle;

N_y o horizonte máximo de predição, sendo neste trabalho N_y = N_u = 1.

O algoritmo GPC tem a característica de poder lidar com plantas instáveis e incorpora o conceito de horizonte de controle, além da ponderação dos incrementos de controle na função

de custo, e é adequado para aplicações de alto desempenho tais como o controle de sistemas flexíveis (Clarke et al., 1987).

Neste trabalho se fez uma implementação do GPC com a utilização de redes neurais MLP como modelo do processo. A utilização de redes neurais leva o algoritmo GPC a ser denominado "controle preditivo neural generalizado"ou *Neural Generalized Predictive Control* - NGPC, o que vem a caracterizar o MPC não-linear devido ao uso de um modelo neural não-linear do processo.

3.4 Formulação da otimização

Uma expressão matemática denominada função objetivo é responsável por definir soluções na otimização de um determinado processo. As limitações ou restrições também fazem parte desta função, servindo como limitadores destas possíveis soluções.

Sendo necessário definir uma expressão que correlacione as variáveis do processo, esta expressão envolve basicamente uma função de custo J e as restrições de igualdade e de desigualdade, se necessárias. A função objetivo, também conhecida como função de custo, é a função que deverá passar pelo processo de otimização[2].

A meta do processo de otimização é encontrar os valores das variáveis no processo que produzem o melhor desempenho em um critério. Para isso alguns passos devem ser seguidos na otimização de processos e envolvem (Edgar e Himmelblau, 2001):

1. A análise do processo com o levantamento das variáveis de interesse;
2. A determinação do critério de otimização, levando à construção da função de custo J;
3. O desenvolvimento do modelo do processo;
4. A determinação de todas as relações de igualdade e/ou desigualdade;
5. Analisar a complexidade do problema e, se for necessário, o dividir em partes ou simplificar o modelo ou a função de custo;
6. A aplicação da técnica de otimização;

[2]Subentende-se como otimização o processo de minimização ou maximização da função de custo

7. A análise do resultado com os coeficientes adotados ou suposições feitas. Subentende-se aqui que se os resultados não forem satisfatórios, parte-se para a análise dos passos anteriores novamente.

Em problemas de otimização, muitos destes podem ser formulados em termos de uma variável. Outros entretanto devem ser formulados em termos de mais variáveis, daí os termos Otimização Unidimensional e Otimização Multidimensional. Em termos da forma com que acontece a computação da função de custo, esta pode ser dividida conforme o método usado na computação, sendo:

- **Método Direto** - utiliza apenas a computação da função de custo. Assim, o valor da função é comparada em vários pontos de avaliação na busca pelo extremo;
- **Método Indireto** - utiliza-se de uma condição necessária para um extremo. A expressão analítica da derivada é utilizada na computação, além do uso da computação da função objetivo.

Dentro destas duas categorias, as classificações dos métodos de otimização mais comumente conhecidas são (Edgar e Himmelblau, 2001):

- Otimização Unidimensional Sem Restrições

1. Métodos Indiretos
    - Método de Newton
    - Método de Quasi-Newton
    - Método da Secante
2. Métodos Diretos
    - Método de Eliminação de Região
    - Métodos por Aproximação Polinomial - Interpolação Quadrática ou Interpolação Cúbica

- Otimização Multidimensional Sem Restrições

1. Métodos Indiretos
    - Método do Gradiente ou Método do Gradiente Descendente
    - Método do Gradiente Conjugado
    - Método de Newton
    - Método de Levenberg-Marquardt
    - Método da Secante ou Quasi-Newton
2. Métodos Diretos
    - Busca Aleatória
    - Grade de Busca
    - Busca Unidimensional
    - Método Simplex ou do Poliedro Flexível

3.5 Método de otimização Newton-Raphson

O Método de Newton como sendo um método de segunda ordem, utiliza a informação da curvatura da função por meio de sua aproximação quadrática. Como é o método utilizado neste trabalho, aqui será dada então, uma visão geral do método.

É sabido que a primeira condição necessária para que um mínimo local ocorra é que $f'(x)$ = 0. O que implica que, com a resolução da equação $f'(x)$ = 0, pode-se obter, pelo Método de Newton,

$$x^{k+1} = x^k - \frac{f'(x^k)}{f"(x^k)} \tag{3.2}$$

garantindo que em cada instante de tempo k, $f(x^{k+1}) < f(x^k)$ para um mínimo. Uma ilustração pode ser vista na Figura 3.10

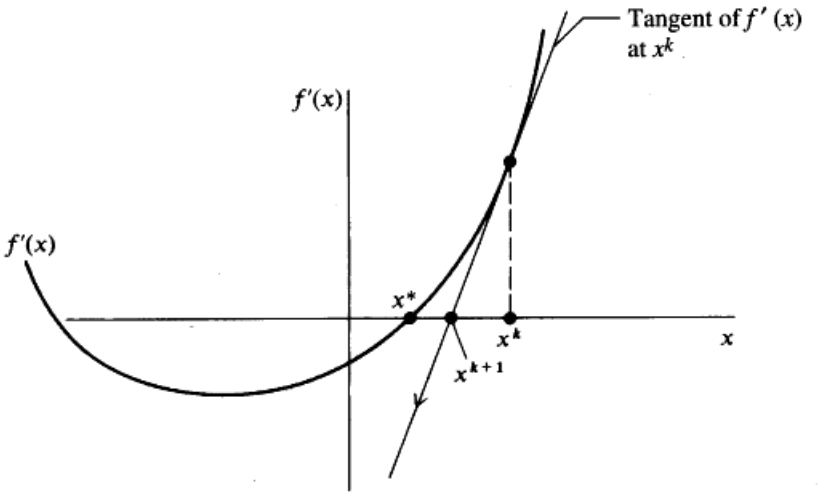


Figura 3.10: O método de Newton aplicado à solução de $f'(x) = 0$. Fonte: Edgar e Himmelblau (2001).

Como dito anteriormente, o método de Newton-Raphson é um método de segunda ordem, por isso uma aproximação de segunda ordem em torno do ponto de operação é necessária. Supondo $f(x)$ ser uma aproximação quadrática em x^k

$$f(x) = f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}f"(x^k)(x - x^k)^2 \tag{3.3}$$

A fim de encontrar $df(x)/dx = 0$, diferencia-se então a Equação 3.3 com respeito a x (Edgar e Himmelblau, 2001):

$$\frac{df(x)}{dx} = f'(x^k) + f"(x^k)(x - x^k) = 0 \tag{3.4}$$

Separando os termos pode-se chegar à Equação 3.2. Por análise, o Método de Newton equivale a usar um modelo quadrático (aproximação quadrática) para a função de otimização, que pode ser usada como um processo de minimização ou maximização.

O Método de Newton possui algumas vantagens. O procedimento de otimização converge quadraticamente para um extremo se e somente se $f"(x) \neq 0$, e para uma função quadrática, o mínimo é obtido em uma iteração. Como desvantagens tem-se o esforço computacional para

se calcular ambos $f'(x)$ e $f"(x)$; a convergência do método se torna lenta para $f"(x) \rightarrow 0$; há uma necessidade que o ponto inicial esteja próximo o suficiente do mínimo. Para algumas modificações no Método de Newton a fim de garantir convergência em pontos mais distantes do mínimo recomenda-se a leitura de Luenberger (2008).

Se não se pode encontrar as derivadas da função por forma analítica, pode-se recorrer às aproximações das derivadas. Uma aproximação por diferenças finitas pode ser feita para $f'(x)$ e $f"(x)$:

$$f'(x) \approx \frac{[f(x+h) - f(x-h)]}{2h} \tag{3.5}$$

$$f"(x) \approx \frac{[f(x+h) - 2f(x) + f(x-h)]}{h^2} \tag{3.6}$$

Deixando a regra de obtenção do x^* como:

$$x^{k+1} = x^k - \frac{[f(x+h) - f(x-h)]/2h}{[f(x+h) - 2f(x) + f(x-h)]/h^2} \tag{3.7}$$

Na Equação 3.7 utilizou-se diferenças centrais. A principal desvantagem em utilizar diferenças finitas é a introdução de erro.

3.6 Comentários finais

Este capítulo abordou os fundamentos relacionados ao controle baseado em modelo e às formas mais conhecidas de utilização das RNAs em sistemas de controle. Foram abordadas estratégias de controle que fazem uso das RNAs como modelo do processo.

Na Seção 3.2 foi abordado o uso de RNAs em sistemas de controle. A forma de atuação dos neuro-controladores como controle direto e indireto. Foi mostrado a classificação das RNAs quanto sua disposição dentro dos controladores, assumindo a função de controlador e modelo do processo. Apresentou-se as estratégias de controle MRC e IMC.

Na Seção 3.3 abordou-se o controle preditivo baseado em modelo, mais conhecido como a estratégia de controle MPC, definido como não uma estratégia específica mas sim uma classe contendo vários métodos. Apresentou-se as idéias e elementos dos algoritmos MPC, como sua estrutura e estratégia de controle. Mostrou-se a caracterização do algoritmo de controle GPC

que é a abordagem empregada neste trabalho.

A Seção 3.4 tratou da formulação da otimização apresentando alguns passos no projeto de otimização de processos e a classificação dos métodos de otimização. Por fim, na Seção 3.5, o método de otimização de Newton-Raphson foi abordado realizando uma visão geral deste método que é a ferramenta de otimização utilizada neste trabalho de pesquisa.

Capítulo 4

Planta piloto e neuro-controlador utilizado

4.1 Introdução

Os controladores preditivos baseados em modelos têm como característica o uso de ações de controle futuras calculadas com base em valores preditos na saída de um modelo do processo. Publicações recentes demonstram a aplicabilidade da estratégia MPC em problemas de controle (van Overloop et al., 2008; Zhang et al., 2008; Yu e Yu, 2007, 2005; Akesson e Toivonen, 2006).

Dentro da classe MPC de controladores, o controle preditivo generalizado (GPC), introduzido por Clarke et al. (1987), tornou-se um dos métodos de controle mais conhecidos. De acordo com Clarke et al. (1987), o GPC possui robustez em relação a erros de modelagem e foi originalmente desenvolvido com modelos lineares, e se um modelo não-linear for utilizado há a necessidade de um algoritmo de otimização não-linear, que é o caso do algoritmo NGPC.

No presente capítulo é apresentado o controle preditivo baseado em modelos neurais. Serão apresentadas as equações utilizadas no sistema de controle de uma planta piloto de neutralização de *pH*. A utilização de uma planta piloto possibilita a interação do controlador com as dinâmicas do sistema real, assim podendo ser analisado na prática e não apenas em sistemas simulados, já que o intuito principal dos controladores é serem colocados em operação no controle de sistemas reais.

4.2 Planta piloto de neutralização de *pH*

Entre várias aplicações que envolvem *pH*, o controle de neutralização de *pH* tende a ser usado como *benchmark* para testes de controladores. Alguns dos motivos desta escolha como *benchmark* é devida a fatores como: a alta não-linearidade do processo de neutralização; a sensibilidade em adição de reagentes que pode ser grande em determinado ponto e pequena em outros pontos; o ganho do processo que é variante com as diferentes faixas de *pH*; além de tudo isso, às várias perturbações externas que este processo está sujeito, como por exemplo, temperatura dos reagentes ou impurezas nos mesmos.

Assim, para interagir o neuro-controlador proposto com as dinâmicas presentes em sistemas reais e observar seu comportamento no controle de sistemas reais, foram realizados testes na planta piloto de neutralização de *pH*, instalada no laboratório de Modelagem Otimização e Controle de Processos - MOCP, do Centro Universitário do Leste de Minas Gerais - UnilesteMG, situado em Coronel Fabriciano (MG) - Brasil.

A planta piloto de neutralização de *pH* foi primeiramente construída para investigar as melhorias na estimação de parâmetros de modelos NARX (*Nonlinear AutoRegressive with eXogenous input*) racionais com a incorporação de informação auxiliar (Campos, 2007), mas também está sendo utilizada como plataforma para realização de testes de identificação de processos e controle.

4.2.1 Descrição da planta

A planta piloto usada para a neutralização do *pH* é continuamente alimentada por três soluções que são a solução ácida, a solução base e a solução tampão[1]. Estas alimentam um reator central onde se encontra a sonda de *pH* e as comunicações são realizadas via interface USB. Constitui-se basicamente dos elementos:

- três bombas peristálticas para dosagem de reagentes com sinal de controle de 4 a 20mA;
- um agitador magnético;
- um medidor de *pH* com precisão de 0,01pH com transmissor de 4 a 20mA;
- um tanque para reagente ácido (60l);

[1] Uma solução tampão é uma solução que atenua a variação do valor do *pH*. Normalmente utilizada em reações com uso de controladores, contribuindo portanto para a estabilidade destes.

- um tanque para solução base (60l);
- um tanque para solução tampão (60l);
- um reator (2,1l);
- um tanque coletor (140l);
- duas placas de aquisição de dados USB, modelo NI USB-6008, com oito canais de entradas analógicas com resolução de 12 (doze) bits; dois canais de saídas analógicas com resolução de 12 (doze) bits; doze canais de entradas ou saídas digitais configuráveis; e taxa de amostragem de 10kS/s;
- um microcomputador processador Intel® Core 2 Duo T5450 com 2,0 (dois) Gbytes de memória RAM, HD (Disco Rígido) de 120 (cento e vinte) Gbytes, placa de vídeo *on-board* e três portas USB.

O diagrama funcional da planta de neutralização de *pH* pode ser vista na Figura 4.1[2]. As correntes de fluxo de solução da Figura 4.1 são caracterizadas da seguinte forma:

- corrente de fluxo de ácido composta por solução ácida HCl, 60l, *pH* 2,7, alimentando o reator via bomba $Q1$;
- corrente de fluxo de solução tampão composta por solução com $NaHCO_3$, 10l, *pH* 9,5, alimentando o reator via bomba $Q2$;
- corrente de fluxo de base composta por solução base $NaOH$ + $NaHCO_3$, 60l, *pH* 11,7, alimentando o reator via bomba $Q3$.

A foto da planta piloto pode ser vista na Figura 4.2, onde pode ser vista a plataforma utilizada para a realização dos testes como a sua dimensão aproximada e sua estrutura física.

[2]Na figura, a solução ácida apresentada é uma solução de HNO_3, diferente da solução utilizada no presente trabalho, HCl (Ácido Clorídrico).

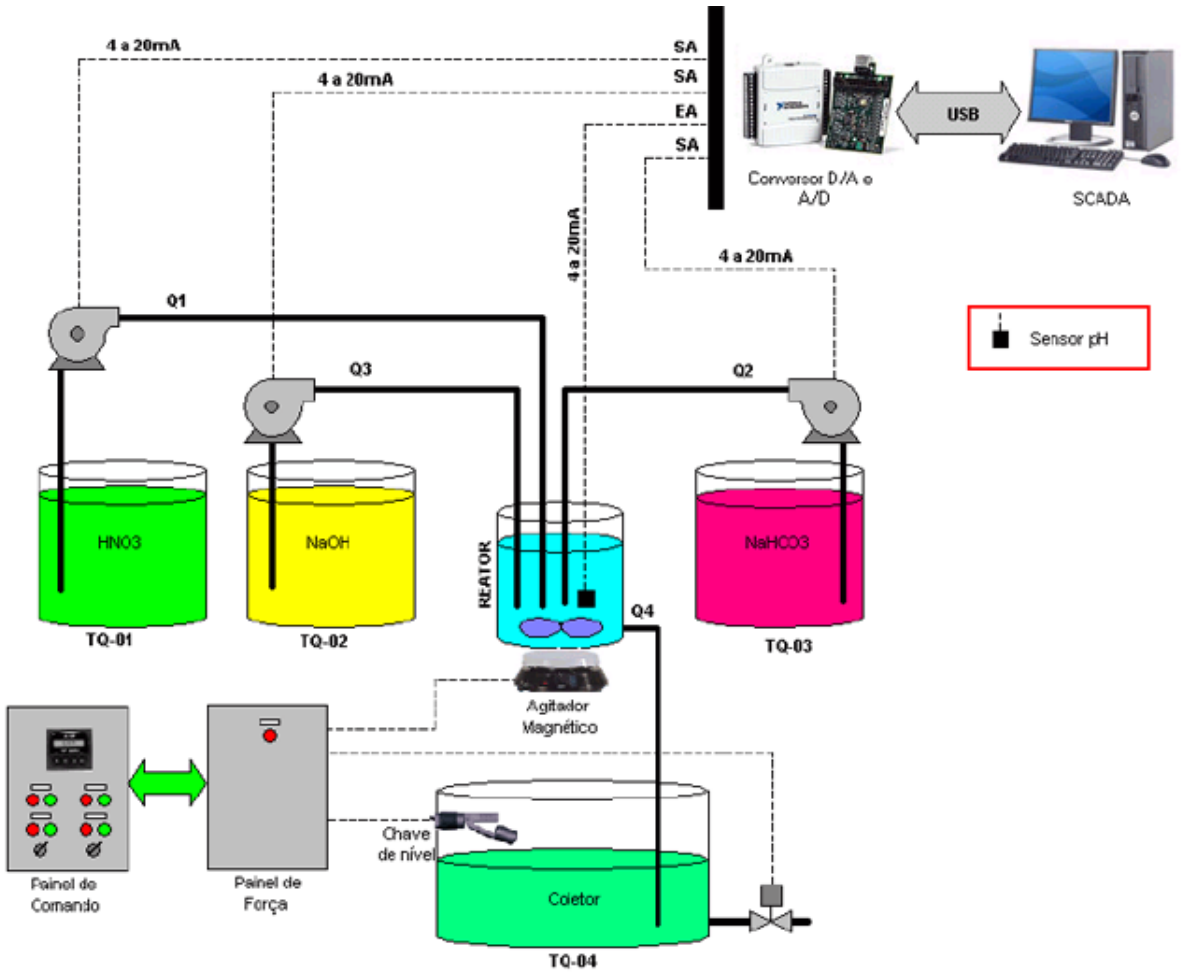


Figura 4.1: Diagrama funcional da planta piloto de neutralização de *pH*. Fonte: Campos (2007)

Figura 4.2: Foto frontal da planta piloto de neutralização de *pH*.

No preparo das soluções, os procedimentos adotados são os mesmos encontrados em Campos (2007). Um detalhamento maior sobre a planta piloto, sua estrutura, as especificações, os materiais e procedimentos podem ser encontrados em Campos (2007).

Na Figura 4.3 pode-se observar o *layout* do processo de neutralização de *pH*. O objetivo é realizar o controle do *pH* do processo na planta piloto manipulando as entradas $Q1$ e $Q3$ e mantendo-se constante a vazão volumétrica $Q2$ (solução tampão). Conforme a vazão desta solução tampão ($Q2$) ou de sua concentração, a curva estática do processo de neutralização de *pH* é alterada.

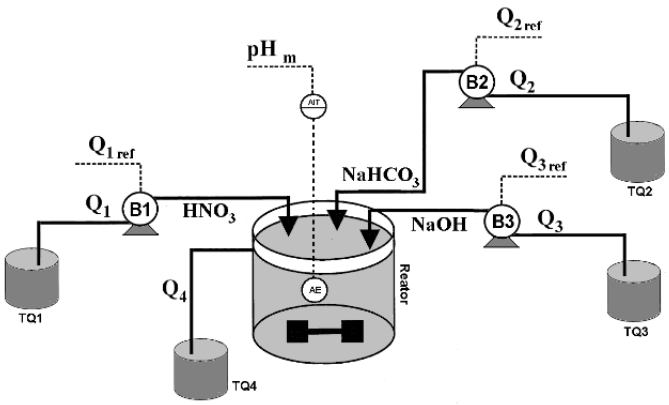


Figura 4.3: *Layout* do processo de neutralização de *pH*. Fonte: Campos (2007)

Como já mencionado anteriormente, o processo de neutralização do *pH* apresenta não-linearidades que, devido à intensidade destas, se torna um processo interessante para validar um controlador. Como forma de se observar estas não-linearidades, na Figura 4.4 pode ser vista a curva estática do *pH*, onde pode ser visto que, para o valor de *pH* menor que 4 o gradiente é menor do que o encontrado no *pH* de valor 8. Quanto ao processo de obtenção da curva, este pode ser encontrado em Campos (2007).

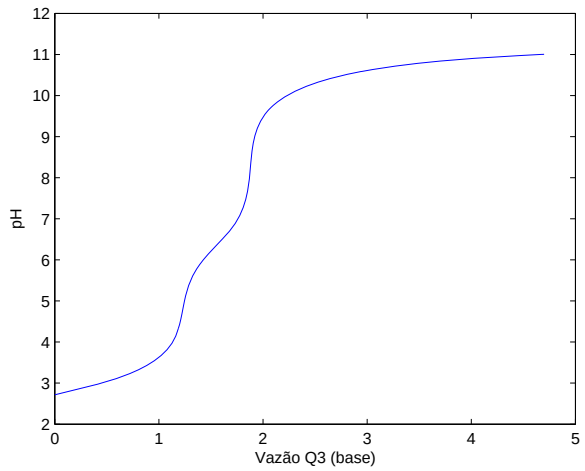


Figura 4.4: Curva de titulação de *pH* em função da vazão de base. Fonte: Campos (2007)

A Figura 4.5 mostra como o ganho do processo muda com o valor do *pH*. Há uma grande variação do ganho[3] nos diferentes valores de *pH*. Neste caso pode-se notar que o maior ganho estático encontra-se entre o *pH* 8 e 8,5, e o menor ganho nas regiões abaixo do *pH* 4 e acima do *pH* 10.

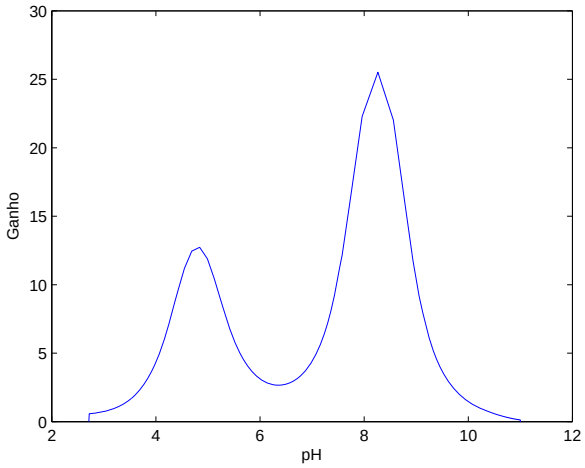


Figura 4.5: Ganho estático do processo versus *pH*. Variação do *pH* com pequenas variações na adição de base.

4.3 Modelo fenomenológico da planta piloto

O modelo outrora apresentado por Campos (2007), é um modelo fenomenológico da planta piloto de neutralização de *pH*, que foi utilizado para a realização de testes que serão apresentados no próximo capítulo juntamente com testes realizados na planta piloto de neutralização de *pH*. O modelo foi simulado com Runga-Kutta de quarta ordem e foi escolhido por sua proximidade ao processo real controlado neste trabalho de pesquisa. Este modelo foi utilizado como forma de verificar o desempenho e comportamento do neuro-controlador antes de ser aplicado ao processo real. Todas as equações do modelo fenomenológico podem ser encontradas em Campos (2007).

[3] A medida ou intensidade com que o *pH* varia com pequenas variações nas adições de ácido ou base.

4.4 Neuro-controlador

A arquitetura do neuro-controlador pode ser vista na Figura 4.6, e consiste de três elementos, a saber: a planta ou processo a ser controlado, a RNA que é o modelo do processo, e o algoritmo de minimização da função de custo (Otimizador), que determina a entrada necessária para levar o processo ao ponto desejado. O algoritmo NGPC consiste do bloco otimizador e do bloco neural.

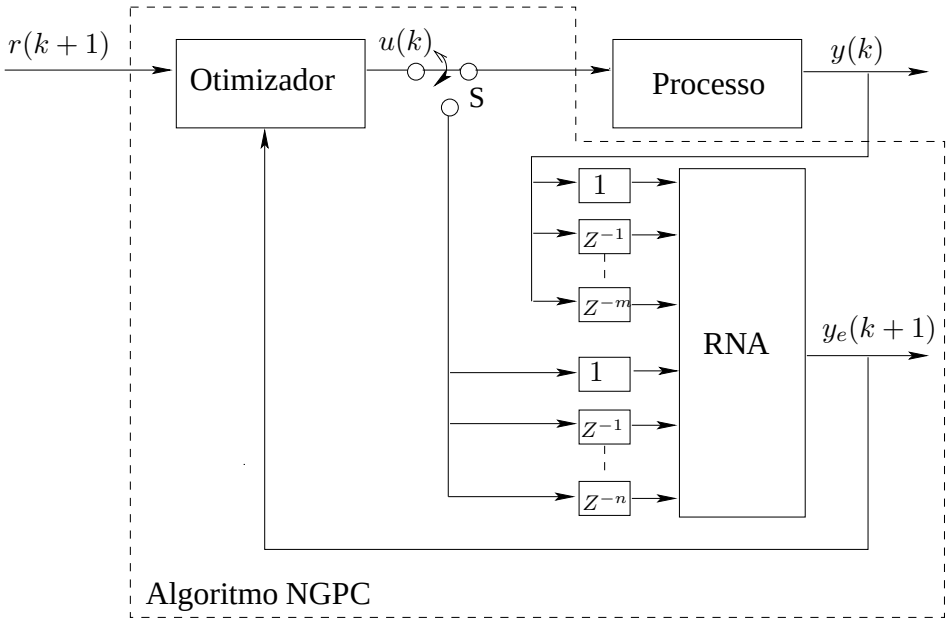


Figura 4.6: Arquitetura do neuro-controlador para uma variável.

Todo o processo de controle é resumido nos seguintes estágios:

- O algoritmo recebe a entrada $r(k+1)$ que é a referência para a qual se deseja levar o processo;
- Entre amostras, o seletor S é colocado para o modelo da planta, onde o otimizador usa a saída do modelo para calcular a ação de controle $u(k+1|k)$, que é encontrada utilizando as predições do modelo;
- O seletor S é colocado para o processo quando o otimizador encontra a melhor entrada $u(k)$ que minimiza a função de custo específica, e que portanto leva o processo para a referência desejada desde que o modelo represente de forma adequada as dinâmicas do processo.

O algoritmo de controle é implementado da seguinte forma:

1. Gere o sinal de referência;

2. Inicie com a ação de controle previamente calculada e faça a predição da planta usando o modelo, medindo o desempenho;

3. Calcule uma nova ação de controle que minimize a função de custo J;

4. Repita os passos 2 e 3 até uma minimização desejada seja atingida;

5. Aplique a ação de controle na planta;

6. Repita o processo novamente em cada instante k de amostragem.

4.4.1 Modelo Neural

Para a representação de sistemas dinâmicos é necessário a utilização de realimentação nas RNAs, gerando a recursividade. Fazendo menção aos modelos NARX (*Nonlinear AutoRegressive with eXogenous inputs*), associa-se aos sinais de entrada da rede os próprios valores de entrada e saída passados, caracterizando a recursividade. O modelo neural incorpora uma rede *perceptron* múltiplas camadas e emprega a sua capacidade de mapeamento não-linear.

Seja um modelo não-linear expresso na forma:

$$y(k+1) = \mathcal{F}[y(k), y(k-1),.., y(k-m), u(k), u(k-1),.., u(k-n)] \tag{4.1}$$

sendo $\mathcal{F}$ uma função não-linear dos valores passados da saída e da entrada do sistema.

Para o caso de uma variável a ser controlada a estrutura neural usada é ilustrada na Figura 4.7 abaixo,

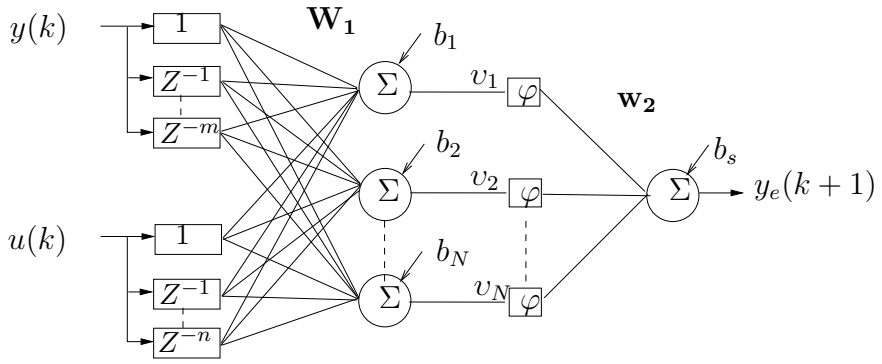


Figura 4.7: Arquitetura de uma rede recorrente genérica.

sendo o número de neurônios da camada intermediária representados por N, as polarizações de cada neurônio desta camada definidos por b_i, o bloco Σ representa o combinador linear e φ denota a função de ativação, sendo υ_i o campo local induzido. O b_s é a polarização da saída da rede. Matrizes $\mathbf{W_1}$ e $\mathbf{w_2}$ representam os pesos da camada oculta e saída respectivamente, $y(k)$ a saída do processo e $y_e(k+1)$ a saída estimada pela rede. O modelo neural utilizado neste trabalho é uma rede MLP e os detalhes da obtenção do modelo são vistos a seguir.

4.4.1.1 Obtenção do modelo neural no sistema simulado

Como o controle implementado é preditivo, há a necessidade da realização de predição através de um modelo do sistema. Optou-se pela utilização das RNAs na criação do modelo do processo, pelas características citadas no Capítulo 2.

De forma genérica, algumas etapas básicas devem ser seguidas para a criação de um modelo neural:

- Coleta de dados feita por meio da excitação do modelo do processo ou do processo real;
- Análise dos dados e escolha da arquitetura da RNA;
- Treinamento da RNA;
- Validação do modelo;
- Execução de simulações e testes.

Os índices de desempenho utilizados na escolha do modelo neural foram o *Mean Squared*

Error - MSE e o índice de correlação[4], além da utilização da Integral do Erro Absoluto - IAE (*Integral of Absolute Error*), definida como $IAE(k) = IAE(k-1) + |r(k) - y(k)|$ como avaliação de desempenho dentro do algoritmo NGPC. Assim, só será mostrada a melhor RNA obtida.

Os dados foram coletados do modelo simulado com Runga-Kutta de quarta ordem em malha aberta, usando sinais de excitação aleatórios tanto em duração de patamar como em amplitude, e podem ser vistos na Figura 4.8. Na Figura 4.8, o eixo abcissa corresponde ao tempo de excitação do sistema e o eixo de coordenadas a variação do *pH* na faixa 2,7 a 11,4. A linha tracejada corresponde aos sinais de excitação do fluxo de ácido e a linha pontilhada ao fluxo de base. A RNA obtida foi inserida no algoritmo NGPC e os ajustes necessários foram realizados para melhorar a resposta do neuro-controlador.

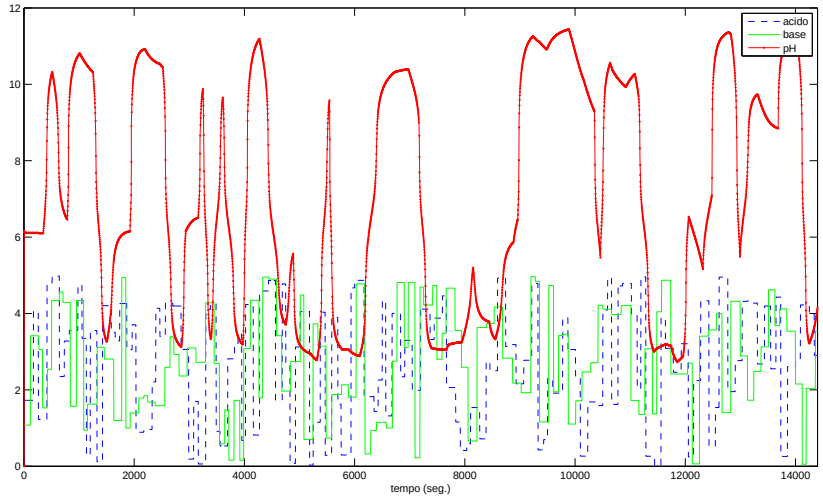


Figura 4.8: Excitação do modelo fenomenológico. Sinais de excitação das entradas referentes ao fluxo de ácido e de base e o valor de *pH* resultante.

Na Figura 4.9 pode ser vista a correlação cruzada dos sinais de excitação e as linhas centrais indicando o intervalo de confiança de 95%. Teve-se o cuidado de manter os sinais de excitação não correlacionados, pois em caso de sinais correlacionados o mapeamento entrada-saída seria prejudicado, não se tendo uma informação confiável de qual das entradas seria a causadora de alterações na saída do processo.

[4]Realiza uma regressão linear entre a resposta da rede e o valor real, calculando o coeficiente de correlação entre ambos. No MATLAB é a função conhecida como POSTREG.

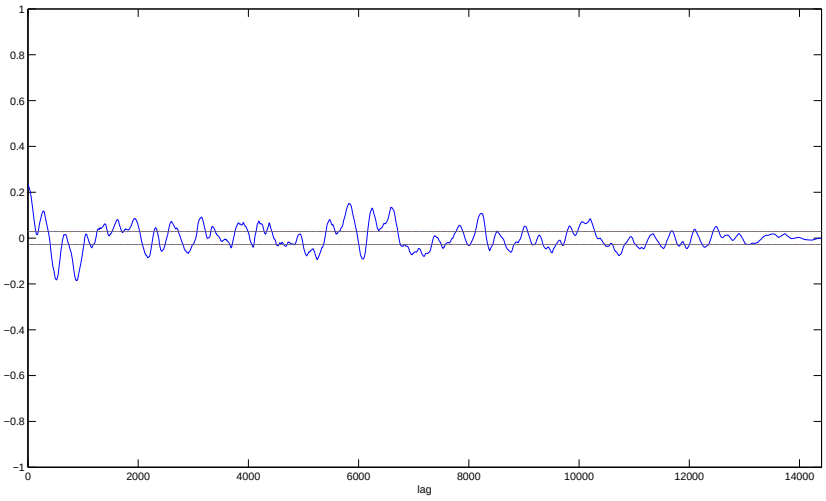


Figura 4.9: Gráfico da correlação cruzada entre os sinais de excitação $u_1(k)$ e $u_2(k)$.

A RNA escolhida para esta etapa de simulações possui as seguintes características:

- Uma camada oculta com 4 neurônios;
- Memória de atraso de ordem 2 para a saída $y(k)$;
- Memória de atraso de ordem 2 para a entrada $u_1(k)$ que se refere ao fluxo de ácido;
- Memória de atraso de ordem 2 para a entrada $u_2(k)$ que se refere ao fluxo de base;
- Funções de ativação sigmóide tangente hiperbólica e linear para a camada oculta e a camada de saída respectivamente.

A estrutura neural escolhida é portanto uma rede neural MLP com 6 entradas, uma camada oculta contendo 4 neurônios e uma camada de saída. No processo da escolha do modelo neural, que envolveu o treinamento de vários modelos neurais, os neurônios da camada oculta foram variados em número de 2 a 8 neurônios, tendo-se a intenção de se buscar uma estrutura com menor número de neurônios possível na camada oculta para facilidade de computação sem prejudicar a generalização. Resultados em termos da resposta em predição e o desempenho do neuro-controlador foram tomados como parâmetros para a escolha do modelo neural.

O algoritmo de treinamento utilizado nesta etapa foi o *Scaled conjugate gradient back-propagation* (Møller, 1993; Andrei, 2008), sendo este utilizado nesta etapa não por motivo específico, mas por ter encontrado o modelo neural que permitiu ao neuro-controlador alcançar resultados considerados satisfatórios já nos testes iniciais.

O modo de treinamento utilizado foi em lote ou batelada. Durante o treinamento foram utilizados 3360 (três mil e trezentos e sessenta) padrões para a etapa de treinamento e 1440 (mil quatrocentos e quarenta) padrões para a etapa de validação.

Nesta etapa, a rede neural não demonstrou predição livre satisfatória mesmo após a realização de vários treinamentos, mas possibilitou o alcance de resultados considerados satisfatórios pelo controlador dentro do algoritmo NGPC. A RNA foi treinada como preditora e na Figura 4.10 pode-se verificar a predição de um passo a frente da RNA aproximadora do modelo fenomenológico. O erro de previsão ($y(k+1)$ - $y_e(k+1)$) para a predição de um passo a frente pode ser visto na Figura 4.11, sendo o erro médio quadrático para a figura de MSE = 0,0068.

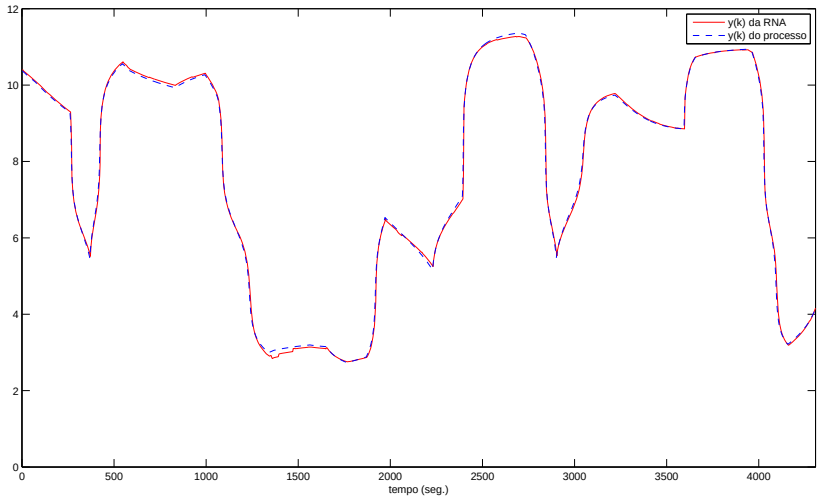


Figura 4.10: Predição de um passo à frente da RNA.

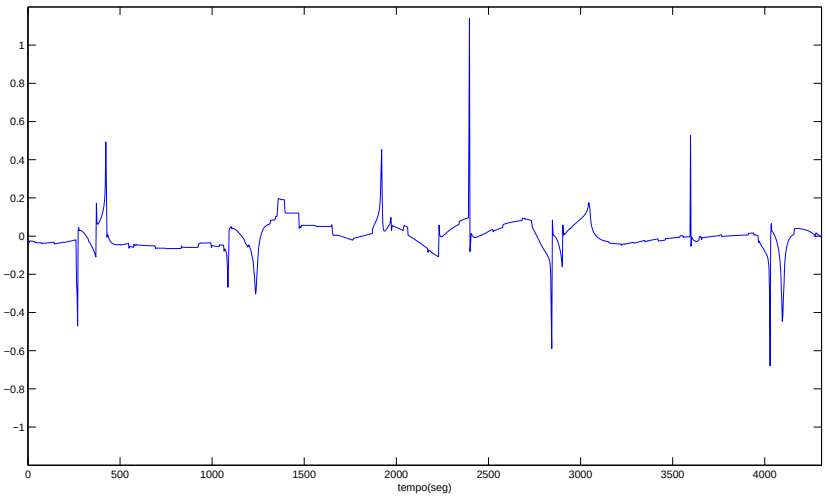


Figura 4.11: Erro de previsão da resposta em predição de um passo a frente.

4.4.1.2 Obtenção do modelo neural da planta piloto

Como mencionado anteriormente, um modelo do processo necessita ser encontrado para que o algoritmo NGPC encontre soluções para levar o processso real à referência desejada. Neste trabalho de pesquisa o modelo foi criado com o uso das RNAs, pela facilidade e pela capacidade das RNAs de captarem não-linearidades. O modelo do processo real foi resumidamente obtido pelas seguintes etapas:

- **Coleta de dados** - o processo real foi excitado com sinais aleatórios em amplitude e duração de patamar aplicados nas bombas $Q1$ e $Q3$, bombas de fluxo de solução ácido e base, respectivamente. A bomba $Q2$ foi mantida com vazão constante;

- **Análise de dados** - Vários treinamentos de RNAs foram feitos e observada a capacidade de generalização das mesmas e sua capacidade de predição mediante o uso do índice MSE e do índice de correlação, retromencionados;

- **Execução de testes** - A RNA foi inserida no algoritmo NGPC e foi realizado os ajustes dos parâmetros do neuro-controlador até se obter resultados satisfatórios. Quando não se obtinha resultados satisfatórios, treinava-se outro modelo neural do processo e realizava-se os reajustes dos parâmetros e a realização de novos testes.

A forma de obtenção da RNA foi pela coleta de dados via leitura da sonda de medição de *pH*, e dos valores aplicados às bombas de fluxo $Q1$ e $Q3$, mantendo-se constante a vazão da solução tampão ($NaHCO_3$) igual a $Q2$ = 0,1 ml/seg. A coleta se deu com o processo em malha aberta. As bombas de fluxo $Q1$ e $Q3$ tiveram suas vazões aleatoriamente variadas durante 3 horas entre valores de 0 ml/seg a 4,4 ml/seg, que corresponde a uma excitação de 0 a 5V. Utilizou-se os valores em Volts como entrada do modelo neural.

Para quantificação, a Equação 4.2 mostra uma relação entre a tensão e a vazão da bomba de fluxo de ácido $Q1$, enquanto a Equação 4.3 mostra a relação tensão e vazão da bomba de fluxo de base $Q3$.

$$Q1 = -0{,}0031V_1^2 + 0{,}85V_1 + 0{,}32 \tag{4.2}$$

$$Q3 = -0{,}029V_3^2 + 0{,}99V_3 + 0{,}24 \tag{4.3}$$

Sendo $Q1$ a vazão em ml/seg da bomba de fluxo de ácido e $Q3$ a vazão em ml/seg da bomba de fluxo de base. V_1 e V_3 são a tensão em Volts da bomba de fluxo de ácido e base respectivamente. Estas relações foram obtidas atráves de dados coletados por Campos (2007).

A taxa de amostragem de dados foi de 15 (quinze) segundos, dentro da faixa obtida por trabalhos anteriormente realizados no laboratório de Modelagem Otimização e Controle de Processos - MOCP, sendo próximo do valor obtido por procedimentos encontrados em Aguirre (2004). Os dados amostrados podem ser vistos na Figura 4.12.

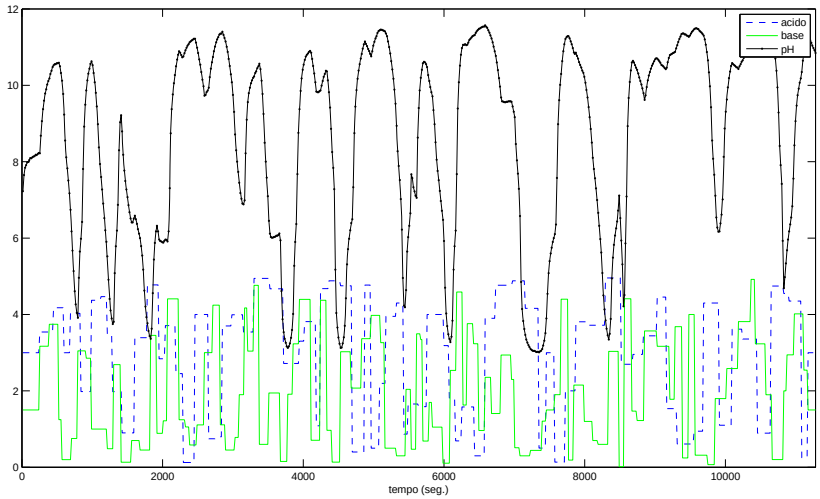


Figura 4.12: Excitação da planta piloto de neutralização de *pH*. Sinais de excitação da bomba de fluxo de ácido e de base e o valor de *pH* resultante.

A etapa de treinamento e validação da RNA foi realizada com 526 (quinhentos e vinte e seis) padrões de treinamento e 226 (duzentos e vinte e seis) padrões de validação. A validação com predição livre da RNA aproximadora do sistema real pode ser vista na Figura 4.13.

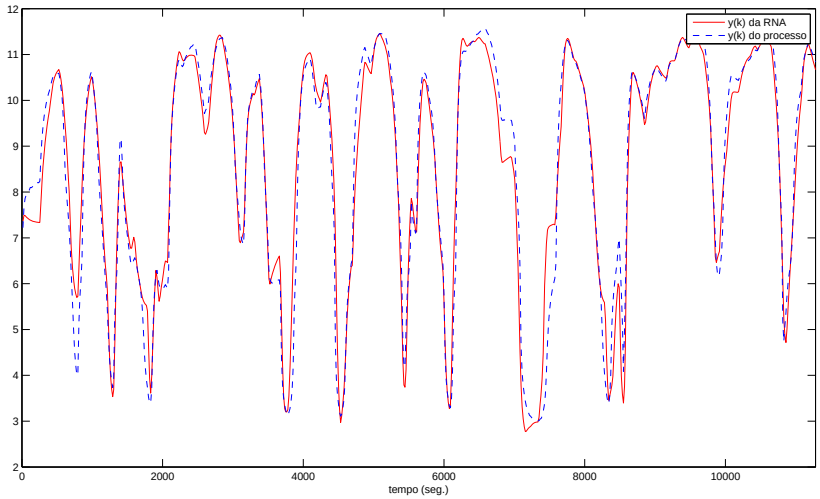


Figura 4.13: Teste de validação da RNA com predição livre.

A RNA utilizada é do tipo MLP com apenas uma camada oculta. Depois de várias arquiteturas avaliadas, com mudanças em número de neurônios da camada oculta e memórias de atrasos tanto para $y(k)$, $u_1(k)$ e $u_2(k)$, chegou-se a uma RNA com as seguintes características:

- Somente uma camada oculta e esta contendo 4 neurônios;
- Memória de atraso de ordem 2 para a saída $y(k)$;
- Memória de atraso de ordem 2 para a entrada $u_1(k)$;
- Memória de atraso de ordem 2 para a entrada $u_2(k)$;
- Funções de ativação do tipo sigmóide tangente hiperbólica e linear para a camada oculta e de saída, respectivamente;
- Treinamento em lote ou batelada;
- Algoritmo utilizado no treinamento: *Bayesian Regulation backpropagation* (Foresee e Hagan, 1997). Conhecido por melhorar a capacidade de generalização das redes, foi o algoritmo que gerou a rede neural que proporcionou melhores resultados em rastreabilidade pelo neuro-controlador nesta etapa.

4.4.2 Otimizador

4.4.2.1 Função de custo

Seja uma função de custo que se procura minimizar, representada por *J*. Uma função usual e bastante empregada é baseada no erro quadrático:

$$J = [y(k) - y_e(k)]^2 = [e(k)]^2 \tag{4.4}$$

sendo,

$y(k)$ a saída real do processo;

$y_e(k)$ a saída estimada pelo modelo;

$e(k)$ o erro estimado;

k o instante de tempo.

Se um controlador preditivo é utilizado, funções com base em apenas erro quadrático já podem garantir desempenhos satisfatórios. Se um horizonte de predição é utilizado, um vetor de erros futuros preditos são usados na minimização que ocorre ao longo de um horizonte de controle, a N_u passos à frente, assim reescrevendo a Equação 4.4.

$$J = \sum_{j=1}^{N_u} [y(k+j) - y_e(k+j)]^2 = \sum_{j=1}^{N_u} [e(k+j)]^2 \tag{4.5}$$

Funções de custo um pouco mais complexas podem ser implementadas. Necessitando-se de uma complexidade maior, a ponderação do esforço de controle pode ser utilizada na função de otimização. O controle preditivo generalizado (*Generalized Predictive Control* - GPC) usa esta abordagem (Clarke et al., 1987); (Chidrawar e Patre, 2008). A função de custo J utilizada neste trabalho é a função utilizada no algoritmo GPC, e esta pode ser expressa como:

$$J = \sum_{j=N_1}^{N_y} [r(k+j) - y_e(k+j)]^2 + \sum_{j=1}^{N_u} \lambda(j)[\Delta u(k+j)]^2 \tag{4.6}$$

sendo,

$r(k)$ a referência a ser seguida;

Δu a atualização no valor da ação de controle [$u(k+j)$ - $u(k+j-1)$];

λ a ponderação à ação de controle que limita o bloco controlador;

N_1 o horizonte mínimo de predição;

N_u o horizonte de controle;

N_y o horizonte máximo de predição.

4.4.2.2 Método de otimização utilizado

Esta seção será dividida em duas partes. A primeira parte diz respeito à derivação das equações referentes ao controle de um processo com uma variável apenas, parte utilizada com intuito didático como introdução à segunda parte, que são as equações referentes ao algoritmo de controle multivariável utilizado neste trabalho.

- **Controle monovariável**

Considera-se o modelo recorrente de entrada-saída apresentado na Figura 4.7. Para este modelo, generaliza-se a expressão, a qual representa a saída estimada $y_e(k+1)$ do modelo neural do processo:

$$y_e(k+1) = b_s + \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi(\upsilon_i) \tag{4.7}$$

sendo,

$$\upsilon_i = b(i,1) + \sum_{j=1}^{m} \mathbf{W_1}(i,j)y(k-j+1) + \sum_{j=1}^{n} \mathbf{W_1}(i,m+j)u(k-j+1) \tag{4.8}$$

Como apresentando no Capítulo 3, o método de Newton-Raphson utiliza informações de derivadas de ordem primeira e segunda. Para a aplicação da lei de controle envolvendo o modelo neural e a utilização do método de otimização de Newton-Rapshon, neste trabalho utilizou-se as expressões de derivada primeira e derivada segunda, especificadas a seguir.

- **Equações da derivada primeira**

Para a aplicação de uma lei de controle envolvendo um modelo neural do processo, necessita-se criar uma expressão que relaciona a saída da RNA com a entrada do processo, já que a entrada do processo será a variável a ser manipulada a fim de levar o processo real a uma referência desejada. Deste modo se pode aproveitar da capacidade da RNA em realizar predições e assim calcular ações de controle futuras que minimizem determinada função de custo.

Baseando-se na Equação 4.7, realizando a derivada parcial em relação à entrada $u(k)$, para o modelo neural, tem-se:

$$\frac{\partial y_e(k+1)}{\partial u(k)} = \frac{\partial}{\partial u(k)}\left[bs + \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi(\upsilon_i)\right] \tag{4.9}$$

Utilizando o conceito da regra da cadeia pode-se escrever:

$$\frac{\partial y_e(k+1)}{\partial u(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(\upsilon_i)\frac{\partial \upsilon_i}{\partial u(k)} \tag{4.10}$$

Sendo φ' a derivada primeira da função de ativação utilizada. Com base na Equação 4.8, expande-se a expressão $\frac{\partial \upsilon_i}{\partial u(k)}$:

$$\frac{\partial \upsilon_i}{\partial u(k)} = \frac{\partial}{\partial u(k)}\left[b(i,1)\right] + \frac{\partial}{\partial u(k)}\left[\sum_{j=1}^{m} \mathbf{W_1}(i,j)y(k-j+1)\right] + \frac{\partial}{\partial u(k)}\left[\sum_{j=1}^{n} \mathbf{W_1}(i,m+j)u(k-j+1)\right] \tag{4.11}$$

Verifica-se que os termos $y(k-1)$, $y(k-2)$,..., $y(k-m)$ e os termos $u(k-1)$, $u(k-2)$,..., $u(k-n)$ são valores passados e não dependem de $u(k)$, sendo o somatório apenas não nulo em j=1. Portanto:

$$\frac{\partial \upsilon_i}{\partial u(k)} = \mathbf{W_1}(i,m+1) \tag{4.12}$$

Realizando a substituição da Equação 4.12 na Equação 4.10, tem-se a generalização da expressão de $\frac{\partial y_e(k+1)}{\partial u(k)}$ para a RNA utilizada:

$$\frac{\partial y_e(k+1)}{\partial u(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(\upsilon_i)\mathbf{W_1}(i,m+1) \tag{4.13}$$

A equação 4.13 relaciona a saída da rede MLP com a entrada $u(k)$ que é a variável manipulada. Como mencionado anteriormente, há também a necessidade da geração de uma expressão da derivada segunda, que se segue.

- **Equações da derivada segunda**

A derivada segunda da saída do modelo neural em relação a entrada $u(k)$ pode ser encontrada derivando-se a Equação 4.10, o que resulta em:

$$\frac{\partial^2 y_e(k+1)}{\partial u^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\left[\varphi'(\upsilon_i)\frac{\partial^2(\upsilon_i)}{\partial u^2(k)} + \varphi"(\upsilon_i)\frac{\partial(\upsilon_i)}{\partial u(k)}\frac{\partial(\upsilon_i)}{\partial u(k)}\right] \tag{4.14}$$

sendo $\frac{\partial^2(\upsilon_i)}{\partial u^2(k)}$ = 0. Assim a Equação 4.14 é reescrita na forma:

$$\frac{\partial^2 y_e(k+1)}{\partial u^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi"(\upsilon_i)\frac{\partial(\upsilon_i)}{\partial u(k)}\frac{\partial(\upsilon_i)}{\partial u(k)} \tag{4.15}$$

- **Minimização da função de custo**

O objetivo do algoritmo NGPC (*Neural Generalized Predictive Control*) é minimizar J com respeito a $u(k+1|k)$, sendo esta a ação de controle a ser aplicada ao processo. Para a minimização da função de custo será usada a abordagem $u(k+1|k)$ para indicar a procura pela ação de controle que levará a saída do processo à referência desejada, em um instante de tempo à frente. Esta forma de abordagem evita confusões quanto a etapa anterior de derivação da rede neural.

Os controladores preditivos utilizam expressões para modificação das ações de controle que são função da função de custo a ser minimizada. A regra mais conhecida é a do gradiente descendente (também conhecido na literatura como *stepest descent method*) que pode ser vista na Equação 4.16. A atualização é feita na direção contrária do gradiente da função, procurando o ponto de mínimo:

$$u(k+1) = u(k) - \lambda\frac{\partial J}{\partial u(k)} \tag{4.16}$$

sendo,

$u(k)$ o valor da ação de controle no instante k;

λ a ponderação à ação de controle;

$\frac{\partial J}{\partial u(k)}$ a derivada primeira da função de custo J com relação a ação de controle atual.

No surgimento de um horizonte de controle N_u, um vetor de ações de controle pode ser escrito na forma:

$$U(k) = \left| \begin{array}{c} u(k+1) \\ u(k+2) \\ \vdots \\ u(k+N_u) \end{array} \right| \tag{4.17}$$

Assim escrevendo na forma vetorial a Jacobiana:

$$\frac{\partial J}{\partial U(k)} = \left| \begin{array}{c} \frac{\partial J}{\partial u(k+1)} \\ \frac{\partial J}{\partial u(k+2)} \\ \vdots \\ \frac{\partial J}{\partial u(k+Nu)} \end{array} \right| \tag{4.18}$$

Na presente pesquisa a derivada segunda da função de custo foi também utilizada. A regra de Newton-Raphson foi utilizada e esta é expressa como:

$$u(k+1) = u(k) - \left(\frac{\partial^2 J}{\partial u^2(k)} \right)^{-1} \frac{\partial J}{\partial u(k)} \tag{4.19}$$

A Equação 4.19 mostra a regra de atualização utilizada e corresponde a uma *"pure version"* (Edgar e Himmelblau, 2001) do método de Newton-Rapshon que normalmente não converge se o ponto inicial não é perto o suficiente de um mínimo local.

A expressão da derivada segunda pode ser colocada na forma matricial (Pam e Don, 1997); (Chidrawar e Patre, 2008) e é chamada de matriz Hessiana, que será vista mais à frente neste capítulo, podendo assumir a seguinte forma para uma ação de controle (Pam e Don, 1997):

$$\frac{\partial^2 J}{\partial U^2(k)} = \left| \begin{array}{cccc} \frac{\partial^2 J}{\partial u(k+1)^2} & \frac{\partial^2 J}{\partial u(k+2)\partial u(k+1)} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu)\partial u(k+1)} \\ \frac{\partial^2 J}{\partial u(k+1)\partial u(k+2)} & \frac{\partial^2 J}{\partial u(k+2)^2} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu)\partial u(k+2)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u(k+1)\partial u(k+Nu)} & \frac{\partial^2 J}{\partial u(k+2)\partial u(k+Nu)} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu)^2} \end{array} \right| \tag{4.20}$$

Seja agora a função de custo do algoritmo NGPC definida como:

$$J = [r(k+1) - y_e(k+1|k)]^2 + \lambda[\Delta u(k+1|k)]^2 \tag{4.21}$$

sendo,

$$[r(k+1) - y_e(k+1|k)] = e(k+1|k) \tag{4.22}$$

$$\Delta u(k+1|k) = u(k+1|k) - u(k+1|k-1) \tag{4.23}$$

Para a regra de atualização de Newton-Raphson da Equação 4.19, necessita-se da primeira e segunda derivada da função de custo J. O termo $y_e(k+1|k)$ da Equação 4.21 é dependente da ação de controle que é aplicada ao processo e sua derivada é a Equação 4.13. Com esta informação a derivada primeira da função de custo J é:

$$\frac{\partial J}{\partial u(k+1|k)} = -2e(k+1|k)\frac{\partial y_e(k+1|k)}{\partial u(k+1|k)} + 2\lambda\Delta u(k+1|k) \tag{4.24}$$

A derivada segunda da função de custo é encontrada derivando-se a Equação 4.24 novamente em relação a $u(k+1|k)$. Usando a regra da cadeia, tem-se que:

$$\frac{\partial^2 J}{\partial u^2(k+1|k)} = -2e(k+1|k)\frac{\partial^2 y_e(k+1|k)}{\partial u(k+1|k)} + 2\frac{\partial y_e(k+1|k)}{u(k+1|k)}\frac{\partial y_e(k+1|k)}{u^2(k+1|k)} + 2\lambda \tag{4.25}$$

Reorganizando-se os termos:

$$\frac{\partial^2 J}{\partial u^2(k+1|k)} = 2\left[\frac{\partial y_e(k+1|k)}{u(k+1|k)}\frac{\partial y_e(k+1)}{u(k+1|k)} - \frac{\partial^2 y_e(k+1|k)}{\partial u^2(k+1|k)}e(k+1|k) + \lambda\right] \tag{4.26}$$

- **Controle multivariável**

Seja o modelo não-linear expresso da seguinte forma:

$$y(k+1) = \mathcal{F}[y(k), y(k-1),.., y(k-m), u_1(k), u_1(k-1),.., u_1(k-n), u_2(k), u_2(k-1),.., u_2(k-o)] \tag{4.27}$$

A função $\mathcal{F}$ é uma função não-linear dos valores passados da saída e da entrada do sistema. Para o caso de duas variáveis a serem controladas, a estrutura neural utilizada é ilustrada na

Figura 4.14 abaixo.

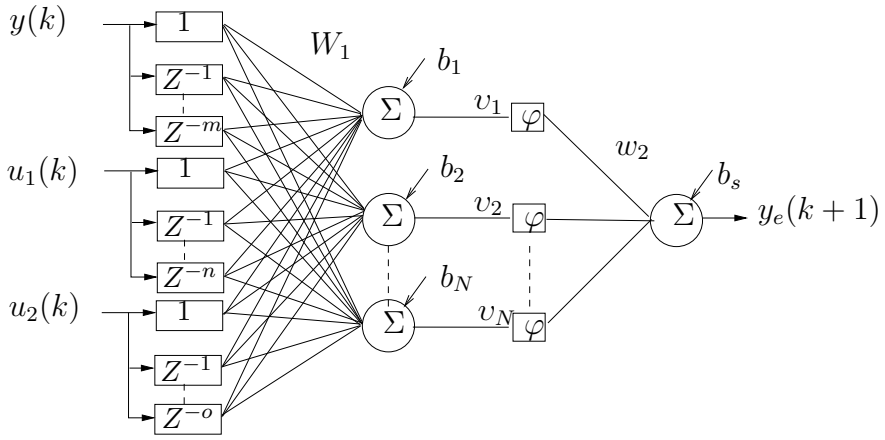


Figura 4.14: Arquitetura de uma rede recorrente genérica.

Nesta, o número de neurônios da camada intermediária é representado por N, as polarizações de cada neurônio desta camada denotado por b_i, o bloco Σ representa o combinador linear e υ_i o campo local induzido. O b_s a polarização da saída da rede neural. Matrizes $\mathbf{W_1}$ e $\mathbf{w_2}$ os pesos da camada intermediária e saída respectivamente. As entradas $u_1(k)$ e $u_2(k)$ são as entradas que correspondem às ações de controle que ora estão sendo modificadas para minimizar a função de custo J, ora estão sendo aplicadas ao processo real após otimização.

A expressão que generaliza a saída do modelo neural do processo é definida na Equação 4.7, sendo,

$$\upsilon_i = b(i,1) + \sum_{j=1}^{m} \mathbf{W_1}(i,j).y(k-j+1) + \sum_{j=1}^{n} \mathbf{W_1}(i,m+j)u_1(k-j+1) + \sum_{j=1}^{o} \mathbf{W_1}(i,m+n+j)u_2(k-j+1). \quad (4.28)$$

- **Equações da derivada primeira**

Agora com o surgimento de um termo adicional, há o surgimento de um vetor de derivadas primeira chamado de Jacobiana e a formação de uma matriz de derivadas segunda chamada Hessiana. Agora duas derivadas primeira precisam ser avaliadas já que o processo será controlado

com duas ações de controle. Esta derivação da saída da RNA em relação às entradas é crucial, já que é a informação necessária para que o algoritmo NGPC seja informado das modificações feitas nas ações de controle e seus respectivos efeitos no modelo neural. Estas modificações devem minimizar a função de custo J e assim quando aplicadas ao processo, devem garantir sua convergência para a referência desejada.

A derivada da saída em relação à entrada $u_1(k)$ é semelhantemente à Equação 4.13:

$$\frac{\partial y_e(k+1)}{\partial u_1(k)} = \frac{\partial}{\partial u_1(k)} \left[bs + \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi(\upsilon_i) \right] \tag{4.29}$$

Utilizando o conceito da regra da cadeia pode-se escrever:

$$\frac{\partial y_e(k+1)}{\partial u_1(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(\upsilon_i) \frac{\partial \upsilon_i}{\partial u_1(k)} \tag{4.30}$$

A derivada da Equação 4.28 em relação a $u_1(k)$ é:

$$\frac{\partial \upsilon_i}{\partial u_1(k)} = \mathbf{W_1}(i,m+1) \tag{4.31}$$

Realizando a substituição da Equação 4.31 na Equação 4.30, tem-se a derivada primeira da saída do modelo neural em relação à entrada específica $u_1(k)$:

$$\frac{\partial y_e(k+1)}{\partial u_1(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(\upsilon_i)\mathbf{W_1}(i,m+1) \tag{4.32}$$

Analogamente para $u_2(k)$:

$$\frac{\partial y_e(k+1)}{\partial u_2(k)} = \frac{\partial}{\partial u_2(k)} \left[bs + \sum_{i=1}^{N} w_2(1,i)\varphi(\upsilon_i) \right] \tag{4.33}$$

Utilizando o conceito da regra da cadeia pode-se escrever que:

$$\frac{\partial y_e(k+1)}{\partial u_2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(v_i)\frac{\partial v_i}{\partial u_2(k)} \tag{4.34}$$

A derivada da Equação 4.28 em relação a $u_2(k)$ é:

$$\frac{\partial v_i}{\partial u_2(k)} = \mathbf{W_1}(i,m+n+1) \tag{4.35}$$

Realizando a substituição na Equação 4.34, tem-se a derivada primeira da saída do modelo neural em relação à entrada específica $u_2(k)$:

$$\frac{\partial y_e(k+1)}{\partial u_2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi'(v_i)\mathbf{W_1}(i,m+n+1) \tag{4.36}$$

- **Equações da derivada segunda**

Como já mencionado o NGPC utiliza a atualização das ações de controle baseada no método de Newton-Raphson, tornando-se necessário encontrar a derivada segunda da saída em relação às entradas. Partindo da Equação 4.30, tem-se para a variável $u_1(k)$:

$$\frac{\partial^2 y_e(k+1)}{\partial u_1^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\left[\varphi'(v_i)\frac{\partial^2(v_i)}{\partial u_1^2(k)} + \varphi"(v_i)\frac{\partial(v_i)}{\partial u_1(k)}\frac{\partial(v_i)}{\partial u_1(k)}\right] \tag{4.37}$$

sendo $\frac{\partial^2(v_i)}{\partial u_1^2(k)} = 0$. Assim a Equação 4.37 é reescrita na forma:

$$\frac{\partial^2 y_e(k+1)}{\partial u_1^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi"(v_i)\frac{\partial(v_i)}{\partial u_1(k)}\frac{\partial(v_i)}{\partial u_1(k)} \tag{4.38}$$

De modo semelhante a derivada segunda da saída do modelo neural em relação à entrada $u_2(k)$ se torna:

$$\frac{\partial^2 y_e(k+1)}{\partial u_2^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\left[\varphi'(v_i)\frac{\partial^2(v_i)}{\partial u_2^2(k)} + \varphi"(v_i)\frac{\partial(v_i)}{\partial u_2(k)}\frac{\partial(v_i)}{\partial u_2(k)}\right] \tag{4.39}$$

sendo $\frac{\partial^2(v_i)}{\partial u_2^2(k)} = 0$. Assim a Equação 4.39 é reescrita na forma:

$$\frac{\partial^2 y_e(k+1)}{\partial u_2^2(k)} = \sum_{i=1}^{N} \mathbf{w_2}(1,i)\varphi"(v_i)\frac{\partial(v_i)}{\partial u_2(k)}\frac{\partial(v_i)}{\partial u_2(k)} \tag{4.40}$$

- **Minimização da função de custo**

Com duas ações de controle, a função de custo é reescrita na forma da Equação 4.21, tomando a forma:

$$J = [r(k+1) - y_e(k+1|k)]^2 + \lambda_1[\Delta u_1(k+1|k)]^2 + \lambda_2[\Delta u_2(k+1|k)]^2 \tag{4.41}$$

sendo,

$$[r(k+1) - y_e(k+1|k)] = e(k+1|k) \tag{4.42}$$

As derivadas são colocadas como elementos da Jacobiana. Deriva-se para isso a Equação 4.41 em relação às entradas $u_1(k)$ e $u_2(k)$, com a Jacobiana assumindo a forma:

$$\frac{\partial J}{\partial U(k)} = \left| \begin{array}{c} -2e(k+1|k)\frac{\partial y_e(k+1|k)}{\partial u_1(k+1|k)} + 2\lambda_1\Delta u_1(k+1|k) - 2\lambda_2\Delta u_2(k+1|k) \\ \\ -2e(k+1|k)\frac{\partial y_e(k+1|k)}{\partial u_2(k+1|k)} + 2\lambda_2\Delta u_2(k+1|k) \end{array} \right| \tag{4.43}$$

Observando-se o trabalho de Pam e Don (1997), que utiliza o algoritmo NGPC para o controle de uma planta não-linear com uma ação de controle, verificou-se a possibilidade de adequar o algoritmo para o controle do processo de neutralização do *pH* sob o âmbito de duas ações de controle. O aparecimento do termo de suavização relacionado à segunda ação de controle no primeiro termo da Equação 4.43 é herdado das equações de derivada segunda, relativo aos termos cruzados, que são a diagonal segunda da matriz Hessiana (Pam e Don, 1997). Todo o processo de derivação aqui apresentado é uma adequação do trabalho de Pam e Don (1997) para um processo com duas ações de controle.

Assim, as equações que formam a matriz Hessiana utilizada neste trabalho para o controle de um processo com duas ações de controle, são mostradas a seguir. Observando-se o método de otimização Newton-Raphson descrito em Edgar e Himmelblau (2001) e derivando-se as equações da RNA para duas entradas, denotadas como ações de controle, pode-se chegar às seguintes equações de derivadas segunda que formam a matriz Hessiana usada pelo algoritmo NGPC:

Termo (1,1):

$$\frac{\partial^2 J}{\partial u_1^2(k+1|k)} = 2\left[\frac{\partial y_e(k+1|k)}{\partial u_1(k+1|k)}\frac{\partial y_e(k+1|k)}{\partial u_1(k+1|k)} - \frac{\partial^2 y_e(k+1|k)}{\partial u_1^2(k+1|k)}e(k+1|k) + (\lambda_1 - \lambda_2)\right] \tag{4.44}$$

Termo (1,2):

$$\frac{\partial^2 J}{\partial u_1(k+1|k)\partial u_2(k+1|k)} = 2\left[\frac{\partial y_e(k+1|k)}{\partial u_1(k+1|k)}\frac{\partial y_e(k+1|k)}{\partial u_2(k+1|k)} - \frac{\partial^2 y_e(k+1|k)}{\partial u_1(k+1|k)\partial u_2(k+1|k)}e(k+1|k) - \lambda_2\right] \tag{4.45}$$

Termo (2,1):

$$\frac{\partial^2 J}{\partial u_2(k+1|k)\partial u_1(k+1|k)} = 2\left[\frac{\partial y_e(k+1|k)}{\partial u_2(k+1|k)}\frac{\partial y_e(k+1|k)}{\partial u_1(k+1|k)} - \frac{\partial^2 y_e(k+1|k)}{\partial u_2(k+1|k)\partial u_1(k+1|k)}e(k+1|k) - \lambda_2\right] \tag{4.46}$$

Termo (2,2):

$$\frac{\partial^2 J}{\partial u_2^2(k+1|k)} = 2\left[\frac{\partial y_e(k+1|k)}{\partial u_2(k+1|k)}\frac{\partial y_e(k+1|k)}{\partial u_2(k+1|k)} - \frac{\partial^2 y_e(k+1|k)}{\partial u_2^2(k+1|k)}e(k+1|k) + \lambda_2\right] \tag{4.47}$$

Formando a matriz Hessiana:

$$\frac{\partial^2 J}{\partial U^2(k)} = \begin{vmatrix} \frac{\partial^2 J}{\partial u_1^2(k+1|k)} & \frac{\partial^2 J}{\partial u_1(k+1|k)\partial u_2(k+1|k)} \\ \frac{\partial^2 J}{\partial u_2(k+1|k)\partial u_1(k+1|k)} & \frac{\partial^2 J}{\partial u_2^2(k+1|k)} \end{vmatrix} \tag{4.48}$$

A arquitetura do neuro-controlador para as duas ações de controle pode ser vista na Figura 4.15. O algoritmo de controle para duas variáveis é implementado da seguinte forma:

1. Gere o sinal de referência;
2. Inicie com as ações de controle previamente calculadas e faça a predição da planta usando o modelo neural, medindo o desempenho;
3. Calcule as novas ações de controle que minimizem a função de custo J;
4. Repita os passos 2 e 3 até que uma minimização desejada seja atingida;
5. Aplique as ações de controle na planta;
6. Repita o processo novamente em cada instante k de amostragem.

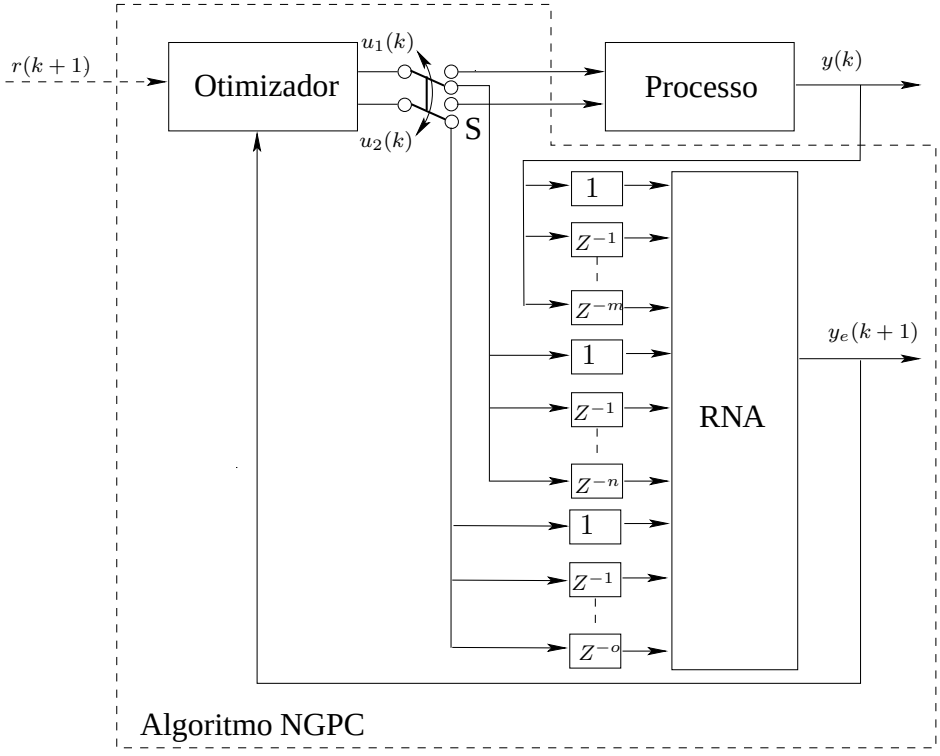


Figura 4.15: Arquitetura do neuro-controlador para duas ações de controle.

4.4.3 Parametrização do algoritmo NGPC

Antes de se passar para os resultados de testes experimentais no próximo capítulo torna-se necessário falar sobre os parâmetros do neuro-controlador. Tais parâmetros são de grande

importância pois estão diretamente relacionados com o desempenho do neuro-controlador. Os parâmetros presentes no neuro-controlador são:

- Os parâmetros λ_1 e λ_2 presentes na função de custo da Equação 4.41 ajustados empiricamente, que são a ponderação dada à variação das ações de controle. Estão diretamente relacionados à suavidade das ações de controle;
- Os parâmetros usados como condições de parada presentes no otimizador que definem a mudança mínima buscada das ações de controle entre iterações durante a minimização (m_u). Para melhor entendimento, é a condição de parada para que as ações de controle sejam aplicadas ao processo. Neste trabalho de pesquisa, o otimizador finaliza o processo de minimização quando a variação das ações de controle for menor que 10%. De forma genérica m_u é definido como:

$$m_u = \frac{\Delta u(k+1)}{u(k+1|k-1)} \tag{4.49}$$

  sendo,

$$\Delta u(k+1) = u(k+1|k) - u(k+1|k-1) \tag{4.50}$$

  Não se pode confundir a Equação 4.50 com a Equação 4.23. A Equação 4.50 diz respeito à ação de controle dentro do processo de minimização enquanto a Equação 4.23 diz respeito às ações já aplicadas ao processo real.

- Os parâmetros que definem os limites inferior e superior que as ações de controle podem atingir antes de serem aplicadas ao processo (restrições das ações de controle).

Um parâmetro inserido no neuro-controlador durante este trabalho de pesquisa que demonstrou melhores resultados no comportamento das ações de controle, foi um limitador de iterações do otimizador. Como verificado em Pam e Don (1997):

> "*O método de otimização de Newton-Raphson foi implementado, e para várias plantas, foi encontrado que ele converge para bons resultados com duas iterações.*"

Assim um limite de 3 (três) iterações foi permitido ao otimizador para o processo de minimização. Verificou-se que 3 foi o valor onde o algoritmo NGPC encontrava as ações de controle

necessárias com a melhor suavização das mesmas, sendo que para maior número de iterações não se observou melhoras significativas no comportamento das ações de controle, tornando assim estes cálculos dispensáveis. Este número pequeno de iterações é graças à rápida convergência do método de Newton-Raphson. Estes parâmetros quando ajustados corretamente, juntamente com o modelo satisfatório do processo, permitem o funcionamento estável do neuro-controlador. No próximo capítulo pode-se observar os testes realizados com a mudança de alguns destes parâmetros e os resultados obtidos.

4.5 Comentários finais

O capítulo iniciou abordando a planta piloto de neutralização de *pH* na Seção 4.2. Descreveu-se sucintamente o sistema simulado, dedicando a Seção 4.4 ao neuro-controlador utilizado, abordando também a obtenção do modelo neural e as equações utilizadas.

Na Seção 4.4.3 os parâmetros do neuro-controlador foram descritos. Estes são os parâmetros ajustados dentro do algoritmo NGPC até se obter um estado de trabalho satisfatório.

O neuro-controlador proposto neste capítulo utiliza o método de Newton-Raphson como ferramenta de otimização, sendo considerado um método de rápida convergência.

Capítulo 5

Estudo de casos, resultados e discussões

5.1 Introdução

Para a validação do neuro-controlador proposto, foram realizados testes que demonstram o comportamento do controlador aplicado no modelo simulado e no processo real apresentados no Capítulo 4. Os resultados serão vistos neste capítulo.

O capítulo está dividido em duas partes, sendo a primeira parte referente aos testes realizados no modelo fenomenológico da planta piloto, e uma outra parte referente aos testes realizados na planta piloto de neutralização de *pH*.

5.2 Modelo fenomenológico da planta piloto

Os seguintes resultados são referentes aos testes realizados no sistema simulado. De antemão, o valor dos parâmetros ajustados dentro do algoritmo NGPC e escolhidos como finais para esta etapa, por apresentarem resultados satisfatórios foram:

- λ_1 e λ_2, $8{,}0 \times 10^{-4}$ e $9{,}0 \times 10^{-4}$ respectivamente, encontrados empiricamente;
- m_u igual a 10% para ambas ações de controle;
- restrições das ações de controle em intervalo [1..3] para a ação de controle relativa ao fluxo de ácido e em intervalo [1..2] para a ação de controle relativa à base;
- número máximo de iterações do otimizador limitada em 3.

5.2.1 Simulações e resultados

Vários testes foram realizados com o neuro-controlador antes de sua aplicação à planta piloto de neutralização de *pH*. Os testes foram realizados a fim de verificar as características do neuro-controlador e sua capacidade em atender o objetivo de controlar a planta piloto mediante modificações nas duas ações de controle, que são o fluxo de ácido e o fluxo de base.

Nesta seção, as figuras estão divididas em três partes, representadas por letras. Estas letras mostram diferentes gráficos sendo:

- (a) referente ao sinal de referência $r(k)$ e a saída do processo $y(k)$;
- (b) referente ao valor das ações de controle aplicadas ao processo;
- (c) referente ao número de iterações usadas na minimização.

- **Teste de Rastreabilidade**

Na Figura 5.1 pode-se observar a saída do processo e a respectiva referência, como as ações de controle tomadas para levar o processo à referência desejada, e também o número de vezes que o algoritmo NGPC precisou computar até chegar às ações de controle escolhidas antes de serem aplicadas ao processo.

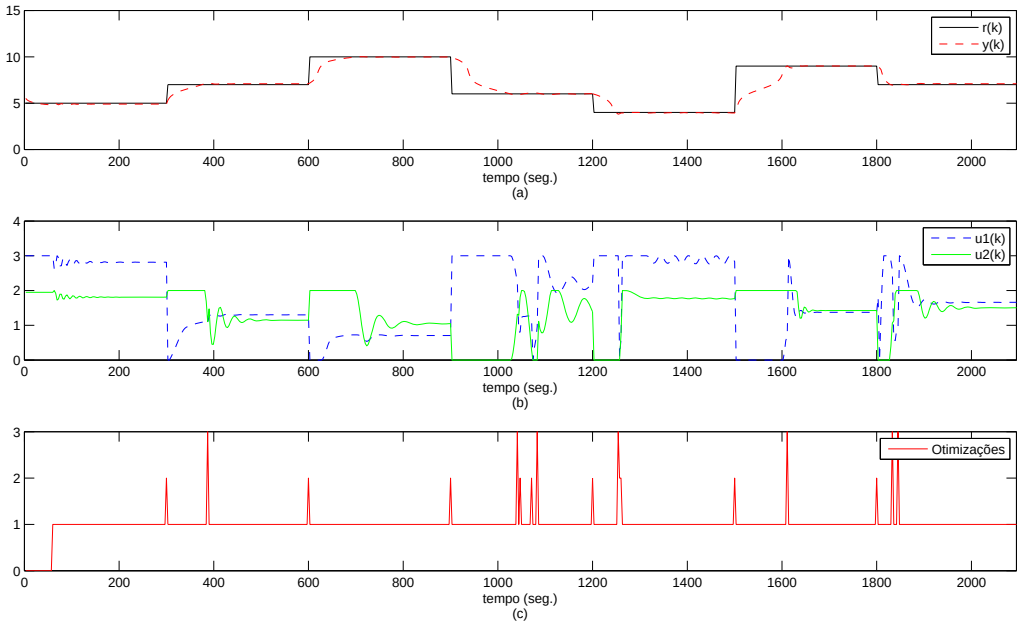


Figura 5.1: Modelo fenomenológico: teste de rastreabilidade. IAE = 255,35.

Note-se na Figura 5.1(a) que o neuro-controlador conseguiu seguir de forma satisfatória as trajetórias definidas para o teste de rastreabilidade. Os tempos de subida e de descida foram considerados satisfatórios, com mínimo sobre-sinal. Em (b), as mudanças nas ações de controle tomadas foram suaves, sendo uma característica desejada em sistemas reais mecânicos. A duração de tempo para se elevar o *pH* de 4 a 9 foi de 100 segundos, do tempo 1500 ao 1600. Em (c) é visto o número de operações realizadas pelo otimizador antes de aplicar as ações de controle ao modelo fenomenológico. Note-se que na maioria do tempo de simulação, apenas uma iteração foi necessária para que o otimizador encontrasse as melhores ações de controle. O índice IAE (Seção 4.4.1.1) será utilizado para efeito quantitativo das respostas em alguns gráficos neste capítulo. Quanto menor o valor de IAE, pode-se subentender que mais rápido o processo foi levado à referência e com menor número de oscilações em torno da mesma, sendo utilizado como medida de desempenho para a comparação de alguns resultados. Para a Figura 5.1 IAE = 255,35.

Para uma comparação da resposta obtida em rastreabilidade, uma maior folga foi permitida à variável que limita o número de otimizações realizadas pelo neuro-controlador. Na Figura 5.2, pode-se observar os resultados obtidos com um limite de 10 no número de otimizações permitidas.

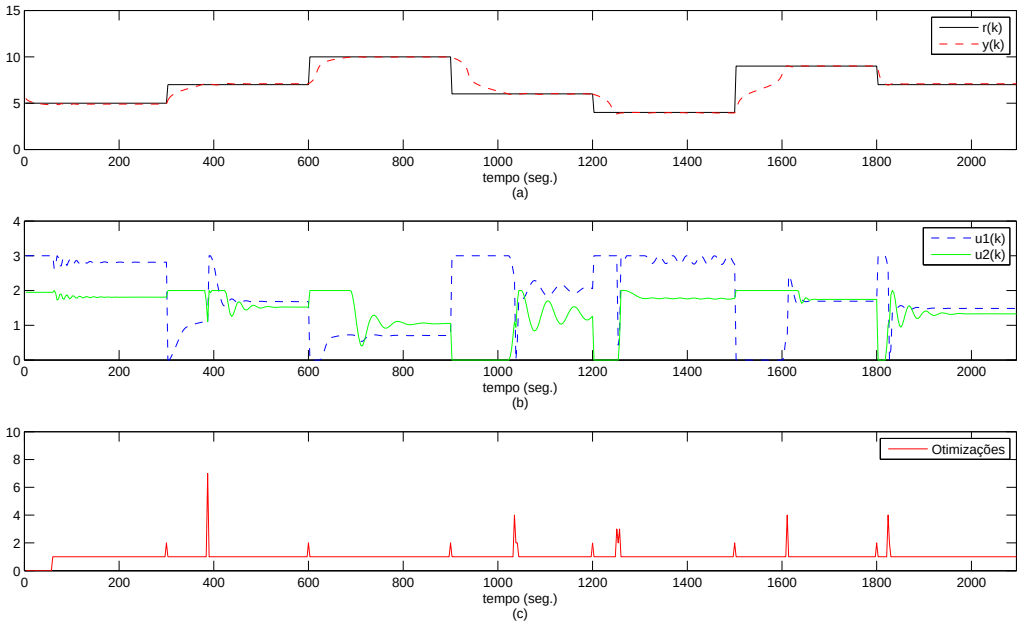


Figura 5.2: Modelo fenomenológico: teste de rastreabilidade com diferença na limitação da otimização. IAE = 242,55.

Note-se na Figura 5.2, próximo ao tempo 400 segundos, que o número de otimizações realizadas no processo de minimização foi 7. É um número maior que o obtido no teste da Figura 5.1 e apresentou uma oscilação maior na variável $u_1(k)$. O fato de se limitar o número de otimizações permitidas é que nos testes não se observou melhoras significativas em comportamento para permitir uma busca que iria implicar na necessidade de um tempo maior de otimização. O índice IAE nesta situação foi de IAE = 242,55, valor próximo ao encontrado na situação anterior.

- **Teste de parametrização λ_1 e λ_2**

Alguns ajustes foram realizados até se obter a melhor resposta do neuro-controlador. Os parâmetros λ_1 e λ_2 da Equação 4.41 foram ajustados. Como forma de mostrar o comportamento do neuro-controlador ao se modificar estes parâmetros, são apresentadas a seguir a resposta do algoritmo NGPC para mudanças dos mesmos durante ajuste do neuro-controlador. As Figuras 5.3, 5.4, 5.5 e 5.6 mostram os resultados quanto à rastreabilidade e comportamento das ações de controle.

Na Figura 5.3, para valores de λ_1 e λ_2 igual a $8{,}0 \times 10^{-3}$ e $9{,}0 \times 10^{-4}$ respectivamente, a rastreabilidade foi perdida entre o tempo 300 e 1200 segundos, onde as ações de controle respectivas não foram modificadas pelo algoritmo NGPC. Note-se que a partir do tempo 1200 segundos, a rastreabilidade é satisfatória, notando-se algumas oscilações na ação de controle $u_2(k)$ correspondente à aplicação de solução base. O índice IAE foi IAE = 970,55.

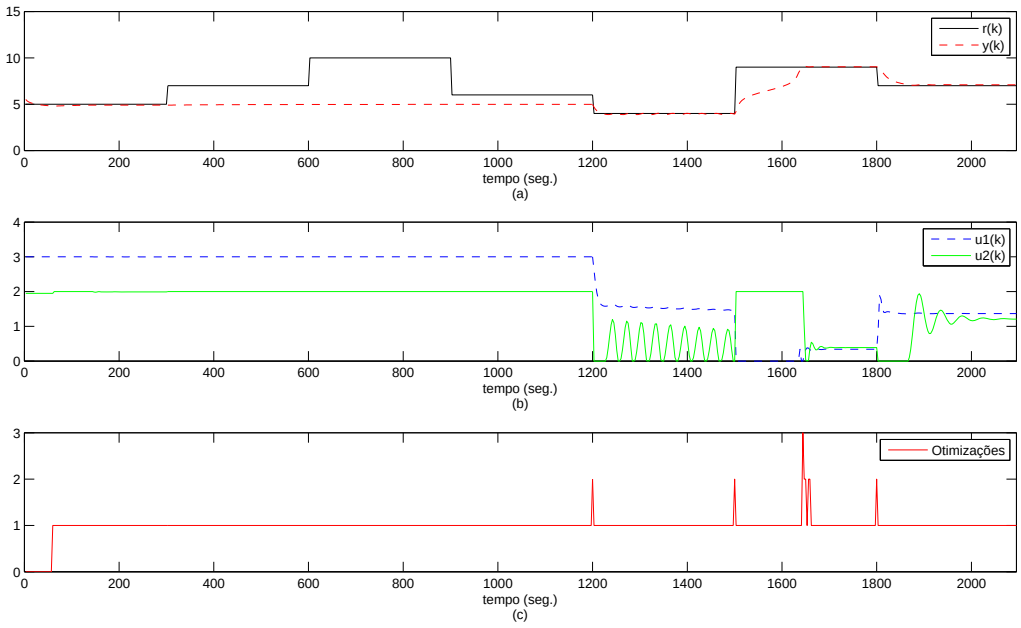


Figura 5.3: Modelo fenomenológico: teste de rastreabilidade para λ_1 e λ_2 igual a $8{,}0\times10^{-3}$ e $9{,}0\times10^{-4}$ respectivamente. IAE = 970,55.

Na Figura 5.4 os valores λ_1 e λ_2 foram $8{,}0 \times 10^{-5}$ e $9{,}0 \times 10^{-4}$ respectivamente. Com λ_1 100 vezes menor que a simulação anterior, a rastreabilidade ocorre de forma satisfatória, mas as ações de controle aplicadas entre a faixa de tempo 1200 e 1500 segundos (que compreende o patamar de *pH* 4) são consideravelmente oscilantes, mas fora desta faixa o resultado é bem satisfatório tanto na rastreabilidade como nas ações de controle aplicadas. Melhor desempenho em rastreabilidade é verificado pelo índice IAE = 276,26.

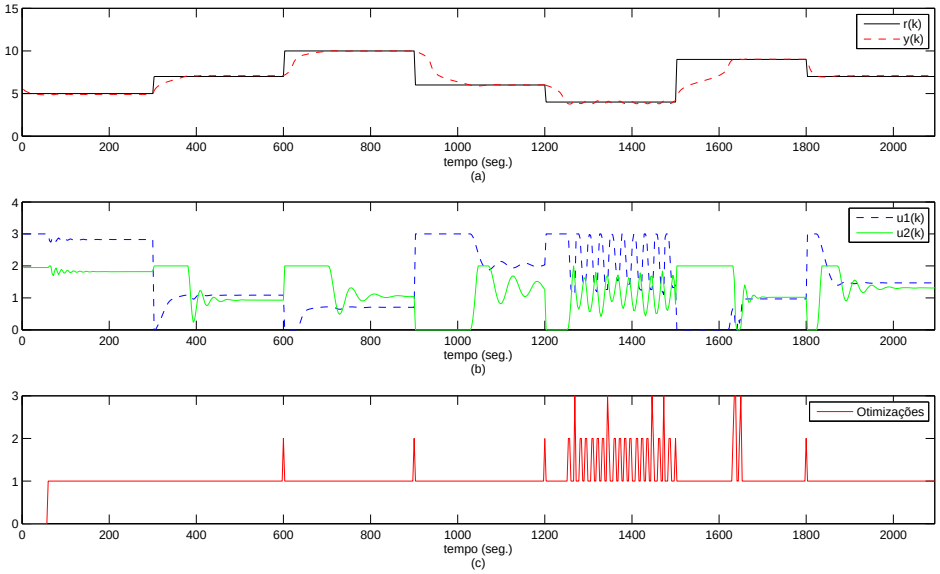


Figura 5.4: Modelo fenomenológico: teste de rastreabilidade para λ_1 e λ_2 igual a $8{,}0\times10^{-5}$ e $9{,}0\times10^{-4}$ respectivamente. IAE = 276,26.

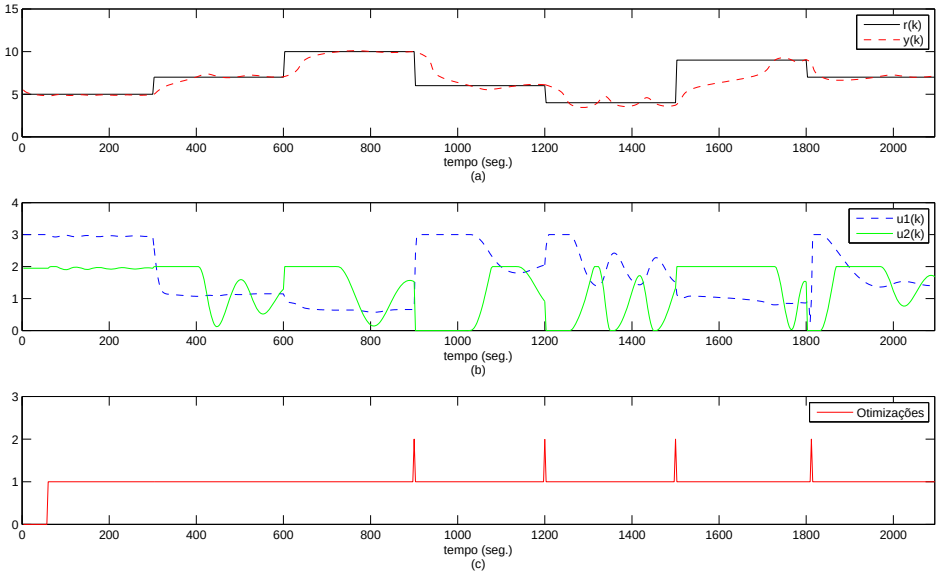


Figura 5.5: Modelo fenomenológico: teste de rastreabilidade para λ_1 e λ_2 igual a $8{,}0\times10^{-4}$ e $9{,}0\times10^{-3}$ respectivamente. IAE = 447,39.

Na Figura 5.5, os valores de λ_1 e λ_2 adotados foram $8{,}0 \times 10^{-4}$ e $9{,}0 \times 10^{-3}$ respectivamente. Se comparada à simulação anterior, o número de oscilações no patamar de *pH* 4 foram bruscamente reduzidas, mas em contrapartida houve um aumento na amplitude de sobre-sinal (*overshoot*) e maior tempo para estabilização em patamar, que podem ser verificados através do índice IAE = 447,39. Já o número de otimizações caiu consideravelmente.

Na Figura 5.6, o valor de λ_1 foi mantido igual a $8{,}0 \times 10^{-4}$ e λ_2 foi colocado em $9{,}0 \times 10^{-5}$. A rastreabilidade foi novamente perdida entre o tempo 300 e 1200 segundos e muita oscilação ocorreu a partir do tempo 1200 segundos na mudanças das ações de controle. Note-se que o número de cálculos realizados pelo otimizador aumentou consideravelmente dentro desta faixa de operação se comparada à Figura 5.3 e o índice IAE foi de IAE = 941,85.

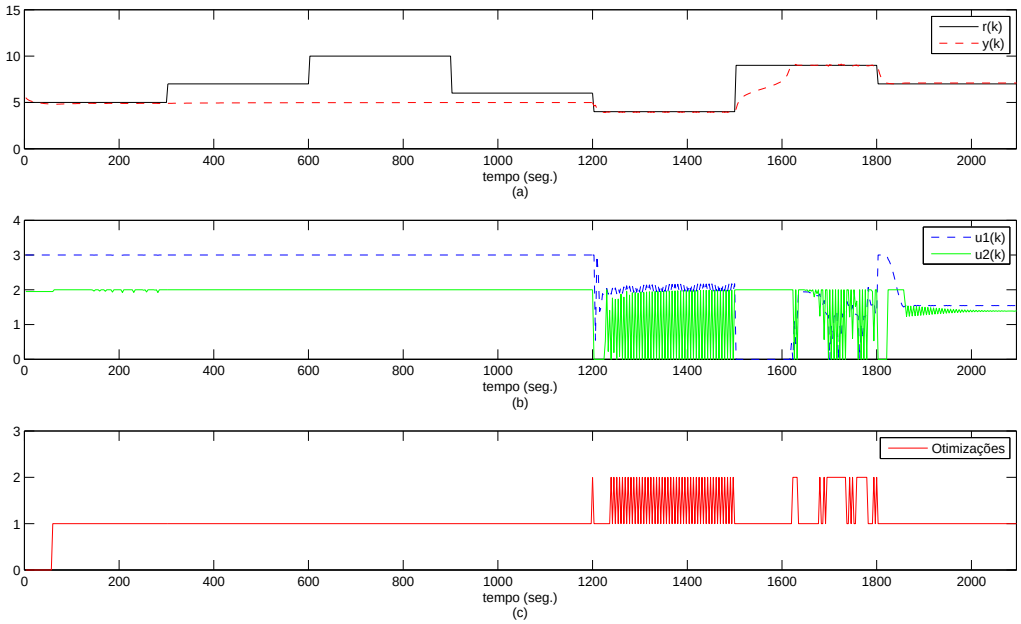


Figura 5.6: Modelo fenomenológico: teste de rastreabilidade para λ_1 e λ_2 igual a $8{,}0 \times 10^{-4}$ e $9{,}0 \times 10^{-5}$ respectivamente. IAE = 941,85.

Verificou-se durante os ajustes destes parâmetros:

- Que para melhores resultados λ_1 deve ser sempre mantido menor que λ_2. O valor de λ_1 menor do que λ_2 para este trabalho utilizando o modelo neural específico, proporcionou melhores resultados em controle;

- Um novo modelo neural ao ser inserido no algoritmo NGPC dificilmente retornou os mesmos resultados, necessitando-se então do reajuste dos parâmetros.

A etapa de excitação do processo para a obtenção do modelo neural é uma etapa delicada. Idealmente os sinais devem excitar igualmente as várias faixas de operação do processo. Isto na prática não é fácil em sistemas altamente não-lineares, quase sempre fazendo o modelo neural obtido ser tendencioso a alguma faixa de operação podendo, por exemplo, surgirem erros estáticos dentro de determinadas faixas durante o controle do processo.

- **Teste de travamento e variação da ação de controle**

Este teste visa verificar o comportamento do neuro-controlador em situações onde uma bomba de fluxo de ácido ou base sofre um travamento físico ou se estagna em determinado patamar e permanece inalterada, mesmo que o neuro-controlador envie informações para sua mudança. Desta forma se testa a capacidade do neuro-controlador em sentir esta situação e assim mesmo tentar levar o processo à referência desejada. Numa primeira parte será demonstrado o comportamento do neuro-controlador com travamento de fluxo de ácido. Numa segunda parte, será demonstrado o comportamento do neuro-controlador com travamento de fluxo de base. Será aproveitada esta última parte para verificar o comportamento do neuro-controlador nas situações:

1. Com os mesmos parâmetros utilizados no teste de suposto travamento da bomba de fluxo de ácido;
2. Com a modificação do limite superior que restringe a ação de controle referente à bomba de fluxo de ácido;
3. Com a modificação do limite superior que restringe a ação de controle referente à bomba de fluxo de ácido e a modificação do parâmetro m_u (Seção 4.4.3).

Na Figura 5.7 pode-se observar a simulação com travamento de fluxo de ácido no patamar 2 a partir do tempo 200 (duzentos) segundos. Os valores de λ_1 e λ_2 são $8{,}0 \times 10^{-4}$ e $9{,}0 \times 10^{-5}$ respectivamente. As ações de controle foram escolhidas pelo controlador com m_u igual a 10%. Os limites mínimo e máximo que restringem as ações de controle foram de 1 e 4 para a ação de controle relativa ao fluxo de ácido e de 1 e 3 para a ação relativa ao fluxo de base. Note-se que na ação de controle referente ao fluxo de ácido, os limites foram mantidos para cálculo interno do bloco otimizador e fixado no patamar 2 ao ser aplicado ao processo.

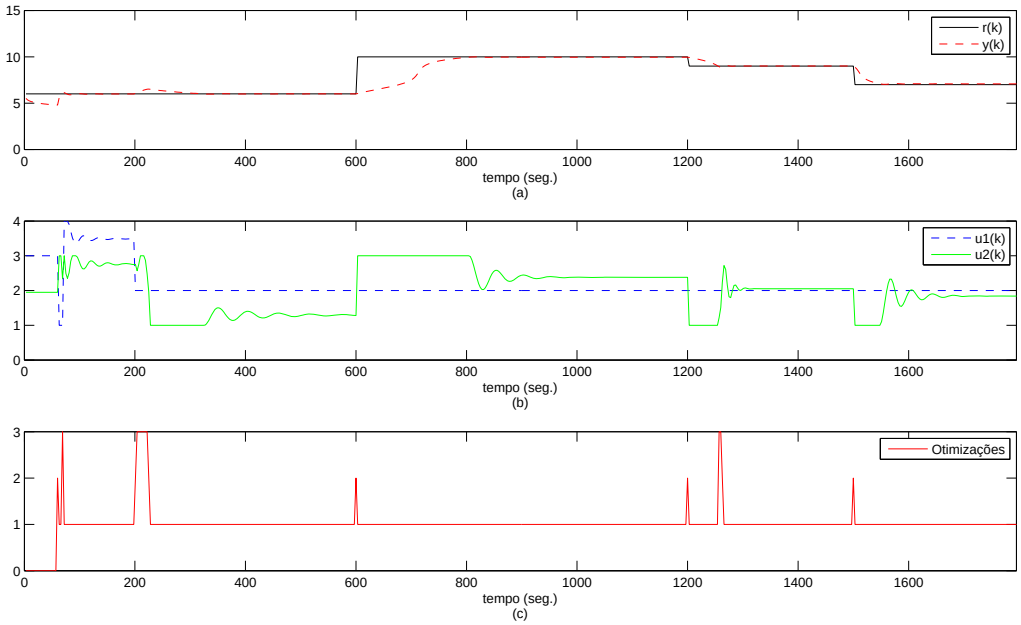


Figura 5.7: Modelo fenomenológico: Travamento da bomba de ácido.

Pode ser visto na Figura 5.7 que um travamento forçado da ação de controle $u_1(k)$ referente ao fluxo de ácido, não impediu que o neuro-controlador conseguisse fazer o sistema seguir a trajetória especificada. A ação de controle $u_2(k)$ referente ao fluxo de base se comportou com mudanças suaves. O neuro-controlador, mesmo calculando as duas ações de controle em todo o tempo percebeu que as mudanças efetuadas em $u_1(k)$ não surtiam efeito, e nestas circunstâncias fez o possível para manter o sistema fixo na trajetória estipulada, realizando as modificações necessárias em $u_2(k)$. A informação do número de iterações será mantida nas figuras seguintes do capítulo como informação adicional, vindo algumas vezes ser feito menção a estas em algumas figuras.

Com o travamento do fluxo de base no patamar 2 a partir do instante de tempo 200 segundos, pode ser visto o comportamento do neuro-controlador nas Figuras 5.8, 5.9 e 5.10. Na Figura 5.8 o teste foi realizado com restrição de 1 e 4 para a ação de controle relativa ao fluxo de ácido. O parâmetro m_u permanece no padrão utilizado neste trabalho, definido em 0,1 para as duas ações de controle. Na Figura 5.9, sob as mesmas condições, as restrições agora foram de 1 e 3 para a ação de controle relativa ao fluxo de ácido. O parâmetro m_u permanece definido em 0,1. Na Figura 5.10, a diferença foi a definição do parâmetro m_u igual a 0,005 para a ação de controle relativa ao fluxo de ácido.

Na Figuras 5.8, 5.9 e 5.10, o sistema seguiu todas as trajetórias de forma satisfatória, não havendo sobre-sinais de importância a fim de serem considerados. A diferença nestas figuras é a tentativa de se obter modificações o mais suaves possíveis na ação de controle referente ao fluxo de ácido sem comprometer a resposta do sistema no que se diz respeito a seguir a trajetória estipulada. Pode ser visto na Figura 5.9 que apenas uma modificação na restrição de limite superior da variável $u_1(k)$ (valor máximo que será aplicado ao sistema), restringe as oscilações em determinados patamares de operação, como nos patamares de *pH* 5 localizado entre os tempos 0 e 600 segundos, e no patamar de *pH* 7 localizado após o tempo de 1500 segundos.

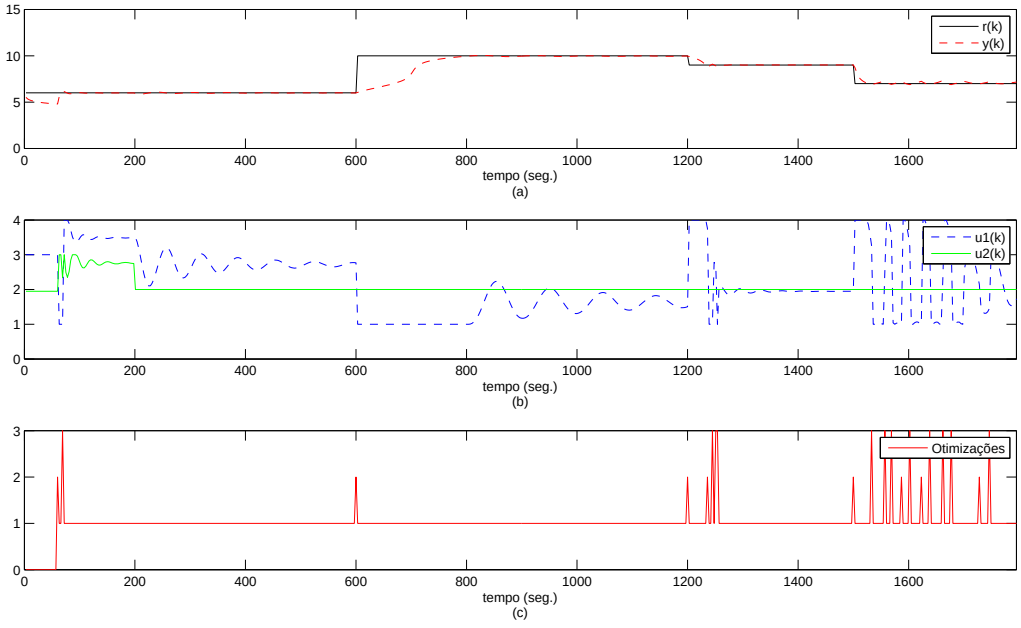


Figura 5.8: Modelo fenomenológico: Travamento da bomba de base, situação 1.

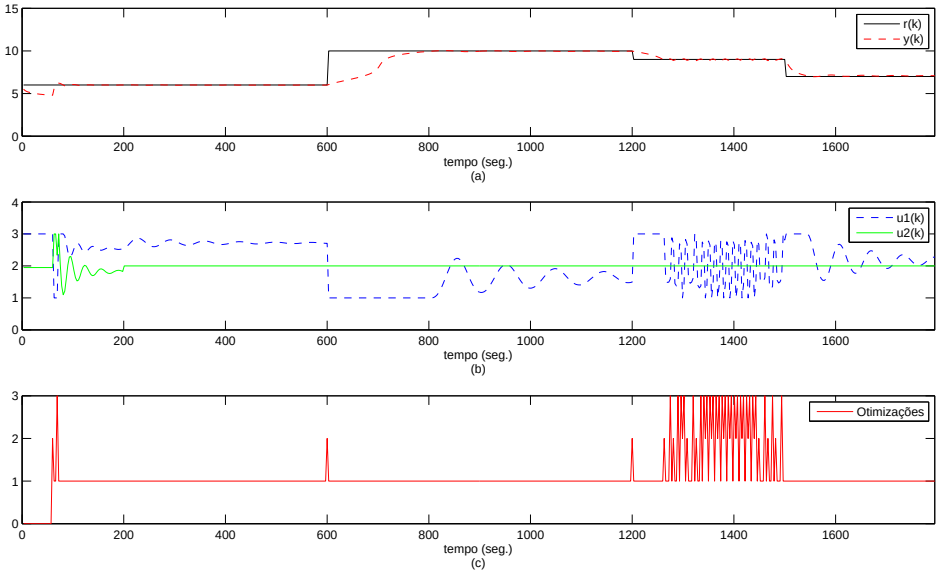


Figura 5.9: Modelo fenomenológico: Travamento da bomba de base, situação 2.

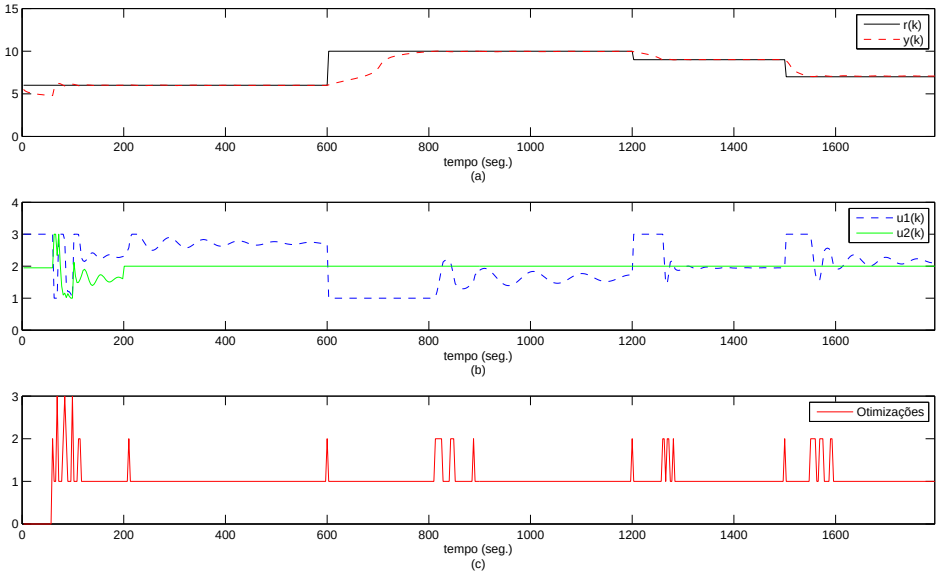


Figura 5.10: Modelo fenomenológico: Travamento da bomba de base, situação 3.

Note-se claramente a melhora significativa na Figura 5.10 do comportamento da ação de controle referente ao fluxo de ácido e o decaimento do número de otimizações realizadas. O processo ficou menos custoso computacionalmente e com maior suavização da ação de controle. A modificação realizada no parâmetro m_u da ação de controle referente ao fluxo de ácido, fixou que uma variação de 0,5% seria o critério para o otimizador parar o processo de minimização e aplicar esta ação de controle ao sistema. Com esta restrição maior em m_u verifica-se em comparação com as Figuras 5.9 e 5.8 a diminuição da oscilação em $u_1(k)$, principalmente nos patamares de *pH* situados entre 7 e 10. Estas observações também se aplicam quando o processo está em funcionamento livre (sem simulação de suposto travamento). Dentro de uma faixa de operação pode-se encontrar o valor dos parâmetros a fim de se ter a melhor resposta.

5.3 Planta piloto de neutralização de *pH*

Os seguintes resultados são referentes a resultados de testes realizados na planta piloto de neutralização de *pH*. De antemão, para as simulações realizadas na planta piloto de neutralização de *pH*, os valores dos parâmetros escolhidos como os que apresentaram resultados satisfatórios foram:

- λ_1 e λ_2, 0,01 e 0,06 respectivamente;
- m_u igual a 5% para ambas ações de controle;
- restrições das ações de controle em 1 e 4 para a ação de controle relativa ao fluxo de ácido e de 0.5 e 2 para a ação de controle relativa à base. As restrições inferiores garantem que sempre se manterá um fluxo mínimo de solução dentro do reator, o que é esperado e é o que basicamente acontece em processos industriais;
- número máximo de iterações do otimizador limitada em 3.

5.3.1 Testes e resultados

Nesta seção se encontra os resultados obtidos com os respectivos testes realizados na planta piloto de neutralização de *pH*. Contempla a aplicação do neuro-controlador proposto ao processo real.

- **Teste de rastreabilidade**

Para verificar o comportamento do neuro-controlador em várias faixas de operação foi realizado o teste de rastreabilidade no processo real. A Figura 5.11 mostra o resultado obtido. Foi passado ao sistema real a referência $r(k)$ para que o neuro-controlador em tempo real tomasse as ações necessárias para que o processo seguisse a referência desejada. A parte (a) da Figura 5.11 mostra o sinal de referência $r(k)$ e a saída do processo real. A parte (b) mostra as ações de controle calculadas e aplicadas ao processo real para levar a saída à referencia. A parte (c) mostra o número de iterações que o algoritmo NGPC utilizou para chegar à solução. Em (a), o eixo abcissa comporta o tempo de realização do teste em segundos enquanto o eixo de coordenadas indica o valor do *pH*; em (b) o eixo coordenada indica a tensão em Volts aplicada nas bombas peristálticas. Note novamente o número pequeno de iterações que o método de Newton-Raphson precisou utilizar em (c).

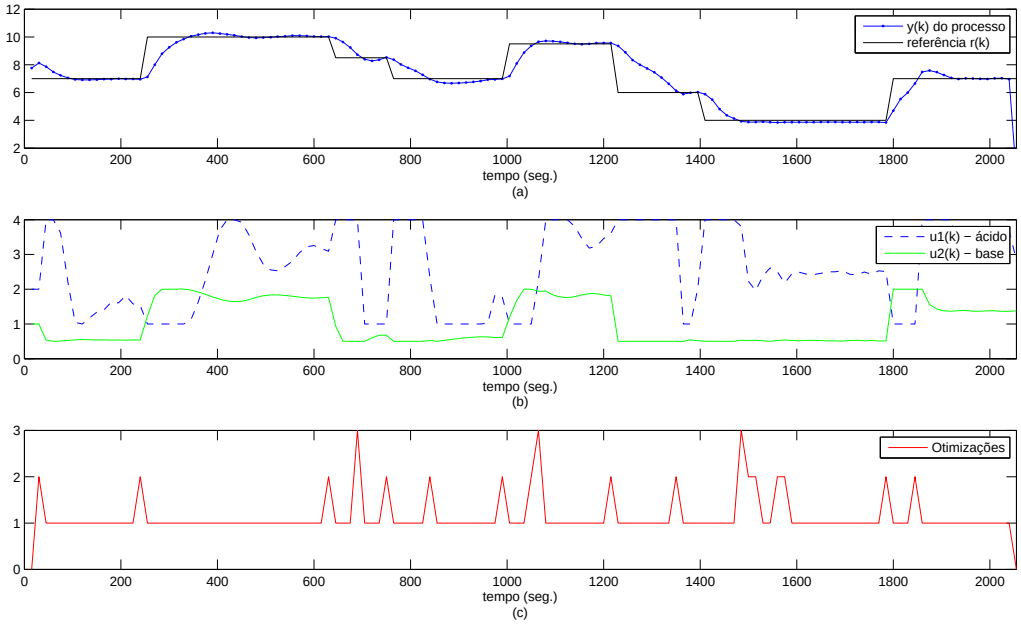


Figura 5.11: Teste de rastreabilidade no processo real. IAE = 54,59.

Como pode ser visto na Figura 5.11, pequenos sobre-sinais apareceram. Os patamares de *pH* 4 e 10 com tempo de duração maior, foram inseridos de forma proposital para verificar o comportamento do neuro-controlador nas regiões de *pH* baixo e alto. O processo se estabiliza nos patamares de forma satisfatória, não apresentando oscilações bruscas, indesejáveis em processos industriais. As oscilações em (b) referentes às ações de controle ácido e base foram

consideradas aceitáveis para a aplicação. Pode ser notado em (b), entre os tempos 400 e 600, e 1500 e 1800 segundos, que as ações de controle tendem a estabilização. Como pode ser visto em (c) poucas iterações são necessárias para o método de Newton-Raphson encontrar a melhor solução dentro das restrições dadas. O índice IAE foi de IAE = 54,59.

- **Teste de travamento**

Este teste simula o travamento físico de um bomba de fluxo no processo real para se observar a capacidade do neuro-controlador em perceber a situação e contornar o problema. Este teste simula casos onde alguma bomba de determinado processo industrial sofra desgastes mecânicos ou problemas de comunicação e pode-se verificar como o neuro-controlador agiria nessas situações. O teste foi realizado enviando um sinal fixo para o processo independentemente do valor calculado pelo neuro-controlador.

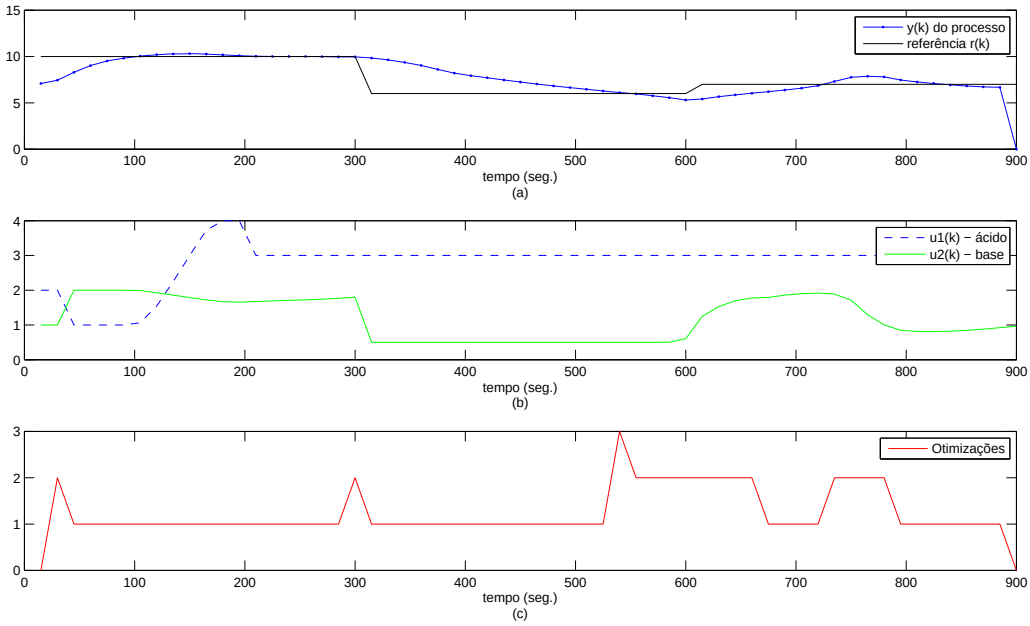


Figura 5.12: Teste de travamento da bomba de fluxo de ácido no processo real. IAE = 48,54.

Na Figura 5.12, a partir do tempo 200 (duzentos) segundos um sinal fixo igual a 3V é enviado para bomba de fluxo de ácido levando seu fluxo ser igual a aproximadamente 2,84 ml/seg, simulando um problema de comunicação ou travamento neste patamar. Durante o decorrer dos

testes o patamar de referência $r(k)$ foi variado de *pH* 10 para 6 e em seguida para 7. Com a bomba de fluxo de ácido travada em 2,84 ml/seg, o neuro-controlador tomou as decisões variando a vazão na bomba de fluxo de base que está restrita a um valor mínimo de vazão de 0,7 ml/seg. Note-se que o neuro-controlador, mesmo nesta situação, realizou modificações na ação de controle respectiva ao fluxo de base a fim de manter o processo na referência.

No teste seguinte é fixado o valor de vazão da bomba de fluxo de base em um valor igual a 0.75V ou 0.96 ml/seg. O valor foi mantido fixo até o final do teste, iniciando-se no tempo igual a 200 (duzentos) segundos. Observando a forma com que o neuro-controlador leva o processo à referência, pode-se notar em termos de rastreabilidade um desempenho melhor (IAE = 36,40) sobre o teste anterior quando a bomba de fluxo de ácido estava com valor fixo. Este desempenho está ligado ao valor de λ_1 que dá mais liberdade de ação para a bomba de fluxo de ácido, e maior rapidez no controle.

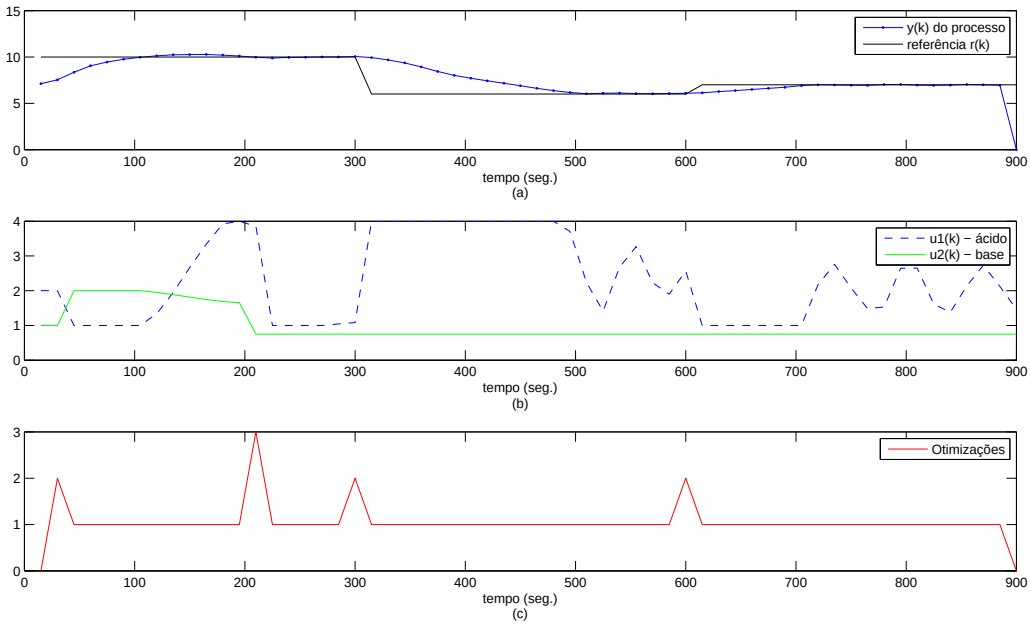


Figura 5.13: Teste de travamento da bomba de fluxo de base no processo real. IAE = 36,40.

- **Teste de mudança de *pH* do fluido a ser tratado**

Nos processos industriais, quando de um processo de neutralização de *pH*, podem haver situações onde diferentes soluções com diferentes valores de *pH* são levadas a tratamento antes

de serem liberadas ao meio ambiente. As soluções precisam ter seu *pH* neutralizado a fim de minimizar o impacto causado. Este teste visa simular esta situação, onde a solução a ser tratada pode muitas vezes vir mais ácida ou mais básica. O resultado do teste pode ser visto na Figura 5.14 e é comentado em sequência.

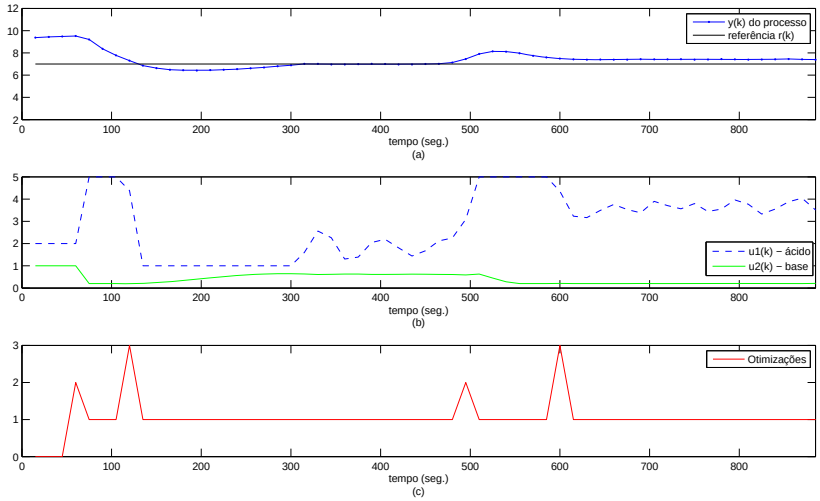


Figura 5.14: Teste de mudança de solução no processo de neutralização. IAE = 31,41.

O teste foi inicializado sob as seguintes condições:

- solução de *pH* 2,7 armazenada no recipiente dito solução ácida;
- solução de *pH* 11,7 armazenada no recipiente dito solução base;
- solução tampão com vazão constante de 0,1 ml/seg.

Após o tempo de 450 (quatrocentos e cinquenta) segundos, retirou-se a mangueira da solução ácida de *pH* 2,7 e a colocou em um recipiente com solução de *pH* 3,5. Após o intervalo de tempo de alguns segundos quando o processo recebe a nova solução no reator, o *pH* começa a subir. A partir dos 500 (quinhentos) segundos de simulação percebe-se a atuação do neuro-controlador com as modificações nas ações de controle. Para esta simulação foi necessário liberar o neuro-controlador para operar no máximo da bomba de fluxo de ácido, pois somente assim foi percebido que o sistema tentava levar o processo ao *pH* 7 para neutralização. Note-se um erro estático final que ficou em torno de de 0,4 com *pH* final de 7,4. O erro

ocorre porque internamente no algoritmo NGPC, o modelo neural do processo havia encontrado a referência (*pH* 7) minimizando a função de custo e assim o neuro-controlador entendia que já havia alcançado seu objetivo. Então, a disparidade do modelo neural treinado *off-line*, com o processo, fica representada pelo erro estático.

5.4 Comentários finais

De forma a verificar o comportamento do neuro-controlador proposto, foram apresentados resultados de simulações realizadas no sistema simulado e no processo real. Testes de rastreabilidade mostraram que o neuro-controlador proposto é capaz de seguir trajetórias diferentes realizando modificações coerentes nas ações de controle. A intenção do trabalho além de cumprir o objetivo de construir um neuro-controlador multivariável, foi de abrir campo para a aplicação de controles deste tipo na planta piloto de neutralização de *pH*.

Dentre os testes, um teste para simular supostas falhas nas bombas de fluxo foi realizado. A idéia é testar se o neuro-controlador consegue perceber se uma bomba de fluxo não está respondendo de forma adequada, e nestas situações, observar a sua atitude. Falhas deste tipo em processos industriais podem ocorrer e se espera que o controlador minimize os efeitos negativos destas falhas, dentro do tempo de percepção da falha e do tempo necessário para sua intervenção.

O teste de rastreabilidade é executado no processo e é observado o comportamento das ações de controle encontradas pelo neuro-controlador. A ação de controle referente ao fluxo de ácido teve maior flexibilidade por causa do fator λ_1. Testes de travamento de bomba demonstraram a capacidade do neuro-controlador em minimizar efeitos indesejáveis, como não seguir a referência desejada.

Outro teste realizado foi a mudança do *pH* da solução ácida levada à neutralização. Como era de se esperar, o erro estático final aparece, devido não existir uma solução adaptativa para compensar o uso de uma solução com *pH* diferente da utilizada no treinamento *off-line* do modelo neural.

No ajuste dos parâmetros, realizado empiricamente, não foi trivial encontrar os valores que permitiram o neuro-controlador atuar de forma satisfatória. Estes parâmetros são fixos para as várias faixas de operação. O processo de neutralização de *pH* é altamente não-linear e oscilações foram frequentes e em maior número durante o ajuste dos parâmetros, até se obter os parâmetros que permitiam resultados satisfatórios em todas as faixas. Possivelmente esta dificuldade em encontrar parâmetros adequados é bastante reduzida em problemas com menor

grau de não-linearidades.

Verificou-se que a mudança na limitação imposta ao número de iterações permitidas pelo otimizador não trouxe resultados satisfatórios o suficiente para utilizar um limitador de maior valor, sendo portanto escolhido neste trabalho o número de 3 (três) as otimizações permitidas.

Quanto a problemas de otimização, há chances do método de Newton-Raphson convergir para mínimos locais apresentando ações de controle menos eficientes, mas pelos resultados obtidos pode-se verificar que não houve um comprometimento do sistema de controle em nenhum dos testes realizados.

Capítulo 6

Conclusões e sugestões para trabalhos futuros

Um modelo neural foi utilizado na arquitetura de controle como modelo explícito do processo. A incorporação satisfatória de dinâmicas do processo a ser controlado e a facilidade de ser obtida face ao grande número de dados disponíveis, foram as características principais para a escolha da RNA como modelo do processo dentro do sistema de controle.

O neuro-controlador proposto tem a característica de utilizar ações de controle futuras calculadas com base em valores preditos na saída do modelo neural do processo. Estas ações de controle futuras vêm do processo de otimização e possuem a capacidade de levar o processo a se comportar de forma desejada e em tempo o menor possível.

A facilidade da classe de controladores MPC trabalhar em âmbito multivariável ajudou na estruturação do sistema de controle proposto neste trabalho, que foi a implementação de um controlador preditivo no controle multivariável de um processo de neutralização de *pH*. O fato também do esforço de controle poder ser inserido na própria função de custo facilitou sua implementação.

O ajuste dos parâmetros λ_1 e λ_2 foi a etapa mais árdua do trabalho. Os ajustes eram simultaneamente realizados de forma empírica e muitos testes foram realizados até se obter uma resposta considerada satisfatória. Um mal ajuste destes parâmetros inviabiliza o controle do processo, sendo o ajuste destes parâmetros uma etapa que exige muita paciência por parte do pesquisador.

Na obtenção do modelo neural, várias sequências de sinais aleatórios tanto em amplitude

como em duração foram gerados e aplicados ao processo a fim de se obter o modelo neural. Teve-se o cuidado também de manter os sinais de excitação com a mais baixa correlação possível entre eles para que o modelo obtido fosse mais próximo possível do processo.

Testes com alguns modelos neurais apresentaram erros estáticos quando da realização do controle do processo. Condição devida a que ao se excitar o processo, algumas regiões de operação eram menos excitadas, fazendo o modelo neural ser tendencioso a outras regiões. Nestas situações uma nova geração de dados de excitação amenizou o problema. A obtenção de um modelo neural em faixa ampla para processos altamente não-lineares como o caso de processos de *pH*, exige um tempo considerável.

A validação em predição livre foi utilizada neste trabalho para verificar se o modelo neural incorporou certas dinâmicas do processo modelado. Isto dá uma medida da capacidade das RNAs em incorporar dinâmicas de processos. Para o modelo neural, um número de apenas 4 neurônios com 2 atrasos de tempo nas variáveis de entrada e saída do processo, foi o suficiente para modelar a planta piloto de neutralização de *pH*.

O neuro-controlador proposto foi aplicado e respectivos testes realizados em um modelo fenomenológico do processo de neutralização de *pH* e também na planta piloto de neutralização de *pH*. Aproveitou-se o processo de neutralização de *pH* por ser um processo considerado *benchmark* quando o assunto é não-linearidades. Assim, poderia levar o controlador a regiões de operação consideradas altamente não-lineares e portanto tirar conclusões mais concretas de seu comportamento.

Verificou-se que a mudança na limitação imposta ao número de iterações (otimizações) permitidas pelo otimizador não trouxe resultados satisfatórios o suficiente, e valor foi fixado em 3 (três) já que foi verificado convergências em média de 2 (duas) iterações.

Durante a realização de testes com alguns modelos neurais obtidos, verificou-se que um novo modelo neural ao ser inserido no algoritmo NGPC, dificilmente permitiu ao neuro-controlador alcançar resultados semelhantes aos obtidos utilizando outro modelo neural do processo. Alguns destes modelos neurais apresentavam boa resposta em predição e mesmo assim não possibilitavam o neuro-controlador obter bons resultados. O método parecia não convergir a uma solução e o controle muitas vezes não era realizado. Uma análise sobre a matriz Hessiana foi realizada já que o método de Newton-Raphson encontra problemas de convergência quando de problemas de condicionamento numérico da mesma. Como os valores dos pesos da RNA são utilizados na computação vetorial e matricial para formação da matriz Hessiana, suspeita-se fortemente que alguns modelos neurais mesmo apresentando boa resposta em predição, dispõem a Hessiana a não ser positiva-definida, condição que leva a um ponto de não-mínimo,

assim impedindo a minimização da função de custo e o consequente controle do processo. Esta hipótese pode ser analisada observando-se as Equações 4.48, 4.47, 4.44, 4.32, 4.38, 4.36 e 4.40.

Dos modelos neurais que possibilitaram a realização do controle pelo algoritmo NGPC, a possibilidade de convergência em mínimos locais do método de otimização de Newton-Raphson não parece ter afetado o desempenho no que se diz respeito em o processo seguir trajetórias estipuladas, tanto que os resultados no Capítulo 5 em cenário de rastreabilidade comprovam isto.

6.1 Sugestões para trabalhos futuros

Durante o desenvolvimento do trabalho, algumas simulações não foram realizadas, até mesmo pelo grande número de simulações que podem ser realizadas no projeto de um neuro-controlador. Assim, deixa-se algumas sugestões para trabalhos futuros:

- Adequação do neuro-controlador proposto para trabalhar com adaptação *on-line*;
- Substituir o modelo neural do processo por outra representação matemática;
- Expandir o número de variáveis do processo a serem controladas como a introdução do fluxo da solução tampão no controle, a fim de comprovar uma generalização para a geração das relações matemáticas da Jacobiana e Hessiana;
- Utilizar a metodologia Otimização por superfície de resposta (Barros Neto et al., 1995) na escolha dos parâmetros λ_1 e λ_2.

Referências Bibliográficas

Agarwal, M. (1997). A systematic classification of neural-network-based control. *IEEE Control Systems Magazine*, 17(2):75–93.

Aguirre, L. (2004). Introdução à identificação de sistemas: técnicas lineares e não lineares aplicadas a sistemas reais. *UFMG, Belo Horizonte Brasil*.

Akesson, B. e Toivonen, H. (2006). A neural network model predictive controller. *Journal of Process Control*, 16(9):937–946.

Andrei, N. (2008). A scaled nonlinear conjugate gradient algorithm for unconstrained optimization. *Optimization*, 57(4):549–570.

Aström, K. e Hägglund, T. (2001). The future of PID control. *Control Engineering Practice*, 9(11):1163–1175.

Barakat, F. R. (2006). Simulations of imitative learning. Dissertação de Mestrado, Department of Computer and Information Science - Norwegian University of Science and Technology.

Barros Neto, B., Scarminio, I., e Bruns, R. (1995). Planejamento e otimização de experimentos. *Editora da Unicamp, Campinas*, 299.

Bharathi, N., Shanmugam, J., e Rangaswamy, T. (2007). Intelligent control of pH in a neutralization process. *Asian Journal of Information Technology*, 6(6):667–673.

Bhat, N. e McAvoy, T. (1990). Use of neural nets for dynamic modeling and control of chemical process systems. *COMP. CHEM. ENG.*, 14(4):573–583.

Böling, J., Seborg, D., e Hespanha, J. (2007). Multi-model adaptive control of a simulated pH neutralization process. *Control Engineering Practice*, 15(6):663–672.

Braga, A. P., Ludermir, T. B., e Carvalho, A. C. (2007). *Redes Neurais Artificiais: Teoria e Aplicações*. LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, RJ, 2^a edição.

Camacho, E. e Bordons, C. (1997). *Model predictive control in the process industry*. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

Camacho, E. e Bordons, C. (2004a). Control predictivo: Pasado, presente y futuro. *Revista Iberoamericana de Automatica e Informatica Industrial*, 1(3):5–28.

Camacho, E. e Bordons, C. (2004b). *Model predictive control*. Springer Verlag.

Campos, R. C. C. (2007). Projeto e construção de planta piloto de neutralização de pH e proposta de metodologia para incorporação de informações auxiliares na identificação narx racional. Dissertação de Mestrado, Centro Universitário do Leste de Minas Gerais - UnilesteMG.

Chen, L. e Narendra, K. (2001). Nonlinear adaptive control using neural networks and multiple models. *Automatica*, 37(8):1245–1255.

Chidrawar, S. e Patre, B. (2008). Generalized predictive control and neural generalized predictive control. *Leonardo Journal of Sciences*, 13:133–152.

Chow, S. e Li, Y. (2009). Model reference control for sirs models. *Discrete and Continuous Dynamical Systems (DCDS-A)*, 24(3):675–697.

Clarke, D. (1988). Application of generalized predictive control to industrialprocesses. *IEEE Control Systems Magazine*, 8(2):49–55.

Clarke, D., Mohtadi, C., e Tuffs, P. (1987). Generalized predictive control, parts 1 and 2. *Automatica*, 23(2):137–160.

Cutler, C. e Ramaker, B. (1980). Dynamic matrix control-a computer control algorithm. Em *Proceedings of the joint automatic control conference*.

Cybenco, G. (1989). Aproximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems*, 2:303–314.

De Keyser, R. e Van Cauwenberghe, A. (1985). Extended prediction self-adaptive control. Em *Identification and system parameter estimation: selected papers from the... IFAC/IFORS Symposium*, page 1255. Published for the International Federation of Automatic Control by Pergamon Press.

Dougherty, D. e Cooper, D. (2003). A practical multiple model adaptive strategy for single-loop MPC. *Control engineering practice*, 11(2):141–159.

Econmou, C., Morari, M., e Palsson, B. (1986). Internal model control 5. extension to nonlinear systems. *Ind Eng Chem Proc Des*, 25:403–411.

Edgar, T. F. e Himmelblau, D. M. (2001). *Optimization of chemical processes*. McGraw-Hill Publishing Co., New York, 2^a edição.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Findeisen, R. e Allgower, F. (2002). An introduction to nonlinear model predictive control. Em *21st Benelux Meeting on Systems and Control*, volume 11. Citeseer.

Findeisen, R., Imsland, L., Allgower, F., e Foss, B. (2003). State and output feedback nonlinear model predictive control: An overview. *European Journal of Control*, 9(2-3):190–206.

Foresee, F. D. e Hagan, M. T. (1997). Proceedings of the 1997 international joint conference on neural networks.

Galán, O., Romagnoli, J., e Palazoglu, A. (2004). Real-time implementation of multi-linear model-based control strategies—-an application to a bench-scale ph neutralization reactor. *Journal of Process Control*, 14(5):571–579.

Garcia, C. e Morari, M. (1982). Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323.

Garcia, C., Prett, D., e Morari, M. (1989). Model predictive control: Theory and practice - a survey. *Automatica (Journal of IFAC)*, 25(3):335–348.

Gustafsson, T., Skrifvars, B., Sandstroem, K., e Waller, K. (1995). Modeling of pH for control. *Industrial & Engineering Chemistry Research*, 34(3):820–827.

Hagan, M., Demuth, H., e Jesus, O. (2002). An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control*, 12(11):959–985.

Hassibi, B., Wolff, G., e Stork, D. (1996). Optimal brain surgeon and general network pruning. *Neural networks theory, technology, and applications*, page 56.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 2^a edição.

Henson, M. e Seborg, D. (1994). Adaptive nonlinear control of a pH neutralization process. *IEEE Transactions on Control Systems Technology*, 2(3):169–182.

Hinton, G. (1989). Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation*, 1(1):143–150.

Huang, J. e Lewis, F. (2003). Neural-network predictive control for nonlinear dynamic systems with time-delay. *IEEE Transactions on Neural Networks*, 14(2):377–389.

Hunt, K., Sbarbaro, D., Żbikowski, R., e Gawthrop, P. (1992). Neural networks for control systems: a survey. *Automatica (Journal of IFAC)*, 28(6):1083–1112.

Itoa, M., Nodaa, K., Hoshinoa, Y., e Tanib, J. (2006). Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19:323–337.

Lawrence, S., Giles, C. L., e Tsoi, A. C. (1998). What size neural network gives optimal generalization? convergence properties of backpropagation. Technical report, Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland.

Le Cun, Y., Denker, J., Solla, S., Howard, R., e Jackel, L. (1990). Optimal brain damage. *Advances in neural information processing systems*, 2(1).

LeCun, Y., Denker, J. S., e Solla, S. A. (1990). Optimal brain damage. Em Touretzky, D. S., editor, *Advances in Neural Information Processing Systems II*, pp. 598–605, San Mateo, CA. Morgan Kauffman.

Luenberger, D. (2008). *Linear and nonlinear programming*. Springer, 3^a edição.

Ma, Z. e Liu, H. (2007). Pipeline defect detection and sizing based on MFL data using immune RBF neural networks. Em *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, pp. 3399–3403.

Maksumov, A., Mulder, D., Harris, K., e Palazoglu, A. (2002). Experimental application of partitioned model-based control to pH neutralization. *Ind. Eng. Chem. Res*, 41(4):744–750.

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Minsky, M. (1963). Steps toward artificial intelligence. Em *Proceedings of the IRE*, Cambridge, Mass. Dept. of Mathematics and Computation Center, Res. Lab. of Electronics, M. I. T.

Minsky, M. e Papert, S. (1969). Perceptrons: An introduction to computacional geometry. *MIT Press, Massachusetts*.

Misra, B., Biswal, B., Dasb, P., e Panda, G. (2007). Simplified polynomial neural network for classification task in data mining. Em *IEEE Congress on Evolutionary Computation, 2007. CEC 2007*, pp. 721–728.

Møller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks-Oxford*, 6:525–525.

Nahas, E., Henson, M., e Seborg, D. (1992). Nonlinear internal model control strategy for neural network models. *Computers & chemical engineering*, 16(12):1039–1057.

Narendra, K. e Parthasarathy, K. (1990). Identification and control of dynamical systems using neuralnetworks. *IEEE Transactions on Neural Networks*, 1(1):4–27.

Newell, A., Shaw, J. C., e Simon, H. A. (1957). Empirical explorations with the logic theory machine: a case study in heuristics. Technical report, RAND CORP SANTA MONICA CALIF.

Nikolaou, M. (2001). Model predictive controllers: A critical synthesis of theory and industrial needs. *Advances in Chemical Engineering*, 26:132–204.

Noor, M., Khor, W., e Yaacob, M. (2004). Fuzzy logic control of a nonlinear pH neutralization in waste water treatment plan. *International Journal of Engineering and Technology*, 1(2):197–205.

Ou, J. e Rhinehart, R. (2003). Grouped neural network model-predictive control. *Control Engineering Practice*, 11(7):723–732.

Oukhellou, L. e Aknin, P. (1997). Modified fourier descriptors: A new parametrization of eddy current signatures applied to the rail defect classification. Em *III International workshop on advances in signal processing for non destructive evaluation of materials*, volume 3, pp. 163–169.

Pam, H. e Don, S. (1997). Neural generalized predictive control: A newton-raphson implementation. Technical report, NASA Langley Research Center, Hampton, VA 23681-0001.

Pam, H., Don, S., e Brian, G. (1999). Real-time adaptive control using neural generalized predictive control. *NASA Langley Technical Report Server*.

Peymani, E., Fatehi, A., Bashivan, P., e Sedigh, A. (2008). An experimental comparison of adaptive controllers on a pH neutralization pilot plant. Em *Annual IEEE India Conference, 2008. INDICON 2008*, volume 2.

Popov, A., Farag, A., e Werner, H. (2005). Tuning of a PID controller using a multi-objective optimization technique applied to a neutralization plant. Em *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05*, pp. 7139–7143.

Pérez, J., Ocaña, J., Molpeceres, C., Morales, M., e Blasco, M. (2009). Adaptive neural network control system for laser surface heat treatments. *The International Journal of Advanced Manufacturing Technology*, 41(5):513–518.

Psaltis, D., Sideris, A., e Yamamura, A. (1988). A multilayered neural network controller. *IEEE Control Systems Magazine*, 8(2):17–21.

Qin, S. e Badgwell, T. (1997). An overview of industrial model predictive control technology. Em *AIChE Symposium Series*, volume 93, pp. 232–256.

Qin, S. e Badgwell, T. (2000). An overview of nonlinear model predictive control applications. *Nonlinear Predictive Control*, pp. 369–392.

Rapaport, W. (2006). Turing test. *Encyclopedia of Language & Linguistics*, 13:151–159.

Richalet, J. (1993). Industrial applications of model based predictive control. *Automatica (Journal of IFAC)*, 29(5):1251–1274.

Richalet, J., Abu, E., Arber, C., Kuntze, H., Jacubasch, A., e Schill, W. (1997). Predictive functional control: application to fast and accurate robots. Em *10th IFAC World Congress*.

Riedmiller, M. e Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. Em *In Proceedings of the IEEE International Conference on Neural Networks*.

Rouhani, R. e Mehra, R. (1982). Model algorithmic control (MAC): basic theoretical properties. *Automatica*, 18(4):401–414.

Rumelhart, D. E., Hinton, G. E., e Willians, R. J. (1986). Learning representations by back-propagation errors. *Nature*, 323:533–536.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210.

Schei, T. (1994). Automatic tuning of PID controllers based on transfer function estimation. *AUTOMATICA-OXFORD-*, 30:1983–1983.

Schnitman, L. e Fontes, A. d. B. (1999). The basic ideas of neural predictive control. Em *Proceedings of the 7th Mediterranean Conference on Control and Automation (MED99) Haifa, Israel*.

Seborg, D. (1999). A perspective on advanced strategies for process control (revisited). *Advances in control (Springer, 1999)*, pp. 103–34.

Shieber, S. M. (2007). The turing test as interactive proof. *Noûs*, 41(4):686–713(28).

te Braake, H., van Can, E., Scherpen, J., e Verbruggen, H. (1998). Control of nonlinear chemical processes using neural models and feedback linearization. *Computers and Chemical Engineering*, 22(7-8):1113–1127.

Teixeira, R. A. (2001). *Treinamento de Redes Neurais Artificais Através de Otimização Multiobjetivo: Uma Abordagem para o Equilíbrio entre a Polarização e a Variância*. Tese de Doutorado, Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais.

Turing, A. M. (1950). Mind - a quarterly review of psychology and philosophy. *Computing Machinery and Intelligence*, LIX(236).

Valarmathi, K., Devaraj, D., e Radhakrishnan, T. (2009). Intelligent techniques for system identification and controller tuning in pH process. *Brazilian Journal of Chemical Engineering*, 26:99–111.

van Overloop, P., Weijs, S., e Dijkstra, S. (2008). Multiple model predictive control on a drainage canal system. *Control Engineering Practice*, 16(5):531–540.

Wang, C., Venkatesh, S., e Judd, J. (1994). Optimal stopping and effective machine complexity in learning. *Advances in Neural Information Processing Systems 6*.

Widrow, B. e Hoff, M. (1960). Adaptive switching circuits, 1960 IRE WESCON convention record. *New York: IRE*, 4:96–104.

Yang, Z. e Hamer, R. (2007). Bio-basis function neural networks in protein data mining. *Current pharmaceutical design*, 13(14):1403–1413.

Ydstie, B. (1984). Extended horizon adaptive control. Em *Proceedings of the 9th IFAC World Congress*, volume 7, pp. 133–138.

Yu, D. e Yu, D. (2005). Modeling a multivariable reactor and on-line model predictive control. *ISA transactions*, 44(4):539–559.

Yu, D. e Yu, D. (2007). Multi-rate model predictive control of a chemical reactor based on three neural models. *Biochemical Engineering Journal*, 37(1):86–97.

Yu, D., Yu, D., Gomm, J., e Williams, D. (2002). Model predictive control of a chemical process based on an adaptive neural network. Em *In Proceedings of IFAC 15 th World Congress, Barcelona, Spain, July*, pp. 21–26.

Zhang, L., Pan, M., e Quan, S. (2008). Model predictive control of water management in PEMFC. *Journal of Power Sources*, 180(1):322–329.

Zitzler, E., Deb, K., e Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195.

Zitzler, E., Laumanns, M., Thiele, L., et al. (2001). Spea2: Improving the strength pareto evolutionary algorithm. Em *EUROGEN*, pp. 95–100.

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