Pós-Graduação em Ciência da Computação

"Estudo Comparativo de Controladores Auto-Organizáveis"

Por

Paulo Henrique Muniz Ferreira

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 2014

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PAULO HENRIQUE MUNIZ FERREIRA

“Estudo Comparativo de Controladores Auto-Organizáveis"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Prof. Aluízio Fausto Ribeiro Araújo

RECIFE, 2014

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da Silva, CRB4-1217

F383e Ferreira, Paulo Henrique Muniz.
Estudo comparativo de controladores auto-organizáveis / Paulo Henrique Muniz Ferreira. – Recife: O Autor, 2014.
110 f.: il., fig., tab.

Orientador: Aluízio Fausto Ribeiro Araújo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIN. Ciência da Computação, 2014.
Inclui referências e apêndices.

1. Inteligência artificial. 2. Inteligência computacional. 3. Sistemas de controle inteligente. I. Araújo, Aluízio Fausto Ribeiro (orientador). I. Título.

006.3 CDD (22. ed.) UFPE-MEI 2014-145

Dissertação de Mestrado apresentada por **Paulo Henrique Muniz Ferreira** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Estudo Comparativo de Controladores Auto-organizáveis**" orientada pelo **Prof. Aluízio Fausto Ribeiro Araújo** e aprovada pela Banca Examinadora formada pelos professores:

_____
Prof. Tsang Ing Ren
Centro de Informática/UFPE

_____
Prof. Adrião Duarte Dória Neto
Departamento de Engenharia de Computação e Automação/UFRN

_____
Prof. Aluizio Fausto Ribeiro Araújo
Centro de Informática /UFPE

Visto e permitida a impressão.
Recife, 1 de setembro de 2014.

_____
Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Aos meus amados pais, Henrique e Socorro, com muito carinho.

Agradecimentos

Agradeço a minha adorável namorada, Beatriz. Seus incentivos fizeram com que eu tivesse estímulo para driblar os desafios, por vezes, tão recorrentes. Como se não bastasse, ela ainda ajudou na revisão do texto (e das equações!). À ela, toda a minha imensa gratidão e amor. Também agradeço a família dela por ter me ajudado nessa dura batalha acadêmica.

Agradeço imensamente a minha tia Regina. Ela abriu a porta do seu lar e me abraçou como eu fosse um filho. Agradeço também a meus pais, Henrique e Socorro, por terem me apoiado e me dado, dia após dia, as condições de chegar até aqui. Além deles, meus sinceros agradecimentos a minha irmã, Ágata Kellen, com quem sempre pude contar em todos os momentos, tanto nos de felicidade quanto naqueles mais difíceis. Sem o esforço deles, de forma alguma eu teria conseguido realizar todas as minhas conquistas.

Os meus colegas de graduação e mestrado, como Bap (Bruno), Dayvid, Denys, João, Henrique, Guilherme, Pt (Pedro Antonino), Ruan e do laboratório, como André Tiba, Hans, Flávia, Monique, Oriva, Júlio, Renata, Tarcísio, contribuíram muito para tornar esse mestrado mais prazerosos e instigantes.

Gostaria de expressar a minha gratidão aos professores da banca avaliadora, Adrião Duarte (UFRN) e Tsang (UFPE), pelo tempo dedicado à leitura e apreciação do meu trabalho e pelas diversas sugestões e críticas que contribuíram para melhorar substancialmente o conteúdo apresentado nesta dissertação.

Por fim, mas com uma imensa importância, agradeço ao meu orientador, Aluízio Fausto Ribeiro Araújo, pela paciência e confiança que em mim depositou e, consequentemente, possibilitou a conclusão deste trabalho.

Obrigado

Virtus in medio

—

Resumo

Sistema de controle empregando redes neurais vem sendo estudado ativamente desde a década de 1980. Mas ainda assim, é um campo bem desafiador devido a diversas formas de abordagens neurais existente na literatura e a natureza diversa dos sistemas de controle que podem apresentar uma dinâmica variando consideravelmente ao longo do regime de funcionamento. Nesse trabalho de mestrado, foi investigada a aplicação de sistemas de controle auto-organizáveis baseados no Mapa Auto-organizável de Kohonen, em duas diferentes proposições. Ambas consistem de controles com arquitetura modulares e se diferenciam na característica da estrutura topológica. O primeiro controle é concebido com uma estrutura fixa. Enquanto o segundo apresenta uma estrutura que cresce a medida que é necessário. Os modelos foram validados em simulações de problemas mecânicos clássicos (linear e não-linear), como sistema massa-mola-amortecedor e pêndulo invertido. Nos experimentos, os sistemas tinham seus parâmetros variando no tempo e os resultados dos métodos propostos foram comparados com métodos convencionais. Os resultados sugerem que as técnicas de controle inspiradas nos mapas auto-organizáveis controlam adequadamente e são eficazes ao lidar com configurações não apresentadas na etapa de treinamento.

Palavras-chave: Controle Auto-Organizável. Mapas Auto-Organizáveis. Controle Inteligente. Controle Adaptativo. Controle Neural. Múltiplos Módulos. Sistema Massa-Mola-Amortecedor. Pêndulo Invertido.

Abstract

Control system using neural networks has been increasely studied since the 1980s. But it is still a very challenging field due to various forms of neural approaches existing in the literature and the diversified nature of the control systems that can produce significant dynamic changes over the operating regime. In this master dissertation, it was investigated the application of self-organizing control systems based on Self-Organizing Maps considering two different propositions. Both cases consist of controls with modular architecture which are distinct on the characteristic of topological structure. The first control is designed with a fixed structure while the second one has a structure that grows as needed. The models were validated using simulations of classical mechanical problems (linear and nonlinear), such as a system mass-spring-damper and inverted pendulum. In the experiments, the system was time-varying and the proposed methods results were compared with conventional methods. The results suggest control techniques inspired by Self-Organizing Maps are proper to control and they are effective when dealing with example not presented during the training phase.

Keywords: Control Self-Organizing. Self-Organizing Maps. Neural Networks Control. Multiple Modules. Mass-Spring-Damper System. Inverted Pendulum.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AG	Algoritmo Genético
BMM	Best Matching Module (*Módulo com Melhor Correspondência*)
CFC	Convencional Feedback Control (*Controle Realimentado Convencional*)
EDE	Equação Diferencial Estocástica
FLC	Fuzzy Logic Control (*Controle de Lógica Nebulosa*)
LARFSOM	Local Adaptive Receptive Field Self-Organizing Map (*Mapa Auto-organizável com Campo Receptivo Adaptativo Local*)
LLM	Local Linear Maps (*Mapas Lineares Locais*)
LQR	Linear-Quadratic Regulator (*Regulador Linear Quadrático*)
MLP	Multilayer Perceptron (*Perceptron Multicamadas*)
mnSOM	Modular Network SOM (*Rede Modular SOM*)
MPFIM	Multiple Paired Forward-Inverse Models (*Múltiplos Modelos de Pares Direto/Inverso*)
NASA	National Aeronautics and Space Administration (*Administração Nacional da Aeronáutica e do Espaço*)
NNC	Neural Network Controller (*Controlador Neural*)
PDA	Proportional-Derivative-Acceleration Controller (*Controlador Proporcional Derivativo Acelerativo*)
PID	Proportional-Integral-Derivative Controller (*Controlador Proporcional Integral Derivativo*)
PSOM	The Parameterized Self-Organizing Map (*Mapa Auto-organizável Parametrizado*)
QSMC	Quasi-Sliding Mode Controller (*Controlador Modo Quasi-Sliding*)
RBF	Radial-Basis Function Networks (*Redes Neurais com Função de Base Radial*)
RMSE	Root Mean Square Error (*Valor Quadrático Médio do Erro*)
SOAC	Self-Organizing Adaptive Controller (*Controlador Adaptativo Auto-Organizável*)
SOFM	Self-Organizing Feature Map (*Mapa Auto-Organizável de Característica*)
SOI	Self-Organizing Input (*Auto-Organização da Entrada*)
SOM	Self-Organizing Map (*Mapa Auto-Organizável*)
TRN	Topology-Representing Networks (*Rede de Representação Topológica*)

VQTAM Vector-Quantized Temporal Associative Memory (*Memória Associativa Temporal por Quantização Vetorial*)

WNN Wavelet Neural Network (*Rede Neural de Wavelet*)

Lista de Algoritmos

Sumário

1 Introdução

Nesse capítulo introdutório são discutidas a motivação e a justificativa do presente Trabalho de Mestrado no contexto da automação inteligente. Também há uma seção 1.2, que descreve o objetivo geral e as metas específicas desse trabalho. Por fim, é apresentada a estrutura deste documento de uma maneira geral.

1.1 Motivação e Justificativa

Desde meados dos anos 80s, os sistemas de controle inteligente vêm sendo estudados e desenvolvidos fortemente. Atualmente, é um dos campos mais investigados dos controles modernos e tem atraído interesse tanto do ponto de vista acadêmico quanto industrial. Essa classe é referente aos paradigmas de controle que utilizam alguma técnica de inteligência artificial ou inteligência computacional como, por exemplo, redes neurais. Entre os principais tipos de sistemas de controle inteligente estão: os Controles Nebulosos (*Fuzzy*), os Controles Especialistas (*Expert*), Controles Evolucionários e os Controles Neurais PASSINO (2010). Todos eles vêm sendo amplamente aplicados em sistemas industriais, aeroespacial, robótico entre outros. No entanto, o campo dos controles inteligentes ainda mantém muitos tópicos desafiantes tanto em termos teóricos quanto práticos. Um deles é a área de conhecimento sobre os controles auto-organizáveis fundamentados na rede neural proposta por Kohonen KOHONEN (1982, 1990), conhecida por Mapa Auto-Organizável (SOM).

O SOM possui características interessantes para ser aplicado no campo de sistemas de controle. Uma delas é o paradigma de aprendizagem intrínseca a essa rede neural. O Mapa Auto-Organizável de Kohonen tem uma aprendizagem não-supervisionada. Nesse paradigma, a estrutura neural aprende através de dados não-rotulados e o objetivo é identificar os diferentes padrões dos dados e construir representações internas desses padrões. Esse tipo de aprendizagem pode simplificar o controlador e, consequentemente, implicar em uma carga menor na programação de um sistema de controle. Ao mesmo tempo, pode melhorar a robustez para lidar com grande volume de dados e redundância de informações. Outro ponto sobre modelo não-supervisionado é que ele pode ser projetado para ter uma aprendizagem incremental. Desta

forma, o sistema de controle pode, ao longo da execução, aprimorar seu desempenho. Essa característica é encontrada em sistemas controlados por um operador humano. A medida que o operador humano vai ganhando experiência, o sistema vai sendo melhor controlado OGATA (2011a).

Alguns sistemas de controle, particularmente os não-lineares, apresentam uma dinâmica tão complexa que um sistema de controle de modelo único, isto é, um modelo global, apresentará dificuldades para operar adequadamente. Para esses sistemas complexos, modelos locais de controle vêm sendo aplicados MURRAY-SMITH; JOHANSEN (1997a,b). Controle de múltiplos modelos pode simplificar bastante tanto a modelagem quanto o projeto do controlador MURRAY-SMITH; JOHANSEN (1997a,b); NARENDRA; BALAKRISHNAN; CILIZ (1995); NARENDRA; BALAKRISHNAN (1997). A ideia de controles de múltiplos modelos se "encaixa" bem com a estrutura dos mapas auto-organizáveis. O SOM pode ser utilizado para dividir o espaço de operação em subespaços e um modelo de controle apropriado é usado em cada região. Tal abordagem está atrelada ao sucesso do uso do Mapas Auto-Organizável em identificação de sistemas dinâmicos não-lineares WALTER; RITER; SCHULTEN (1990); PRINCIPE; WANG (1995); MOSHOU; RAMON (1997); GE; CHIN; WANG (1999); BARRETO (2007). O aspecto mais interessante em modelar um controle de múltiplos modelos com o SOM é que essa rede neural pode ser usada para generalizar os modelos locais de controle aprendidos. O controle poderá ser robusto para variações de parâmetros no sistema e, além disso, poderá também operar para parâmetros não considerados na etapa de treinamento. A motivação para esta pesquisa, portanto, é explorar a estratégia de controle auto-organizável usando alguns modelos baseados no SOM para controle de sistemas lineares e não-lineares.

1.2 Objetivos da Investigação

1.2.1 Objetivo Geral

A pesquisa relatada nessa dissertação tem por objetivo propor modelos de sistema de controle inteligente, mais precisamente, controles auto-organizáveis inspirados no Mapas Auto-Organizável de Kohonen para sistemas de controles lineares e não-lineares com parâmetros invariantes ou variantes no tempo.

1.2.2 Objetivos Específicos

Este trabalho de mestrado considerou 4 metas específicas relacionadas ao contexto de sistemas de controle e controles inteligentes.

- Foi feito um esboço dos principais tipos de controles, como também, um estudo das abordagens neurais para concepção de controles adaptativos.

- Uma implementação de simuladores para o sistema massa-mola-amortecedor e para o sistema não-linear do pêndulo invertido foi realizada nesse trabalho. Ambos simuladores têm a possibilidade de alterar os parâmetros do sistema ao longo do tempo.
- Estudo detalhado e implementação de um controle auto-organizável proposto na literatura foi realizado.
- Foram executados outros estudos detalhados e implementações de alguns controles convencionais, como o Controlador Proporcional Integral Derivativo (PID) e o Controlador Proporcional Derivativo e Acelerativo (PDA), e um controle ótimo, o Regulador Linear Quadrático (LQR), para efeito geração de dados para aprendizagem dos controle ou comparação dos resultados.

1.3 Contribuições

As principais contribuições feitas neste trabalho com relação à modelagem e controle de sistemas lineares e não-lineares incluem:

O desenvolvimento de uma versão do SOM para controle com modelagem em regiões de operação locais. Nessa versão, o SOM é adaptado para identificar os parâmetros do sistema dinâmico. Enquanto em cada região local, tem um controle adequado para o seu ponto de operação.

Também foi contribuído em um projeto de um controle inspirado em uma variante do SOM chamada Mapa Auto-organizável com Campo Receptivo Adaptativo Local (LARFSOM). Nesse projeto de controle, o número de regiões locais cresce à medida que é necessário. Há uma adequação do LARFSOM para identificar as regiões de operação. Enquanto, o comando de controle é calculado através de um modelo linear local com parâmetros ajustados por um algoritmo genético.

Por último, esta dissertação contribuiu com estudos comparativos de três controles auto-organizáveis. Estes estudos comparativos também incluíram métodos convencionais de controle.

1.4 Estrutura da Dissertação

O restante do trabalho se organiza da seguinte forma: No Capítulo 2, apresenta um esboço sobre métodos de controles. Existe também uma descrição das abordagens neurais para concepção de controles adaptativos. No final do Capítulo 2, é apresentada a definição do problema. No Capítulo 3, encontra-se a definição de controle auto-organizável, que foi o objeto de estudo desse trabalho. Há uma discussão sobre vantagens e desvantagens desse tipo de controle e uma revisão bibliográfica desse método de controle. Também é dedicado a uma descrição minuciosa de um controle auto-organizável proposto na literatura. No Capítulo 4,

são apresentados os dois métodos de controle propostos nesse trabalho. Mas, antes, é feita uma descrição das redes neurais usadas para projetar os controles propostos. O Capítulo 5 e 6 são dedicados a apresentar e comentar os resultados obtidos nas simulações dos sistemas Massa-Mola-Amortecedor e Pêndulo Invertido, respectivamente. Por fim, no Capítulo 7, há as considerações finais e propostas de trabalhos futuros.

Esta dissertação conta também com dois apêndices: O Apêndice A é dedicado para detalhar o algoritmo genético usado no segundo controle proposto. Este apêndice conta também com uma breve descrição do Algoritmo Genético Canônico. O Apêndice B apresenta o regulador quadrático linear (LQR) usado nos experimentos do Pêndulo Invertido.

2

Estratégias de Controle para Sistemas Lineares e Não-Lineares Variantes no Tempo

Este capítulo apresenta alguns métodos de controle utilizados diretamente ou indiretamente neste Trabalho de Mestrado. Também, são abordadas as modelagens neurais para controles adaptativos, fundamentais para um entendimento desta dissertação. Por fim, é enunciado o problema de controle inteligente para sistemas lineares e não-lineares variantes no tempo.

2.1 Métodos de Controle

Sistema de controle consiste de uma planta ou processo construído para obter uma resposta de saída desejada com uma performance específica para uma dada entrada NISE (2011). Em geral, sistemas de controle são encontrados em aplicações de diversas áreas como elétrica, mecânica, espacial, biomédica e química NISE (2011). Há, atualmente, uma crescente necessidade de desenvolver controles para operar em sistemas complexos de múltiplas entradas e múltiplas saídas e variantes no tempo OGATA (2011b). Para atender os requisitos cada vez mais rigorosos no desempenho dos sistemas de controle, vêm sendo empregados diferentes esquemas de controle avançado. Alguns deles são: *Controle Adaptativo*, *Controle Ótimo* e *Controle Inteligente* OGATA (2011b); LEVINE (2010); PASSINO (2010); DAHLEH (2010); LEWIS (2010); KRSTIĆ; KOKOTOVIĆ (2010). Cada uma dessas categorias de controle tem suas características particulares. A seguir, será descrita cada uma dessas categorias sucintamente. Também, é apresentado resumidamente o controle convencional Proporcional Integral Derivativo (PID) e o controle nebuloso, que é um classe de controle inteligente.

2.1.1 Controle Convencional

Controle direto e retroalimentado com ganho fixo (*Fixed-gain Feedback and Feedforward Control*) é o controle mais simples e mais usado nos diferentes problemas de automação SUH et al. (2008). Acredita-se que mais de 90% dos processos de controle atuais são do tipo Proporcional

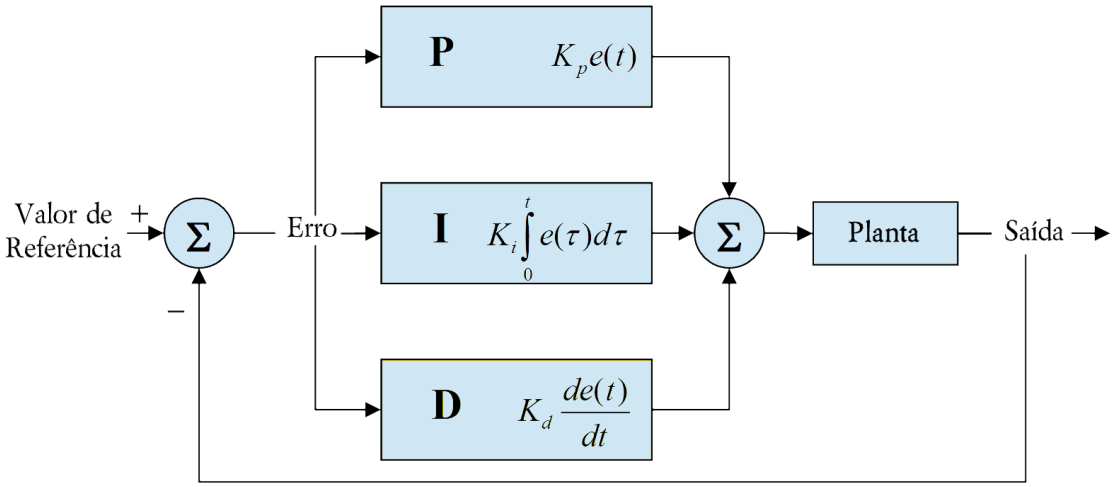

Figura 2.1: Diagrama de blocos de um controle PID.

Integrativo Derivativo (PID) ÅSTRÖM; HÄGGLUND (2001). É comum encontrar essa classe de controle na forma *Proporcional mais Derivativo*, *Proporcional mais Integrativo* ou *Proporcional mais Derivativo mais Integrativo*. Esta última forma é apresentada na Equação 2.1 e na Figura 2.1.

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(t)dt + K_d \cdot \frac{d}{dt}e(t) \tag{2.1}$$

Onde $u(t)$ é o sinal de comando de controle, $e(t)$ é o sinal de erro atuante entre o valor de referência (a saída desejada) e a saída do sistema e K_p é denominado sensibilidade proporcional ou ganho. Já $K_i = \frac{K_p}{T_i}$ e $K_d = K_p \cdot T_d$ onde T_i e T_d representam os tempos integrativo e derivativo, respectivamente.

Uma das principais vantagens ligadas a essa classe de controles é a simplicidade. Isso reflete na sua implementação, que é bastante fácil. Outro ponto relevante, que é consequência de sua simplicidade, é sua capacidade de ser utilizado em esquemas sofisticados de controle. Esse controle, também, tem um boa precisão na posição final, mas tem desempenho fraco para rastrear valores desejados RAINEY III (1990). Esta limitação é observada principalmente para trajetórias desejadas com posições mudando rapidamente. Outra dificuldade encontrada nessa classe de controle é encontrar um ajuste correto para os parâmetros.

2.1.2 Controle Adaptativo

Usualmente, a característica da dinâmica de um sistema a ser controlado não é constante. Isto é devido ao fato que os parâmetros, que definem o sistema, não são constantes. Os parâmetros de um sistema podem se modificar por diversos motivos. A deterioração de seus componentes ao longo do tempo ou variações nas condições do ambiente podem mudar os parâmetros do sistema.

Algumas vezes, os efeitos provocados por pequenas variações nas características dinâmi-

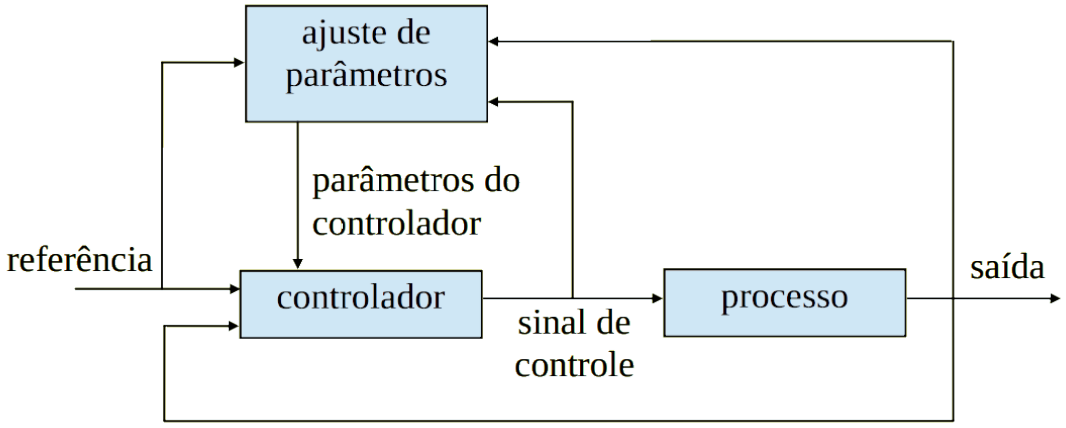

Figura 2.2: Esquema de um controle adaptativo, adaptado de ÅSTRÖM; WITTENMARK (1995).

cas podem ser atenuados por um sistema de controle com realimentação. Contudo, para variações significativas nos parâmetros do sistema, o controle deve possuir a habilidade de adaptação. Esta consiste na capacidade de se autoajustar ou automodificar para se adequar às novas características do ambiente ou da estrutura.

Segundo Aström e Wittenmark ÅSTRÖM; WITTENMARK (1995), um controlador adaptativo é um controlador com parâmetros modificáveis através de mecanismo de ajuste. A Figura 2.2 apresenta um diagrama representando o controle adaptativo. Pode-se notar que em um sistema de controle adaptativo há a necessidade do monitoramento a todo instante do comportamento dinâmico do processo para que se possa modificar corretamente os parâmetros do controle. O conceito de controle adaptativo é bastante atrativo, pois pode responder a variações ambientais e a erros ou incertezas do projeto do controlador. Também compensará falhas de componentes, aumentando, assim, a confiabilidade no sistema OGATA (2011a).

2.1.3 Controle Ótimo

Controle ótimo é um importante componente na teoria de controle moderno. Foi inicialmente desenvolvido na década de 1960 e seu início está ligado a era da corrida espacial, dos computadores e dos robôs LEWIS (2010). Pode-se afirmar que é um ramo da teoria de controle moderno que lida com o projeto de controle para sistemas dinâmicos através da otimização de um índice de performance que, por sua vez, depende das variáveis do sistema. As aplicações do controle ótimo foram bastante significativas no controle de aplicações espaciais, aeroespaciais e militares. No apêndice B, é apresentado o *regulador quadrático linear (LQR)*, que é um controle ótimo usado em alguns experimentos desta dissertação.

Para o problema de controle ótimo, há, tipicamente, uma declaração dos seguintes três elementos:

- Uma modelagem de equações de estados;

- Condições iniciais do sistema;
- Conjunto de objetivos definidos;

O objetivo central na confecção desse controle é encontrar uma lei de controle viável tal que o sistema parta das condições atuais do sistema e atinja o estado alvo otimizando um índice de desempenho.

O controle ótimo consiste de uma extensão do cálculo de variações. Pois, esse problema de otimização está definido em espaços de funções de dimensões infinitas e tem a função objetivo definida por um operador de integração. Há duas poderosas ferramentas para resolver os problemas de conjunto de restrições: Princípio do máximo (ou mínimo) de Pontryagin e Programação dinâmica de Bellman LOCATELLI (2001).

O controle ótimo é muito bem adaptado para problemas aeroespacial e aplicações militares. Aplicações dessa abordagem apareceram no controle de pouso na Lua de uma nave espacial LIU; DUAN; TEO (2008) e no controle de voo de um foguete TEWARI (2011).

2.1.4 Controle Inteligente

Um campo do controle moderno interessante e bastante estudado atualmente é o controle inteligente. Há mais de uma definição para essa classe de controle. Mas pode-se definilo concisamente como um paradigma de controle que propõe usar uma ou mais técnicas de inteligência computacional PASSINO (2010). Esta técnica de controle pode desenvolve ações de controle após a modelagem da planta ou identificação do modelo de planta através de métodos da computação inteligente. Mas também, o controlador inteligente pode ser projetado sem a necessidade de empregar um modelo para a planta. Estes são os controles inteligentes baseados na abordagens controle dirigido por dados HOU; WANG (2013). Em geral, tal tipo de controle pode apresentar as seguintes características ALBUS (1990, 1991):

- Aprendizagem automática;
- Autorreconfiguração;
- Mecanismos de inferência;
- Elaboração e uso de planejamentos;
- Mecanismos de tomada de decisão;
- Capacidade de extrair as informações mais relevantes de dados sem prévio processamento, isto é, de dados bruto sem estarem classificados ou estruturadas e até mesmo com ruído.

Todas essas características dão a esse tipo de controle um alto grau de autonomia. Os sistemas de controles inteligentes podem ter a capacidade de detectar alterações na planta ou

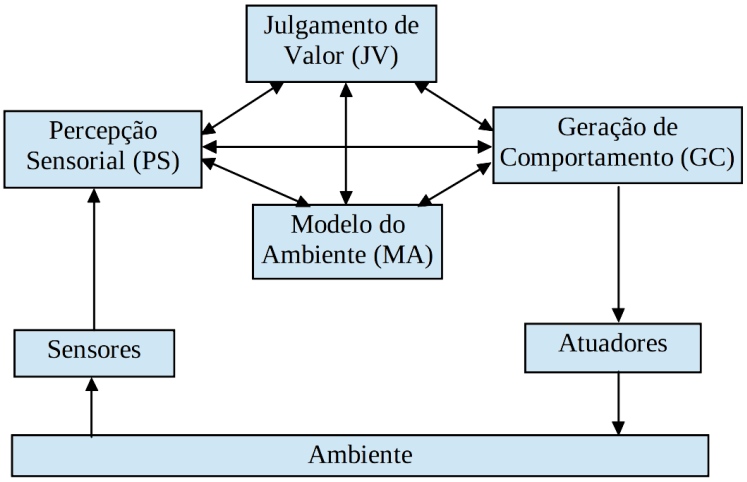

Figura 2.3: Elementos e relações dos elementos de um sistema inteligente. Diagrama adaptado de ALBUS (1990).

até mesmo no ambiente. Podem tomar decisões corretas mesmo na falta de parte da informação ou proveniente de múltiplos sensores. E são capazes de se reconfigurarem na situação em que apresentem falhas internas. Os controles inteligentes são comumente divididos em algumas subclasses. Algumas delas são: Controles Nebulosos (*Fuzzy*), os Controles Especialistas (*Expert*), Controles Evolucionários e os Controles Neurais PASSINO (2010). A próxima seção, é dedica para dissertar sobre abordagens neurais para modelagem de controle.

Há um modelo proposto em ALBUS (1990, 1991) que visa moldar um processo sistematizado para sistemas inteligentes. Na Figura 2.3, é apresentado um diagrama com os elementos do sistema inteligente e as relações entres esses elementos. Há sete elementos básicos, são eles: atuadores, sensores, percepção sensorial (PS), modelo de ambiente (MA), geração de comportamento (GC), julgamento de valor (JV) e comunicação/memória global, que é as conexões entre os setes elementos. Abaixo, é descrito cada elemento.

Atuadores As ações do sistema inteligente no ambiente deriva dos atuadores. O sistema inteligente deve coordenar os diversos atuadores a fim de realizar a tarefa alvo e cumprir um dado objetivo.

Sensores As respostas do ambiente e do estado interno são capturadas pelos diversos sensores do sistema inteligente. Essas entradas observadas são utilizada para a percepção sensorial e podem ser usadas para o monitoramento de ações.

Percepção Sensorial É responsável pela assimilação do mundo externo e interno. As entradas sensoriais são comparadas com as expectativas geradas por um modelo de ambiente. Métodos de processamento sensoriais são usados para integrar similaridades e diferenças entre os dados observados e esperados com objetivo de identificar características, objetos, eventos e relações.

Modelo de Ambiente O sistema inteligente modela o ambiente para criar estimativas do estado do sistema inteligente no ambiente. Esse modelo deve incluir informações sobre o ambiente para gerar as expectativas. Nesse processo de modelagem do ambiente, é possível usar recursos de simulações para construir expectativas e previsões das interações sistema/ambiente. Dessa forma, esse elemento pode fornecer respostas sobre informações futuras do ambiente além das informações presentes lidas pela percepção sensorial e informações passadas armazenadas. Esse elemento é utilizado pelo elemento de percepção para criar correlações entre o ambiente observado e o esperado. Esse elemento também dá suporte ao elemento *geração de comportamento* e *julgamento de valor*. As informações do modelo de ambiente são usadas para criar tarefas e planos inteligentes e para análise de tomada de decisões de ações.

Julgamento de Valor É atribuído a esse elemento o papel de avaliar. Ele é responsável em qualificar se é bom ou ruim, se recompensa ou pune, se é importante ou irrelevante, se é certeza ou improvável as informações, os eventos e as ações passadas, presentes e futuras. Para isso, esse elemento calcula os riscos, custos e benefícios associados as informações observadas do mundo e as expectativas das ações possivelmente tomadas por cada plano. Também pode computar a probabilidade associada ao sucesso de ações e atribuir parâmetros de incertezas sobre variáveis de estado. Dessa maneira, o elemento julgamento de valor também tem a função de criar critérios necessários para tomada de decisão.

Geração de Comportamento O comportamento do sistema é concebido nesse elemento. Sua função é planejar e executar as tarefas. Para isso, esse elemento decompõe as tarefas em subtarefas e as sequenciam para alcançar os objetivos planejados. As escolhas de comportamento são feitas através de um *loop* de interação com os elementos *modelo de ambiente* e *julgamento de ambiente*. Nesse laço, os planos de comportamento são concebidos no elemento *geração de comportamento*. São simulados no elemento *modelo de ambiente* e avaliados pelo elemento *julgamento de valor*. Em seguida, o elemento *geração de comportamento* seleciona os planos com melhores avaliações para execução. Na execução do plano, esse elemento é responsável por monitorar e modificar o plano caso avalie necessário.

2.1.5 Controle Nebuloso

O controle nebuloso, também conhecido por controle de lógica nebulosa (*fuzzy logic control* - FLC), controle difuso ou controle *fuzzy*, é construído sobre bases matemáticas da teoria dos conjuntos e sistemas nebulosos (*fuzzy*) FENG (2006). Ele representa o conhecimento ou experiência em um formato matemático em que processos e características do sistema dinâmico podem ser descritos por conjuntos nebulosos e funções relacionais nebulosas. Decisões de

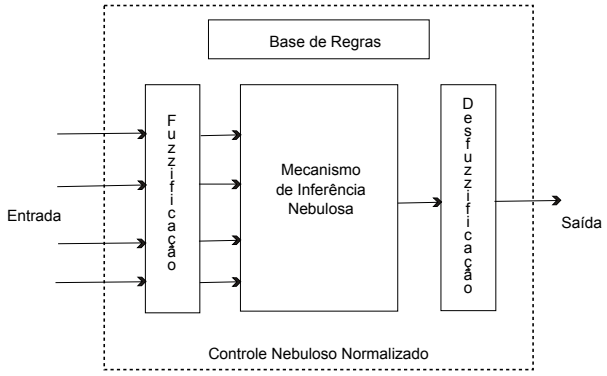

Figura 2.4: Estrutura básica do controle nebuloso, adaptado de PASSINO (2010).

controle podem ser gerados com base nos conjuntos nebulosos e funções com regras. O controle nebuloso é capaz de prover uma metodologia formal para representar, manipular e implementar um conhecimento heurístico humano sobre como controlar o sistema. No projeto de controle nebuloso, há incorporação de conhecimento humano sobre a forma de como controlar um sistema.

O controle nebuloso tem uma arquitetura composta por quatro principais elementos, são eles PASSINO (2010): *Base de regras*, *Mecanismo de Inferência*,*Interface de Fuzzificação* e *Interface de Desfuzzificação*. Na Figura 55.4, é apresentado o diagrama do controle nebuloso básico. Abaixo, é feito uma descrição de cada elemento.

Base de regras Módulo que contém uma quantificação em lógica nebulosa da descrição linguística do especialista de como conseguir um bom controle.

Mecanismo de Inferência Módulo responsável por emular a tomada de decisão do perito na interpretação e aplicação do conhecimento sobre como controlar adequadamente.

Interface de *Fuzzificação* Módulo responsável por converter as entradas do controlador em informações que o mecanismo de inferência pode facilmente usar para avaliar e aplicar as regras.

Interface de *Desfuzzificação* Módulo responsável por converter as conclusões do mecanismo de inferência em real comandos para o sistema.

Controle nebuloso foi originalmente introduzido e desenvolvido como uma abordagem de projeto de controle livre modelo. No início, houve muitas críticas pela falta de análise de estabilidade sistemática. Mas, nos últimos 20 anos, há um esforço em criar controles nebulosos que garantam estabilidade e performance FENG (2006). De fato, o controle nebuloso tem provado ser uma abordagem de controle de sucesso em muitos sistema não-lineares complexos e em sistemas não analíticos FENG (2006).

Sistemas de controle nebuloso foi inicialmente desenvolvido por MAMDANI (1974); MAMDANI; ASSILIAN (1975), onde o controle de um pequeno motor a vapor é considerado.

Desde então, o controle de lógica nebulosa tem atraído grande atenção de ambas as comunidades acadêmica e industrial. Atualmente, o FLC tem sido aplicado com sucesso também em várias áreas, incluindo sistemas de energia, telecomunicações, sistemas mecânicos/robóticos, automóvel, processos industriais/químicos, aeronaves, motores, serviços médicos, eletrônicos de consumo, e em outras áreas como controle de caos e reatores nucleares FENG (2006).

Embora o controle nebuloso tem um grande potencial para resolver problemas complexos de controle, o seu processo de projeto de controle é complicado e requer um grande conhecimento. Além disso, a matemática nebulosas não pertence ao campo da matemática uma vez que muitas operações matemáticas básicas não existem. Por exemplo, a adição inversa não está disponível em matemática *fuzzy*. Então, é muito difícil de resolver uma equação nebulosa. Isso inviabiliza a solução de uma equação diferencial que é uma das práticas básicas de teoria de controle e aplicações de controles. Portanto, a falta de boas ferramentas matemáticas é um problema fundamental para o controle nebuloso.

2.2 Abordagens Neurais para Modelagem de Controle

Do ponto de vista computacional, muitos aspectos de modelagem e controle podem ser capturados pelo modo de computação dos modelos de redes neurais. Também é interessante, no uso de modelos neurais, a possibilidade de ter inspirações nas descobertas da aprendizagem de sistema neurais naturais. Esse fato pode ser observado na abordagem de *aprendizagem por realimentação de erro*. Tal metodologia é inspirada na aprendizagem cerebelar KAWATO; GOMI (1992a,b).

Há uma crescente necessidade de controladores atuando em tarefas cada vez mais complexas e com alta precisão nos problemas atuais da engenharia de controle OGATA (2011b). Modelos conexionistas podem ser usados para desenvolver uma lei de controle que cumpra requisitos mais sofisticados. Isso pode ser justificado pelas características das redes neurais HUNT et al. (1992). Tais modelos podem operar sobre ambiente impreciso e com parâmetros variantes no tempo. Outro ponto atrativo, na utilização de metodologias usando redes neurais, é a capacidade de identificação de mudanças na dinâmica em tempo de execução e se adapta para funcionar sobre esse novo comportamento do sistema. Os modelos neurais também são capazes de modelar um controle com poucas restrições à dinâmica do sistema. Por fim, controles com abordagens neurais podem vir a ter vantagens em termos de sensibilidade à precisão numérica e a ruídos, devido ao poder de generalização intrínseco ao algoritmo de aprendizagem do modelo. Os modelos neurais são capazes de aprender as informações relevantes do ambiente.

A estrutura mais comum para controles com abordagens neurais é baseada em dois componentes principais:

1. Um controlador que gera os comandos de controle a ser enviado para o sistema controlado, e

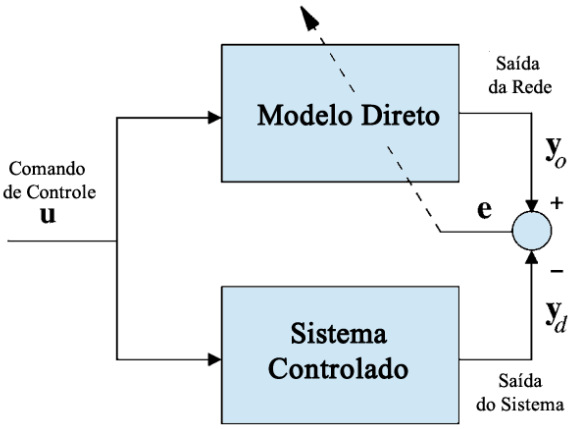

Figura 2.5: Esquema de modelagem direta, adaptado da Figura 1 de BARRETO; ARAúJO; RITTER (2003).

2. Um sistema controlado (por exemplo, uma mão robótica), que converte um comando de controle em uma resposta do sistema.

Nas próximas subseções, há uma descrição de diferentes esquemas de modelagem do controlador neural.

2.2.1 Modelagem Direta

O procedimento de treinamento de uma rede neural para representar a dinâmica direta do sistema controlado (planta ou processo) é referido como *modelagem direta* (Figura 2.5) BARRETO; ARAúJO; RITTER (2003). O modelo de rede neural é colocado em paralelo com o sistema e o erro $\boldsymbol{e}$ entre a saída do sistema $\boldsymbol{y}_d$ e a resposta da rede $\boldsymbol{y}_o$ é, normalmente, utilizado para guiar o treinamento da rede. Durante uma série de tentativas de aprendizagem do modelo direto, a rede aprende a imitar o sistema controlado. Uma vez que o modelo direto é suficientemente preciso, ele pode ser usado para prever o comportamento da planta para uma determinada sequência de comandos de controle ou ser utilizado no treinamento de um controle aprendiz.

2.2.2 Modelagem Inversa

Há dois métodos de modelagem do inverso do sistema, são eles PSALTIS; SIDERIS; YAMAMURA (1988); PRABHU; GARG (1996): *Modelagem inversa direta* e *modelagem inversa especializada*.

Modelagem Inversa Direta

Conceitualmente, a abordagem mais simples para a modelagem inversa é chamada de *modelagem inversa direta* (*direct-inverse modeling*), mas também é referida como *aprendizagem inversa generalizada* PSALTIS; SIDERIS; YAMAMURA (1988); PRABHU; GARG (1996).

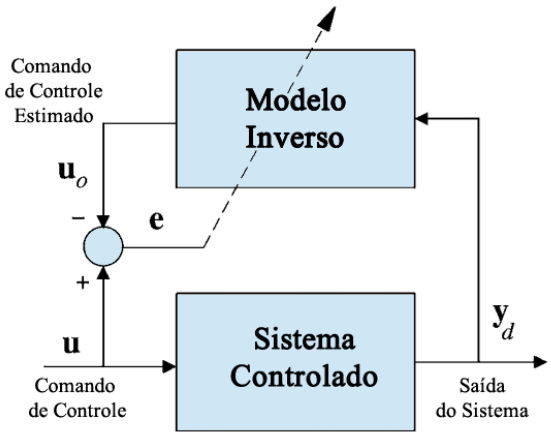

Figura 2.6: Esquema de modelagem inversa direta, adaptado da Figura 2a de BARRETO; ARAúJO; RITTER (2003).

A abordagem consiste em ativar o sistema controlado ou um modelo que o represente com comandos de controle de $\boldsymbol{u}$ gerados aleatoriamente (Figura 2.6). A saída do sistema controlado é utilizada como entrada para o modelo inverso, que, por sua vez, calcula um comando de controle estimado $\boldsymbol{u}_o$. O comando de controle estimado é, então, comparado com o comando de controle real $\boldsymbol{u}$, e um erro $\boldsymbol{e}$ é calculado. O erro $\boldsymbol{e}$ é usado para ajustar o modelo inverso, por exemplo, o erro orienta a atualização dos pesos de uma rede neural, até que, a rede aprenda o comportamento inverso do sistema controlado.

A rede treinada pode, então, ser usada como um controlador. O modelo inverso é simplesmente concatenado, como em um sistema de circuito aberto, com o sistema controlado. Dessa maneira, é construído um mapeamento identidade entre a resposta desejada $\boldsymbol{y}_{ref}$, que é a entrada da rede, e a saída do sistema controlado $\boldsymbol{y}$. Conceitualmente, essa abordagem é bastante simples, mas, na prática, esse esquema de aprendizagem apresenta algumas dificuldades, particularmente, em problemas de controle com redundância não-linear. Isso é bem exemplificado no *problema de Bernstein* que trata da redundância do sistema motor BERNSHTEIN (1967). Ele denominou redundância como a possibilidade de mais de um comando motor poder movimentar um determinado seguimento para uma mesma posição, ou por uma mesma trajetória no espaço. Nessa abordagem de modelo inverso da planta, não há uma exploração dos benefícios dessa redundância. Ao invés disso, há uma tentativa de eliminá-la na etapa de treinamento construindo apenas uma única solução.

Modelagem Inversa Especializada

Um segundo método de aprendizagem do modelo inverso é conhecido como modelagem inversa especializada PSALTIS; SIDERIS; YAMAMURA (1988); PRABHU; GARG (1996). Essa abordagem apresenta duas fases. Na primeira, o modelo direto da planta é treinado de modo que a rede aprenda o comportamento da planta. Na segunda, o controlador neural é ajustado para que se torne o modelo inverso do objeto controlado (Figura 2.7). Nesta fase, os pesos do

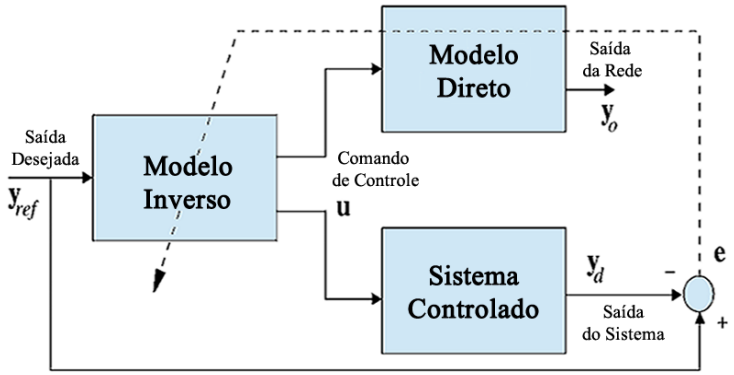

Figura 2.7: Esquema de modelagem inversa geral, adaptado da Figura 2b de BARRETO; ARAúJO; RITTER (2003).

modelo direto são fixados, enquanto os pesos da rede neural do controlador são adaptados. Esta abordagem pode ser usada com o paradigma de aprendizagem incremental ou o não-incremental. Como a rede opera apenas nas regiões de especialização, tal abordagem provê um modo de lidar com sistemas redundantes.

Também é interessante combinar os dois métodos de modelagem inversa. Pode-se usar as duas abordagens de forma que uma complemente a outra. Pode usar primeiro a modelagem inversa generalizada para aprender o comportamento aproximando do sistema a ser controlado. Logo em seguida, usaria a modelagem inversa especializada para um ajuste fino da rede em determinados intervalos de operações BARRETO; ARAúJO; RITTER (2003).

2.2.3 Modelagem por Realimentação de Erro

Originalmente, a abordagem de *aprendizagem por realimentação de erro* foi proposta a partir de uma perspectiva biológica. Alguns modelos internos do sistema nervoso central atuam como um controlador realimentado e as células de Purkinje, neurônios diferentes presentes somente no cerebelo e responsáveis pela realimentação, respondem sinais que representam o erro do comando motor ao invés de representar o erro da trajetória do movimento KAWATO; GOMI (1992a). Isto é, o cerebelo aprende a coordenar os movimentos a partir do erro de comando de força e não a partir de erro na trajetória do movimento. Essa modelagem tenta estabelecer um modelo computacional da aprendizagem do controle motor ocorrida no cerebelo KAWATO; FURUKAWA; SUZUKI (1987).

Na abordagem computacional, o erro da trajetória é convertido pelo controlador realimentado em erro de comando de controle. Por sua vez, este erro é utilizado para guiar a aprendizagem do *modelo inverso*. O modelo inverso recebe como entrada a trajetória desejada $\boldsymbol{y}_{ref}(t)$ e monitora o comando de controle retroalimentado para utilizá-lo como sinal de erro. O modelo inverso faz modificações internas de forma que minimize o sinal de erro, ou seja, é esperado que o sinal de comando de controle do controlador realimentação tenda a zero através do processo de aprendizagem.

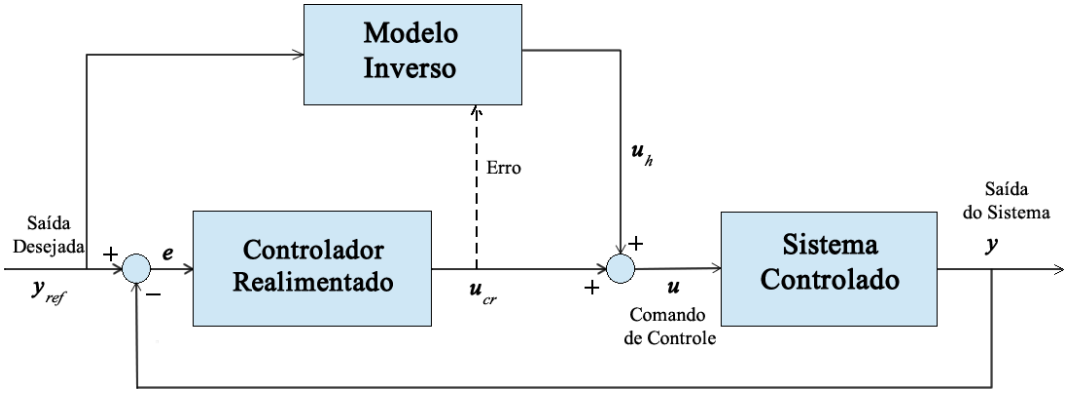

Figura 2.8: Esquema de modelagem por realimentação de erro, adaptado da Figura 1c de KAWATO (1990a). Essa abordagem enfatiza a importância do uso do sinal de comando motor do controle retroalimentado como o sinal de erro.

Na Figura 2.8, é apresentado o diagrama de blocos dessa modelagem. O comando de controle aplicado ao sistema controlado é calculado pela soma de duas componentes. A primeira é o comando de controle do controlador realimentado, normalmente, um controlador convencional como o PID. A segunda é o comando de controle gerado pelo *modelo inverso*. Essa abordagem foi aplicada em diversos trabalhos MIYAMOTO et al. (1988); KATAYAMA; KAWATO (1990); GOMI; KAWATO (1993); WOLPERT; KAWATO (1998); KIM; IM; LEE (2008); MINATOHARA; FURUKAWA (2011); GOPALAN; DEISENROTH; PETERS (2013); SABAHI; GHAEMI; PEZESHKI (2014). Em KAWATO (1990a), há um estudo teórico da estabilidade dessa metodologia de controle. Maiores detalhes dessa abordagens veja KAWATO; FURUKAWA; SUZUKI (1987); KAWATO (1990b,a); KAWATO; GOMI (1992a).

2.3 Controle Inteligente para Sistemas Lineares e Não-Lineares Variantes no Tempo

Como dito anteriormente, sistema de controle consiste de uma planta ou processo construído para obter uma resposta de saída desejada. Sem perda de generalidade, pode-se representar o sistema dinâmico pela seguinte equação de estados THAMPI et al. (2002):

$$\boldsymbol{y}(t+1) = \boldsymbol{f}(\boldsymbol{w}(t), \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.2}$$

onde os vetores $\boldsymbol{y}(t)$, $\boldsymbol{x}(t)$ e $\boldsymbol{u}(t)$ representam a resposta, o estado atual e a entrada do sistema, respectivamente. Enquanto $\boldsymbol{w}(t)$ refere-se aos parâmetros que define função $\boldsymbol{f}$ e t denota o instante de tempo. Já $\boldsymbol{f}$ é uma equação vetorial linear ou não-linear com parâmetros variantes no tempo. Antes de prosseguir, é apresentado conceitos básicos de sistemas dinâmicos OGATA (2004).

Estado O estado do sistema dinâmico é constituído pelo menor conjunto de variáveis, conhecidas por *variáveis de estado*, de modo que como conhecimento destas variáveis no instante

t_0 e a entrada no instante t ($t \geq t_0$), pode-se determinar completamente a saída do sistema no instante t.

Sendo assim, o estado do sistema dinâmico em t é determinado univocamente pelo valores da variáveis no instante t_0 e a entrada no instante t ($t \geq t_0$), e é independente do estado e entrada para $t < t_0$. Normalmente, ao lidar com sistema lineares e invariantes no tempo, é escolhido o tempo t_0 como zero.

Variáveis de estado As variáveis de estados do sistema dinâmico são o menor conjunto de variáveis que determina o estado do sistema dinâmico.

Vetor de Estado O estado do sistema dinâmico é constituído por um conjunto de n variáveis de estado. Essas variáveis podem ser consideradas como as n componentes de um vetor $\boldsymbol{x}(t)$. Tal vetor é chamando de *vetor de estado*.

Espaço de Estado O espaço de estado é um espaço n-dimensional onde n é um número de variáveis de estado. Cada eixo de coordenada representa uma variável de estado e qualquer estado do sistema pode ser representado por um ponto no espaço de estado.

Estabilidade A estabilidade do sistema pode ser *interna* ou *externa*. Neste trabalho, os sistema foram analisados sobre a ótica de estabilidade externa. Na estabilidade externa, o sistema é avaliado aplicando entradas limitadas. Se cada entrada limitada aplicada ao terminal de entrada do sistema resultar em uma saída limitada, é dito que o sistema é *externamente estável*. A estabilidade externa foi escolhida para análise de estabilidade, pois pode-se ser verificada analisando somente o comportamento da entrada e saída do sistema. Esse tipo de estabilidade é comumente conhecida como estabilidade no sentido BIBO (*Bounded-input/Bounded-output*).

Controlabilidade Um sistema é dito controlável no instante t_0 se é possível, usando-se um vetor de controle arbitrário, sem qualquer restrição, transferir o sistema de qualquer estado inicial $\boldsymbol{x}(t_0)$ para qualquer outro estado em um intervalo de tempo finito.

Observabilidade Um sistema é dito observável no instante t_0 se é possível determinar o estado inicial $\boldsymbol{x}(t_0)$ do sistema a partir da observação da saída em um intervalo de tempo finito.

2.3.1 Projeto de Controle

O objetivo do projeto de controle é determinar uma lei de controle que gere entradas para sistema $\boldsymbol{u}$ (comando de controle) de tal forma que a saída do sistema $\boldsymbol{y}(t)$ siga um comportamento desejado. Nesse trabalho de mestrado, os problemas usados nos experimentos foram problemas de controle de movimento. Então, o objetivo do controlador consistiu em gerar comandos de força que faça o sistema controlado rastear uma especificação de comportamento. Os sistemas nos quais a especificação é um estado estacionário são classificados como sistemas de reguladores

automáticos OGATA (2011a). Caso a especificação seja um comportamento dinâmico da posição, velocidade ou aceleração de sistema mecânico, é dito que é um problema de rastreamento OGATA (2011a). Neste trabalho, todos os dois experimentos foram problemas de servomecanismos. Outra peculiaridade dos experimentos, é que os sistemas tinham seus parâmetros variando no tempo. Isso foi feito para simular um comportamento mais complexo da dinâmica do sistema.

Para esse tipo de problema que apresenta a necessidade de lidar com sistemas não-lineares e variantes no tempo é adequado projetar um controle com duas fases: A primeira fase é a identificação. Nesta fase, é identificada a dinâmica do sistema. Já a segunda fase é a de controle. Nesta fase, é gerada a ação de controle apropriada para atender as especificações de saída do sistema. Também, é interessante que a identificação e o controle sejam realizados simultaneamente.

Outro ponto crucial, nesse projeto de controle, é o conhecimento sobre a função $\boldsymbol{f}$ e seus parâmetros. Nem sempre $\boldsymbol{f}$ e $\boldsymbol{w}(t)$ são conhecidos. Para esse caso, as abordagens neurais de controles adaptativos, apresentadas nesse capítulo, são bastante convenientes visto que há apenas as informações de entrada e saída da função $\boldsymbol{f}$. Essas abordagens também são aplicadas adequadamente para o caso no qual $\boldsymbol{f}$ é uma função vetorial não-linear que torna o projeto de controle difícil. Em ambos os casos, $\boldsymbol{f}$ e $\boldsymbol{w}(t)$ são desconsiderados e é construída uma aproximação da função $\boldsymbol{f}$ para projetar o controlador. Há duas abordagens para conceber a função aproximada: Uma é aproximar $\boldsymbol{f}$ através de uma único modelo não-linear $\tilde{\boldsymbol{f}}$. Essa primeira abordagem é comumente chamada de modelagem global. É conveniente utilizar rede neural para aproximar $\boldsymbol{f}$ devido à sua capacidade de mapeamento universal HAYKIN (2009a). A outra abordagem é dividir o espaço em regiões locais e modelar localmente a dinâmica da função $\boldsymbol{f}$. Matematicamente, é feita uma decomposição da função $\boldsymbol{f}$ em uma série de funções $\tilde{\boldsymbol{f}}_r$, com $r = 1,...,N$. Dessa forma, o modelo "global" é constituído pela união de todas as funções $\tilde{\boldsymbol{f}}_r$ PRINCIPE; WANG; MOTTER (1998).

$$\tilde{\boldsymbol{f}}(\boldsymbol{x}(t),\boldsymbol{u}(t)) = \bigcup_{r=1,...,N} \tilde{\boldsymbol{f}}_r(\boldsymbol{x}(t),\boldsymbol{u}(t)) \tag{2.3}$$

Desta segunda maneira, a complexidade dos controladores locais pode ser potencialmente mais simples do que o controlador global do primeiro. Por exemplo, pode-se usar controladores lineares localmente. A etapa de identificação e organização dos controles podem ser realizadas pelo Mapa Auto-Organizável ou por uma variante dessa rede. O SOM é apropriado para criar uma representação do espaço de operação da dinâmica do sistema e organizar os diferentes regimes em vizinhanças topológicas THAMPI et al. (2002).

3

Abordagens de Mapas Auto-Organizáveis para Controle

3.1 Introdução

Neste capítulo, haverá uma análise da literatura existente sobre sistemas de controle com auto-organização inspirados na rede proposta por Kohonen KOHONEN (1982, 1990). Pelos conceitos apresentados no Capítulo 2, esse tipo de controle está inserido no grupo dos controles inteligentes. Primeiramente, são apresentadas as principais características e suas limitações para esse tipo de controle. Depois, são descritos sucintamente alguns sistemas de controle auto-organizáveis. Logo em seguida, é detalhado um controle dessa classe que foi objeto de estudo deste mestrado. Por último, é apresentada uma conclusão do modelo estudado em detalhes neste trabalho.

3.2 Controle Auto-Organizável

O Mapa Auto-Organizável de Kohonen é um modelo de rede neural bastante estudado e com um grande número de aplicações em diversos campos, tais como reconhecimento de padrões, processamento de sinais, otimização, controle de processos e controle motor KOHONEN (2001a,b); KASKI; KANGAS; KOHONEN (1998). Essa rede também fornece um modelo matemático abstrato para uma determinada estrutura cerebral dos humanos que é capaz de realizar um mapeamento topográfico dos sensores visuais (sensores na retina) com o córtex cerebral YIN (2008); CHUANG et al. (2007). Essa rede também é chamada de Mapas Auto-Organizáveis (SOM) ou Mapa Auto-Organizável de Característica (SOFM). Será detalhada no próximo capítulo.

Na subseção 2.1.2, ao discutir sobre controle adaptativos, foi apresentado que controles desta classe têm a característica de se automodificar ou auto-ajustar. Na literatura, alguns autores utilizam o termo *self-organizing* para designar essa característica. Mas está fora do escopo desse capítulo, os controles que se "auto-organizam" e não foram concebidos dos mapas auto-

organizáveis (SOM). Algumas propostas de controle auto-organizável sem empregar o SOM são PROCYK; MAMDANI (1979); SHAO (1988); LIAN (2011); LIAN; HUANG (2012); LIAN (2012, 2013); WONG (2013). Então, o termo controle auto-organizável citado nesse capítulo é usado para designar os controles inteligentes baseados na rede neural de Kohonen (SOM) ou em uma variante dela. No próximo capítulo, também é apresentada uma variante da rede SOM em detalhes.

A primeira característica sobre os controles auto-organizáveis baseados na rede de Kohonen está no seu paradigma de aprendizagem. Esse tipo controle herda, normalmente, o paradigma de aprendizagem da rede SOM, que é chamado de aprendizagem não-supervisionada. Por sua vez, a aprendizagem não-supervisionada tem características bastante atraentes para serem empregadas em sistemas de controle. Na aprendizagem não-supervisionada, a rede neural não necessita de um "professor" externo para informar as respostas corretas. A rede SOM é capaz de se *auto-organizar* para construir representações internas dos padrões de entrada sem um "supervisor". Esta habilidade de *auto-organização* pode simplificar o controlador e, consequentemente, implicar em uma carga menor na programação de um sistema de controle. Consequentemente, o custo da implementação do controle pode ser reduzido. A aprendizagem não-supervisionada é adequada para problemas com grande volume de dados e com redundância de informações. Outro ponto sobre modelos não-supervisionados é que eles podem ser confeccionados para uma aprendizagem incremental. Desta forma, o sistema de controle pode, ao longo da execução, aprimorar seu desempenho.

Em virtude do poder de generalização do SOM, sistemas de controle baseados nessa rede podem ter a capacidade de se adaptar a mudanças significativas nos parâmetros do sistema ou a alterações do ambiente do sistema. Essa particularidade é também importante em diversos sistemas de controle. O controle pode ser capaz de controlar o sistema, mesmo para parâmetros não apresentados na etapa de treinamento.

O SOM também tem outro atrativo. Essa rede tem a capacidade de converter relações não-lineares em simples relações espaciais. Visto que seu objetivo principal é construir uma transformação não-linear dos padrões de entrada de dimensão arbitrária em um mapa discreto normalmente unidimensional ou bidimensional HAYKIN (2009b). Ele é capaz de construir a transformação adaptativamente e com preservação da ordenação topológica dos padrões de entrada.

O ponto negativo presente nos controles auto-organizáveis é uma perda na precisão devido a uma discretização do espaço de comandos de controle. Porém, essa imprecisão pode ser superada de duas maneiras: pode-se ampliar o conjunto de treinamento aumentando a taxa de amostragem e associar a um número maior de protótipos de parâmetros armazenados na rede ou pode-se projetar o controle a partir de uma variante da rede SOM que possui a característica de interpolação da saída da rede.

3.2.1 Trabalhos Anteriores

No trabalho de Coiton *et al.* COITON et al. (1991), o SOM foi usada para controle de posicionamento de braço robótico. Foi proposto um modelo neural com duas camadas, uma camada sensorial e outra, motora. A camada sensorial foi modelada por um SOM e seu objetivo era aprender e integrar informações proprioceptivas e exteroceptivas do braço mecânico. Por fim, há uma ponderação da camada sensorial pelas ligações com a camada motora capaz de gerar comandos para movimentar o braço para a posição desejada. Jones e Verno JONES; VERNON (1994) também apresentaram um Mapa Auto-Organizável capaz de controlar um braço robótico associando estímulos proprioceptivos e exteroceptivos. Sem nenhuma informação a *priori* da cinemática do braço, o controle auto-organizável foi capaz de coordenar o braço para rastrear o alvo apresentado visualmente. Kihl *et al.* KIHL et al. (1995) também propuseram um esquema semelhante ao de Jones e Verno para posicionamento do braço mecânico utilizando uma rede SOM. Em seus resultados, houve aprendizagem rápida do espaço de controle mesmo sem conhecimentos a *priori* de procedimentos de calibração e modelos matemáticos. Buessler *et al.* BUESSLER et al. (1999) construíram um encadeamento de mapas auto-organizáveis para aprender correlações sensoriais-motoras em uma plataforma robótica.

Sbarbaro e Bassi apresentaram um outro controle baseado no SOM. O papel da rede era particionar as regiões de operação de um processo químico não-linear. Principe *et al.* PRINCIPE; WANG; MOTTER (1998) propuseram uma estratégia de controle preditivo de múltiplos modelos locais através de uma modelagem dinâmica local com uma rede SOM. Esse controle foi aplicado ao problema não-linear de identificação e controle de túnel de vento durante teste aerodinâmicos em aeronaves. Thampi *et al.* THAMPI et al. (2002) atacaram o problema de modelagem e controle de sistemas não-lineares utilizando mapas de Kohonen. O método proposto foi esquematizado em múltiplos módulos de identificadores e controladores. O papel do SOM era modelar a dinâmica através do seu poder de representação dos dados de entrada. Cho *et al.* CHO et al. (2006) propuseram um controle no esquema de modelagem linear local através de um Mapa Auto-Organizável. A função da rede era identificar a região de operação do sistema e selecionar adequadamente o controle linear para controlar o sistema. Cho *et al.* CHO et al. (2007) também trabalharam em um controle para uma classe geral de sistemas não-lineares de tempo discreto utilizando a rede SOM para dividir o espaço de estados em regiões de operação. Em cada região, há um controlador local modo *quasi-sliding* (*local Quasi-Sliding Mode Controller - QSMC*).

Zeller *et al.* em ZELLER; WALLACE; SCHULTEN (1995) estudaram controle de movimentos de membro robótico com aplicação da rede SOM. Em outros trabalhos, Zeller *et al.* ZELLER; SHARMA; SCHULTEN (1996, 1997) atacaram o mesmo problema, mas dessa vez projetaram um controle com uma variante do SOM chamada Rede de Representação Topológica (*Topology-Representing Networks* - TRN). Walter e Ritter WALTER; RITTER (1996) investigaram o potencial do uso de uma variante da SOM chamada Mapa Auto-organizável Parametrizado (*The parameterized self-organizing map* - PSOM) para controlar o robô 3-DOF

PUMA. Ritter *et al.* RITTER; MARTINETZ; SCHULTEN (1989) investigaram uma extensão rede SOM para coordenação de um braço robótico simulado. Através de uma extensão da rede SOM chamada Mapas Lineares Locais (*Local linear maps* - LLM), o controle foi capaz de prover comandos de força adequados para movimentos suaves do braço. Barreto e Araújo em BARRETO; ARAúJO (2004) apresentaram uma técnica de modelagem geral chamada Memória Associativa Temporal por Quantização Vetorial (*Vector-Quantized Temporal Associative Memory* - VQTAM) para identificação e controle de sistemas dinâmicos.

Laflamme *et al.* LAFLAMME; SLOTINE; CONNOR (2011) apresentaram um controle neural de Wavelet para operar em sistemas incertos com atuadores semi-ativo. Esta condição aparece em cenário de controle de estrutura civil de larga escala como pontes e viadutos. Esse controle tem o artifício de auto-organização obtido através do SOM. Em outro trabalho, Laflamme *et al.* LAFLAMME; SLOTINE; CONNOR (2012) também propuseram um novo modelo de rede neural inspirado nos mapas auto-organizáveis para controle estrutural. O ponto fundamental desse trabalho esteve no algoritmo de auto-organização da entrada (Algoritmo Self-Organizing Input - SOI). Ele foi responsável por ajustar a camada do espaço de entrada. Já a única camada escondida chamada de Rede Neural de Wavelet (*Wavelet Neural Network* - WNN) foi organizada pelo Mapa Auto-Organizável. Gao *et al.* DAYUAN; ZHENG; HAI (2013) projetaram um controle auto-organizável no esquema de múltiplos módulos para manobra de veículo aéreo não-tripulado. O mapa auto-organizável foi responsável por dividir o espaço de operação. Em cada região, havia um modelo linear aproximado da cinemática não-linear do veículo. Sendo assim, o controlador era projetado baseado no modelo linear local. Também, foi proposto um comutador adaptativo fundamentado no SOM. Os resultados simulados sugeriram uma aumento na velocidade de resposta e na precisão do controle.

Minatohara e Furukawa MINATOHARA; FURUKAWA (2011) trabalharam em um controlador que atendesse a dois requisitos importantes para sistema de controle complexos: (1) uma resposta rápida para mudanças repentinas no objeto controlado e (2) uma aquisição de um comportamento geral a partir de um pequeno número de padrões de treinamento. Para isso, propuseram o Controlador Adaptativo Auto-Organizável (*Self-Organizing Adaptive Controller* - SOAC). Esse controlador foi projetado em uma uma arquitetura de múltiplos módulos para atender a primeiro condição. Enquanto, a segunda propriedade foi atingida através do poder de generalização da rede SOM. Mais precisamente nesse ponto, o SOAC foi elaborado sob uma rede variante do SOM chamada rede modular SOM (mnSOM) TOKUNAGA; FURUKAWA (2009). Cada módulo consiste de um par preditor/controlador. O preditor pode ser visto como um identificador do comportamento da planta. Já o controle a ser aplicado é determinado a cada instante e é escolhido através do preditor que mais se aproxima da planta. Esse controle foi estudado em detalhes e implementado nesse trabalho de mestrado. Uma descrição detalhada está na próxima seção. Foi escolhido o SOAC, pois é um controle projetado para propósito geral, apresentou bons resultados nos seus experimentos e suas limitações ainda não foram exploradas por outras pesquisas. Também foi selecionado porque atende dois requisitos importantes citados

no início desse parágrafo. Fato este não abordado explicitamente nos outros trabalhos anteriores. Estes dois requisitos são, também, alvo das propostas desse trabalho de mestrado.

3.3 Controlador Adaptativo Auto-Organizável - SOAC

O Controlador Adaptativo Auto-Organizável (*Self-Organizing Adaptive Controller - SOAC*) é um controle projetado para atender a duas características importantes para o contexto de sistemas de controle modernos. As duas propriedades são: uma rápida respostas a mudanças inesperadas no objeto controlado; e a aquisição de comportamento geral a partir de uma amostra pequena de exemplos. Esse método de controle é arquitetado usando o esquema modelos múltiplos em um mapa auto-organizável (SOM) para atender esses dois requisitos. A ideia central é usar essas duas abordagens, pois a primeira apresentou bons desempenhos com objetos previamente assumidos e aprendidos em WOLPERT; KAWATO (1998), e a segunda proporciona um alto poder de generalização. O SOAC foi concebido sobre a variante da SOM chamada de rede modular SOM (mnSOM TOKUNAGA; FURUKAWA (2009)). A rede modular SOM pode ser considerada uma generalização da SOM. No caso, cada nodo da mnSOM é denominado de módulo e é constituído de uma rede neural conveniente ao problema. Em geral, é usada uma MLP, RBF ou até mesmo uma SOM. Mais especificamente, é adotado um par preditor/controlador como o módulo da mnSOM. O preditor vai atuar como um modelo do objeto controlado. O controle, obviamente, vai gerar o comando de controle.

3.3.1 Estrutura do SOAC

A estrutura do SOAC é baseada no SOM, mais especificamente no mnSOM. A estrutura é composta por um conjunto de módulos arranjados em uma malha bidimensional (Figura 3.1). Tem-se que cada módulo é constituído por um par preditor/controlador (Figura 3.2).

O papel de cada preditor, como foi dito anteriormente, é modelar a planta para uma dada região de configuração. Isto é, ele é responsável por predizer o comportamento da planta para uma certa configuração dela. As entradas do preditor são o sinal de estado atual do objeto controlado $\boldsymbol{x}(t)$ e o sinal de controle atual $\boldsymbol{u}(t)$. Cada preditor gera uma previsão com Δt de tempo adiante. A equação do preditor pode ser expressa por:

$$\widetilde{\boldsymbol{x}}^k(t+\Delta t) = {}^p\boldsymbol{f}^k({}^p\boldsymbol{w}^k, \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.1}$$

onde ${}^p\boldsymbol{w}^k$ é o vetor de parâmetros da função de aproximação ${}^p\boldsymbol{f}^k$.

Já as entradas do controlador são o sinal de estado atual do objeto controlado $\boldsymbol{x}(t)$ e o estado desejado $\hat{\boldsymbol{x}}(t)$.

$$\boldsymbol{u}^k(t) = {}^c\boldsymbol{f}^k({}^c\boldsymbol{w}^k, \boldsymbol{x}(t), \hat{\boldsymbol{x}}(t)) \tag{3.2}$$

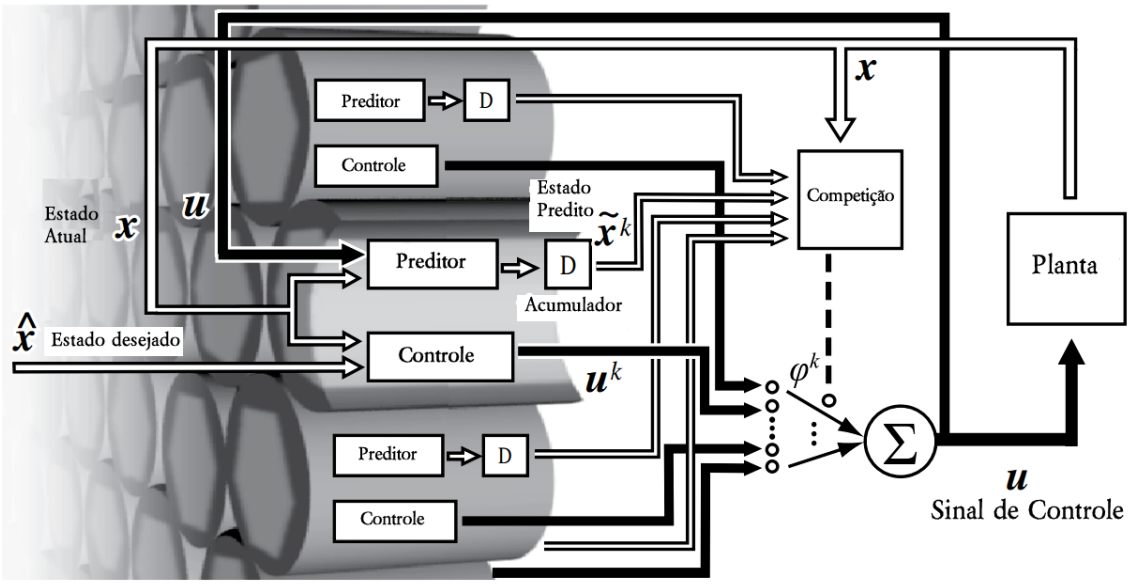

Figura 3.1: Estrutura do SOAC, adaptada de MINATOHARA; FURUKAWA (2011).

onde ${}^{c}\boldsymbol{w}^{k}$ é o vetor de parâmetros da função de aproximação ${}^{c}\boldsymbol{f}^{k}$. O cálculo do erro do preditor é feito usando uma média ponderada dos erros das predições passadas.

$$ {}^{p}e^{k}(t) = (1-\varepsilon) * {}^{p}e^{k}(t-\Delta t) + \varepsilon * \left\| \boldsymbol{x}(t) - \widetilde{\boldsymbol{x}}^{k}(t) \right\| \tag{3.3} $$

No caso, é feito uma ponderação do erro atual e do acumulado dos erros passados. O termo ε determina o grau de esquecimento dos erros passados. Um valor alto para ε dá um comportamento de rápida cumutação entre módulos. Porém, valores muito alto para ε provoca uma característica instável na seleção do módulo.

O módulo vencedor será aquele que minimizar o erro de predição. Então, tem-se:

$$ k^{*}(t) = \underset{k}{arg\,min}\ {}^{p}e^{k}(t) \tag{3.4} $$

Também, há um computação do *sinal de responsabilidade* de um determinado nodo. Esse sinal representa a influência do módulo no comando de controle e na aprendizagem. O cálculo do sinal de responsabilidade é dado por:

$$ \phi^{k} = \frac{exp[\frac{-\left\| \boldsymbol{\varepsilon}^{k} - \boldsymbol{\varepsilon}^{k^{*}} \right\|^{2}}{2\sigma_{\infty}^{2}}]}{\sum_{k'} exp[\frac{-\left\| \boldsymbol{\varepsilon}^{k'} - \boldsymbol{\varepsilon}^{k^{*}} \right\|^{2}}{2\sigma_{\infty}^{2}}]} \tag{3.5} $$

onde $\boldsymbol{\varepsilon}^{k}$ e $\boldsymbol{\varepsilon}^{k^{*}}$ são as coordenadas do k-ésimo módulo e do vencedor no mapa, respectivamente, σ_{∞} é o valor final do raio da função de vizinhança.

O comando de controle total é calculado a partir da soma dos produtos de saída de todos os controladores $\boldsymbol{u}^{k}(t)$ ponderados pelo respectivo sinal de responsabilidade ϕ^{k}.

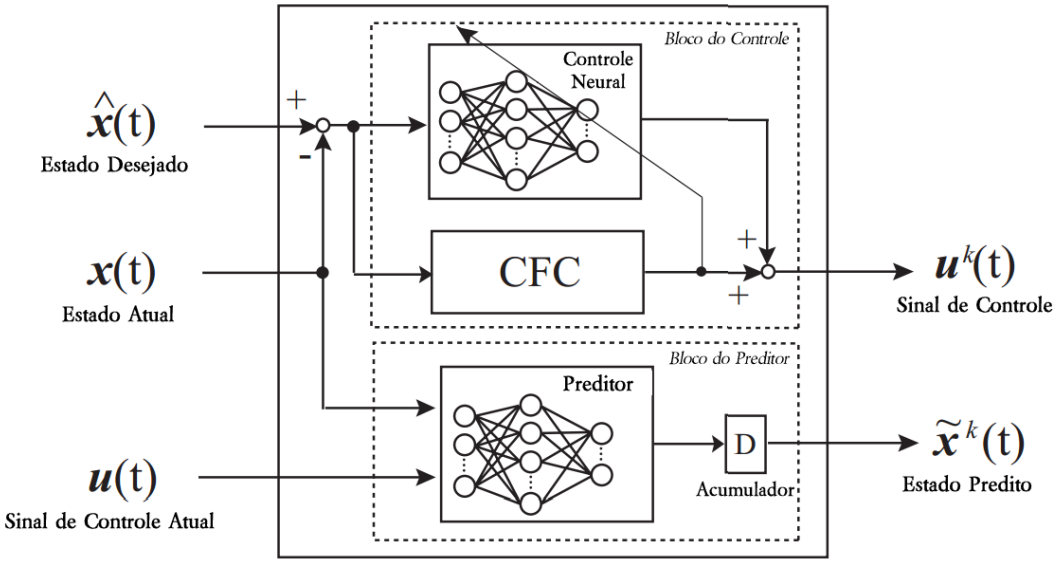

Figura 3.2: Módulo preditor/controlador do SOAC, adaptado de MINATOHARA; FURUKAWA (2005). No bloco do controle, há um controle realimentado convencional (CFC) usado para guiar a aprendizagem do controlador neural.

$$\boldsymbol{u}(t) = \sum_{k=1}^{K} \phi^k \boldsymbol{u}^k(t) \tag{3.6}$$

onde K é o número de módulos. Dependendo da tarefa, pode-se fazer:

$$\boldsymbol{u}(t) = \boldsymbol{u}^{k^*}(t) \tag{3.7}$$

Com isso, foi definido fase de execução. Já a fase de aprendizagem, descrita nas próximas subseções, consiste na fase de determinação das funções ${}^p\boldsymbol{f}^k$ e ${}^c\boldsymbol{f}^k$.

3.3.2 Processo de Aprendizagem do Preditor

Nessa subseção, são descritos detalhes da fase de aprendizagem do modelo da planta, isto é, como é feito o treinamento do preditor. A aprendizagem do preditor é realizada de modo *não-incremental*, isto é, antes da fase de execução. O algoritmo de aprendizagem para o preditor é o mesmo usado no mnSOM. Esse algoritmo possui quatro processos: *Avaliativo, Competitivo, Cooperativo* e *Adaptativo*. Sem perda de generalidade, pode-se assumir que um preditor é composto por um MLP e, portanto, os parâmetros ${}^p\boldsymbol{w}^k$ da função ${}^p\boldsymbol{f}^k$ são os valores do vetor de pesos da MLP.

Antes de detalhar o algoritmo de aprendizagem, é explicado o conjunto de treinamento. Assuma que há um conjunto de I objetos a serem controlados. Todos os objetos controlados deste conjunto têm configurações paramétricas distintas. Estes conjunto é usado para construir o conjunto de treinamento. As sequências de treinamento $\{\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)\}_{i=1,\cdots,I}$ são obtidas a partir de uma sequência de comando de controle arbitrária $\{\boldsymbol{u}_i(t)\}$. A mesma sequência $\{\boldsymbol{u}_i(t)\}$ é

aplicada a todos objetos controlados.

3.3.2.1 Processo de Avaliação

Primeiro, o erro de predição entre a saída do preditor e os dados de treinamento para todas as sequências de treinamento é calculado da seguinte forma:

$$ {}^{p}E_i^k = \frac{1}{T}\int_0^T \left\| \boldsymbol{x}_i(t) - \tilde{\boldsymbol{x}}_i^k(t) \right\|^2 dt \tag{3.8} $$

onde, $\tilde{\boldsymbol{x}}_i^k(t)$ e ${}^{p}E_i^k$ são a saída do k-ésimo preditor e o erro médio do preditor para o i-ésimo sistema do conjunto de treinamento, respectivamente.

3.3.2.2 Processo de Competição

Depois de calcular o erro de predição, no processo anterior, o módulo que tem o valor mínimo de erro de predição é determinado. Chama-se esse módulo de vencedor.

$$ k_i^* = \arg\min_k {}^{p}E_i^k \tag{3.9} $$

3.3.2.3 Processo de Cooperação

O sinal de responsabilidade (ou taxa de aprendizagem) é determinado usando a função de vizinhança que é descrita abaixo:

$$ \psi_i^k = \frac{exp[\frac{-\left\|\boldsymbol{\varepsilon}^k - \boldsymbol{\varepsilon}_i^{k^*}\right\|^2}{2\sigma^2(n)}]}{\sum_{i'=1}^{I} exp[\frac{-\left\|\boldsymbol{\varepsilon}^k - \boldsymbol{\varepsilon}_i^{k^*}\right\|^2}{2\sigma^2(n)}]} \tag{3.10} $$

onde $\boldsymbol{\varepsilon}^k$ e $\boldsymbol{\varepsilon}_i^{k^*}$ são as coordenadas do k-ésimo módulo e do módulo vencedor no mapa espacial, respectivamente. Além disso, $\sigma(n)$ é a função do raio da vizinhança e decresce monotonicamente com o crescimento passo de aprendizagem n. Por exemplo, a seguinte equação pode ser usada com uma função para $\sigma(n)$ com uma constante de tempo τ.

$$ \sigma(n) = \sigma_\infty + (\sigma_0 - \sigma_\infty) exp(-\frac{n}{\tau}) \tag{3.11} $$

Aqui, σ_0 e σ_∞ são, respectivamente, o valor inicial e final de σ durante a fase de aprendizagem.

3.3.2.4 Processo de Adaptação

Finalmente, o vetor de pesos do preditor é atualizado usando a taxa de aprendizagem obtida no processo cooperativo.

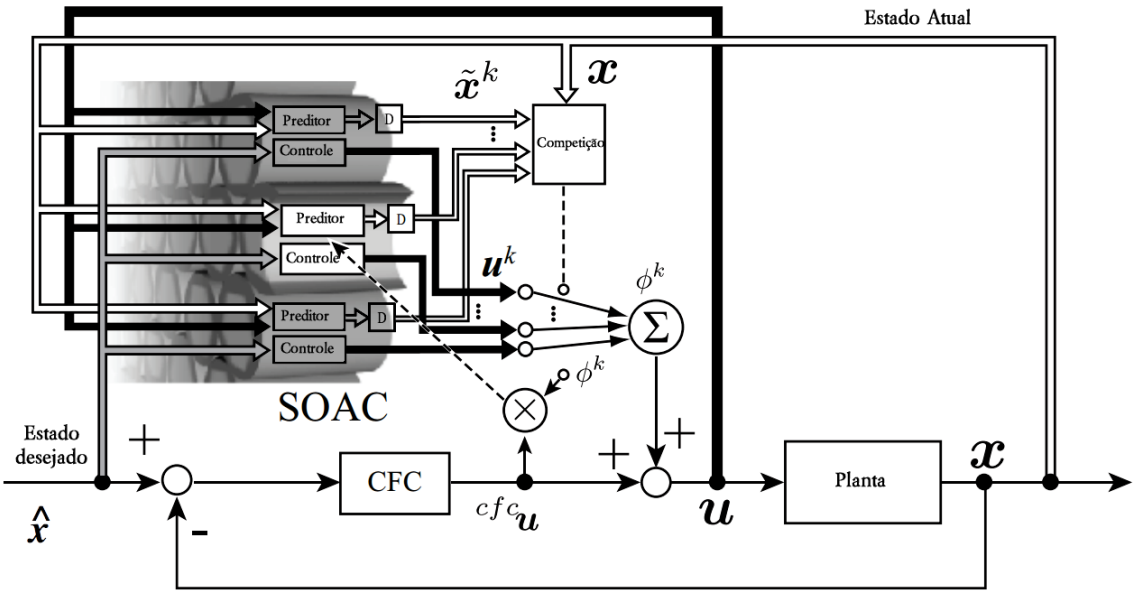

Figura 3.3: Estrutura do SOAC com aprendizagem por realimentação de erro, adaptada de MINATOHARA; FURUKAWA (2011).

$$\Delta^p \boldsymbol{w}^k = -\eta \sum_{i=1}^{I} \psi_i^k \frac{\partial\, {}^pE_i^k}{\partial\, {}^p\boldsymbol{w}^k} \tag{3.12}$$

onde η é um valor constante positivo. Estes quatros processos são iterados até a rede atingir uma condição estacionária. O resultado deste procedimento de aprendizagem será a construção de um mapa espacial com módulos de propriedades similares localizados próximos. Sendo assim, o mapa auto-organizável refletirá várias configurações da dinâmica.

3.3.3 Processo de Aprendizagem do Controlador

Em casos gerais, é usada a regra de comando de controle gerada pela Equação 3.6 ou pela Equação 3.7. Em particular, se um controle analítico é projetado para ser emparelhado ao preditor, então a Equação 3.7 pode ser usada. Isso foi feito no experimento do pêndulo invertido (Capítulo 6). Entretanto, caso contrário, o controlador tem de ser gerado através de aprendizagem. Há várias abordagens para adquirir um modelo da dinâmica inversa (i.e., controlador) através de aprendizagem supervisionada. Tais modelagens foram apresentadas na Seção 2.2. Um diagrama esquemático do SOAC com aprendizagem por realimentação de erro é mostrado na Figura 3.3. O sistema consiste de um sistema controlado, um controle com realimentação convencional (CFC) e K módulos, cada um composto por um preditor e um controle neural para controle direto, sobre uma estrutura de um Mapa Auto-Organizável.

O aprendizado dos pesos da rede através da abordagem de realimentação de erro é realizado de modo *incremental* usando cfc. A lei de controle realimentado pelo cfc é representado da seguinte forma:

$$^{cfc}\boldsymbol{u}(t) = {}^{cfc}\boldsymbol{W}(\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)) \tag{3.13}$$

onde $^{cfc}\boldsymbol{W}$ é um matriz de coeficientes de realimentação. Se um sistema SISO é usado, o comando de controle de realimentação com a regra de controle Proporcional Derivativo Acelerativo (PDA) pode ser escrito da seguinte forma:

$$^{cfc}u(t) = K_P(\hat{x}(t) - x(t)) + K_D(\hat{\dot{x}}(t) - \dot{x}(t)) + K_A(\hat{\ddot{x}}(t) - \ddot{x}(t)) \tag{3.14}$$

onde $[K_P, K_D, K_A]$ é um vetor em que cada elemento corresponde ao ganho de realimentação para o procedimento $^{cfc}\boldsymbol{W}$ e $\boldsymbol{x}$ é composto da posição x, velocidade $\dot{x}$ e aceleração $\ddot{x}$ do sistema.

Por outro lado, a entrada do k-ésimo controle neural (*NNC*) é o estado desejado. Note que nesse caso um *NNC* atua como um controlador direto e o estado atual não é usado como uma entrada. Então, $^{c}\boldsymbol{f}^{k}$ na Equação 3.2 é reescrita da seguinte forma:

$$\boldsymbol{u}^{k}(t) = {}^{c}\boldsymbol{f}^{k}({}^{c}\boldsymbol{w}^{k}, \hat{\boldsymbol{x}}(t)) \tag{3.15}$$

O comando de controle total é calculado a partir da equação:

$$\boldsymbol{u}(t) = \sum_{k=1}^{K} \phi^{k}\boldsymbol{u}^{k}(t) + {}^{cfc}\boldsymbol{u}(t) \tag{3.16}$$

No estado inicial da aprendizagem, o controle é praticamente feito somente pelo cfc. Com o passar do tempo, o processo de aprendizagem vai modificando os controles neurais e gradualmente iniciam suas performances. A regra de ajuste para cada *NNC* é definida por uma ponderação do sinal de erro realimentado normalizado. A ponderação é feita pela multiplicação da sinal do erro vezes o sinal de responsabilidade ϕ^{k} (Equação 3.5). Na Equação 3.17, tem-se a regra de aprendizagem para cada *NNC*.

$$\Delta^{c}\boldsymbol{w}^{k} = \eta \cdot \phi^{k} \frac{\partial\, {}^{c}f^{k}}{\partial\, {}^{c}\boldsymbol{w}^{k}} \cdot {}^{cfc}\boldsymbol{u}(t) \tag{3.17}$$

3.3.4 Mapa de Parâmetros

Foi descrito, nas seções anteriores, como o método SOAC trabalha para construir cada par preditor/controlador. Foi visto que isso é feito através de um processo auto-organizável. É interessante também construir, através do mesmo processo, um estrutura que armazena informações relevantes sobre o objeto controlado. Tais informações, em geral, são os parâmetros físicos da planta. Essas informações estimadas são interessantes, pois podem dar um maior poder de análise do sistema e podem ser, inclusive, usadas para determinar o módulo vencedor no processo de avaliação e competição, nos casos em que há o conhecimento dos valores paramétricos da planta de teste.

Representado através de $\boldsymbol{p}$ o vetor de parâmetros associado a configuração do objeto controlado, pode-se usar $\boldsymbol{p}_i$ para representar a configuração associada ao i-ésimo objeto controlado (normalmente, o i-ésimo objeto de treinamento). Pode-se, para cada módulo, representar o vetor de parâmetro estimado por $\tilde{\boldsymbol{p}}^k$. Na Equação 3.18, tem-se a forma de calcular a estimativa de $\tilde{\boldsymbol{p}}^{\boldsymbol{k}}$.

$$\tilde{\boldsymbol{p}}^{\boldsymbol{k}} = \sum_{i=1}^{I} \psi_i^k \boldsymbol{p}_i \tag{3.18}$$

onde ψ_i^k é o sinal de responsabilidade calculado através da Equação 3.10. Então, o conjunto $\{\tilde{\boldsymbol{p}}^k\}, (k = 1, \cdots, K)$ é chamado de *mapa de parâmetros*. Esse cálculo é realizado após a construção dos preditores. Dessa maneira, não é necessário nenhum calculo especial para computar o mapa de parâmetros em paralelo a aprendizagem. Isso também mostra que o mapa de parâmetros está intrinsecamente associado a geração automática do mapa de preditores.

3.3.5 Pseudocódigos do SOAC

O algoritmo do SOAC consiste de uma fase de aprendizagem e uma fase de execução. Na fase de aprendizagem, os preditores são ajustados. Na fase de execução, são gerados os comandos de controle e pode haver alguma aprendizagem contínua do controlador.

Algoritmo 3.1: Fase de aprendizagem do SOAC

Dados: $S_{treinamento} = \{\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)\}, (i = 1, ..., I)$ conjunto de treinamento
$\{\boldsymbol{p}_i\}, (i = 1, ..., I)$ conjunto de parâmetros associado ao objeto controlado i

Resultado: $SOAC = \{^p\boldsymbol{w}^k, ^c\boldsymbol{w}^k\}, (k = 1, .., K)$ mapa de módulos preditor/controlador com os preditores treinados $P = \{\tilde{\boldsymbol{p}}^{\boldsymbol{k}}\}, (k = 1, .., K)$ mapa de parâmetros estimados

Início

inicialização dos parâmetros do SOAC;

Inicialização();

$n = 0$;

para $n < N$ **faça**

Calcular o erro de predição;

$$^pE_i^k = \frac{1}{T}\int_0^T \left\|\boldsymbol{x}_i(t) - \tilde{\boldsymbol{x}}_i^k(t)\right\|^2 dt;$$

Determinar o BMM;

$$k_i^* = \underset{k}{agr\,min}\ ^pE_i^k;$$

Calcular o raio de vizinhança;

$$\sigma(n) = \sigma_\infty + (\sigma_0 - \sigma_\infty)exp(-\tfrac{n}{\tau});$$

Calcular o sinal de responsabilidade;

$$\psi_i^k = \frac{exp[\frac{-\left\|\boldsymbol{\varepsilon}^k - \boldsymbol{\varepsilon}_i^{k^*}\right\|^2}{2\sigma^2(n)}]}{\sum_{i'=1}^{I} exp[\frac{-\left\|\boldsymbol{\varepsilon}^k - \boldsymbol{\varepsilon}_i^{k^*}\right\|^2}{2\sigma^2(n)}]};$$

Atualizar o vetor de peso de cada preditor;

$$\Delta^p\boldsymbol{w}^k = -\eta\sum_{i=1}^{I}\psi_i^k\frac{\partial\,^pE_i^k}{\partial\,^p\boldsymbol{w}^k};$$

fim para

Calcular o mapa de parâmetros;

$$\tilde{\boldsymbol{p}}^{\boldsymbol{k}} = \sum_{i=1}^{I}\psi_i^k\boldsymbol{p}_i;$$

Fim

Algoritmo 3.2: Fase de execução do SOAC

Dados: $SOAC = \{^{p}\boldsymbol{w}^{k}, ^{c}\boldsymbol{w}^{k}\}, (k = 1,..,K)$ mapa de módulos preditor/controlador com os preditores previamente treinados e $P = \{\tilde{\boldsymbol{p}}^{\boldsymbol{k}}\}, (k = 1,..,K)$ mapa de parâmetros estimados

Entrada: $\boldsymbol{x}(t)$,$^{cfc}\boldsymbol{u}(t)$ estado atual da planta e comando de controle do cfc

Saída: $\boldsymbol{u}(t)$ comando de controle total

Início

Calcular o error de predição;

$$^{p}e^{k}(t) = (1-\varepsilon) * {}^{p}e^{k}(t-\Delta t) + \varepsilon * \left\| \boldsymbol{x}(t) - \widetilde{\boldsymbol{x}}^{k}(t) \right\|;$$

Determinar o BMM que minimiza o error de predição;

$$k^{*} = \underset{k}{agr\,min}\ {}^{p}e^{k};$$

Calcular o sinal de responsabilidade;

$$\phi^{k} = \frac{exp[\frac{-\left\| \boldsymbol{\varepsilon}^{k} - \boldsymbol{\varepsilon}^{k^{*}} \right\|^{2}}{2\sigma_{\infty}^{2}}]}{\sum_{k'} exp[\frac{-\left\| \boldsymbol{\varepsilon}^{k'} - \boldsymbol{\varepsilon}^{k^{*}} \right\|^{2}}{2\sigma_{\infty}^{2}}]};$$

Gerar o comando de controle total fornecido para o objeto controlado;

$$^{cfc}\boldsymbol{u}(t) = {}^{cfc}\boldsymbol{W}(\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t));$$

$$\boldsymbol{u}^{k}(t) = {}^{c}f^{k}({}^{c}\boldsymbol{w}^{k}, \boldsymbol{x}(t), \hat{\boldsymbol{x}}(t));$$

$$\boldsymbol{u}(t) = \sum_{k=1}^{K} \phi^{k}\boldsymbol{u}^{k}(t) + {}^{cfc}\boldsymbol{u}(t);$$

Atualizar o peso do controle;

$$\Delta^{c}\boldsymbol{w}^{k} = \eta \cdot \phi^{k} \frac{\partial\, {}^{c}f^{k}}{\partial\, {}^{c}\boldsymbol{w}^{k}} \cdot {}^{cfc}\boldsymbol{u}(t);$$

Fim

3.4 Discussão Final

O *Self-Organizing Adaptive Controller* (controlador adaptativo auto-organizável) tenta prover duas características bastante importantes para controles modernos. Uma é ser capaz de controlar adequadamente mesmo que o objeto controlado mude seus parâmetros internos ou que haja mudanças dos parâmetros do ambiente. A outra propriedade é ser apto a controlar objetos com parâmetros não conhecidos previamente. A primeira é atendida através de uma modelagem de controles de múltiplos módulos. Nesse ponto, o SOAC foi inspirado nos Múltiplos Modelos de Pares Direto/Inverso (MPFIM) WOLPERT; KAWATO (1998). Já a segunda é provida pelo poder de generalização intrínseco a rede do tipo SOM. Mais precisamente, o SOAC é projetado sobre uma rede modular SOM (mnSOM).

A concepção do SOAC parece ser bastante promissor. Trabalhos recentes em controles inteligentes vêm utilizando múltiplos módulos para o projeto do controlador, abordagem esta compatível com o emprego do SOM. Porém, a estrutura topológica de módulos do SOAC é fixa. Consequentemente, o número de módulos de controle é fixo e a organização topológica dos controles também não pode ser dinamicamente explorada. Poderia ser explorada uma versão de controle de múltiplos módulos inspirado em uma variante do SOM com topologia variante no tempo. Outra limitação do SOAC, é a difícil compreensão do comportamento e influência dos sinais de responsabilidades na etapa de aprendizagem dos preditores e dos controles.

4

Controles Auto-Organizáveis Desenvolvidos

Neste capítulo, serão apresentadas duas proposições de métodos para controle de movimentos. A primeira solução é baseada no emprego de uma rede SOM para controle e foi utilizado o esquema módulo preditor/controle. Já a segunda, é fundamentada sobre uma variante da rede SOM chamada LARFSOM. Essa solução foi projetada usando o esquema de modelagem inversa direta.

Em um primeiro momento, serão descritas as redes neurais utilizadas em cada solução. Logo após, serão apresentadas as duas soluções. Por fim, é feita uma discussão sobre os métodos propostos.

4.1 Modelos de Redes Neurais

Nessa seção, é explicada, brevemente, cada rede neural usada nas duas soluções. Primeiramente, é apresentado o Mapa Auto-organizável (SOM) original. Por fim, é descrito um outro tipo de Mapa Auto-organizável, o LARFSOM.

4.1.1 SOM - Mapa Auto-Organizável

O Mapa Auto-organizável (*Self-organizing Map* - SOM) também chamado de Mapa Auto-organizável de Característica (*Self-organizing Feature Map* - SOFM) foi desenvolvido por Teuvo Kohonen na década de 80 KOHONEN (1982) e tem sido usado com sucesso para resolver um grande número de aplicações em reconhecimento de padrões. O Mapa Auto-organizável de Característica de Kohonen é um modelo de rede neural competitiva com a propriedade de reduzir padrões de alta dimensionalidade em um mapa de dimensão inferior. Em geral, o mapa é constituído por uma grade uni ou bidimensional de nodos (Figura 4.1). O mapa topológico é automaticamente organizado através de um processo cíclico de comparação dos padrões de entrada com o vetor peso de cada nodo e ajuste do nodo mais semelhante e de sua vizinhança. Dessa forma, o algoritmo de aprendizagem organiza os nodos em uma malha onde cada região desta malha atua como um “categorizador de características” dos dados.

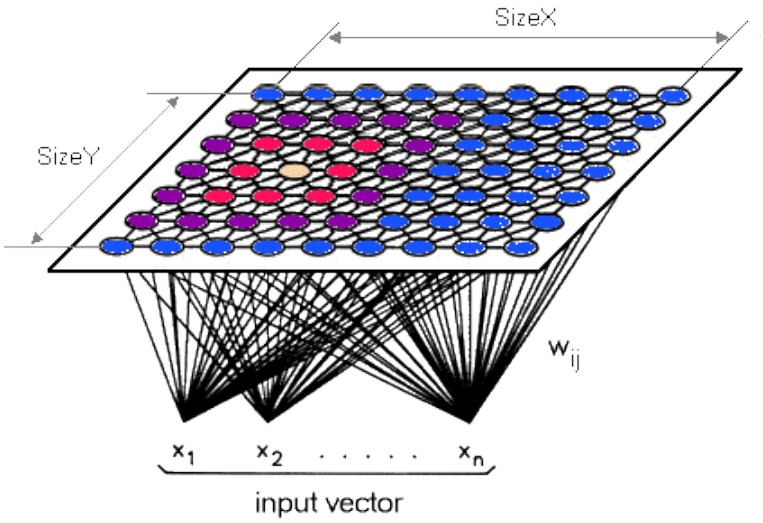

Figura 4.1: Exemplo de topologia usada para SOM. As unidades de entrada são conectadas a todos os nodos da camada de saída.

O SOM funciona basicamente da seguinte forma: quando um padrão de entrada p é apresentado, a rede procura o *Módulo com Melhor Correspondência* (*Best Matching Module* - BMM) com p, isto é, o nodo cujo vetor de pesos a ele associado é o mais próximo do padrão de entrada segundo uma medida de distância. Esse nodo também é chamando de *vencedor*. Durante o seu treinamento, a rede aumenta a semelhança do nodo vencedor e de seus vizinhos ao padrão p através de uma regra de ajuste que está especificada na Equação 4.1. O processo termina quando as modificações no mapa são insignificantes. Dessa maneira, a rede constrói um mapa topológico onde nodos que estão topologicamente próximos respondem de forma semelhante a padrões de entrada parecidos.

A regra de ajuste dos pesos tem a seguinte forma:

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \alpha(t) h_{bmm,k}(t) \left[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)\right] \tag{4.1}$$

no qual $\boldsymbol{w}_k(t)$ é o vetor de pesos da unidade k no instante t. Normalmente, o valor inicial $\boldsymbol{w}_k(0)$ é um valor aleatório. $\boldsymbol{x}(t)$ é o vetor de entrada apresentado no instante t. $\alpha(t)$ é a taxa de aprendizagem representada por uma função que decresce monotonicamente à medida que t cresce. Outro ponto importante é que $0 < \alpha(t) < 1$. $h_{bmm,k}(t)$ é a função vizinhança cujo valor depende da unidade vencedora (BMM), do nodo atual k e do instante t. É, também, uma função com valor máximo unitário e mínimo nulo e decrescente à medida que se distancia do nó vencedor. Para assegurar a convergência $h_{bmm,k}(t) \to 0$ quando $t \to \infty$. Normalmente, é escolhida uma função gaussiana para representar $h_{bmm,k}(t)$:

$$h_{bmm,k}(t) = exp(-\frac{\|\boldsymbol{r}_k(t) - \boldsymbol{r}_{bmm}(t)\|^2}{2\sigma^2(t)} \tag{4.2}$$

no qual $\boldsymbol{r}_k(t)$ e $\boldsymbol{r}_{bmm}(t)$ são as coordenadas dos nodos k e do BMM, respectivamente. $\sigma(t)$ define

a região de vizinhança. Assim como $\alpha(t)$, $\sigma \to 0$ quando $t \to \infty$. Foram escolhidas as seguintes Funções 4.3 e 4.4 para representar $\alpha(t)$ e $\sigma(t)$, respectivamente.

$$\alpha(t) = \alpha_\infty + (\alpha_0 - \alpha_\infty) exp(-\frac{t}{\tau_\alpha}) \tag{4.3}$$

$$\sigma(t) = \sigma_\infty + (\sigma_0 - \sigma_\infty) exp(-\frac{t}{\tau_\sigma}) \tag{4.4}$$

É apresentado em Pseudocódigo 4.1, o procedimento de treinamento desta rede neural.

Algoritmo 4.1: Treinamento da rede SOM

Dados: $S_{treinamento} = \{\boldsymbol{x}_i\}, (i = 1, ..., I)$ conjunto de treinamento

Resultado: $SOM = \{\boldsymbol{w}_k\}, (k = 1, .., K)$ mapa de nodos

Início

```
//Construção do mapa inicial;
Mapa ← Inicialização();
//Inicializar pesos;
para w_k ∈ Mapa faça
    w_k ← valores aleatórios entre [0,499; 0,501]
fim
n = 0;
para n < N faça
    para cada x_i em S_treinamento faça
        //Definir nodo vencedor;
        Selecionar o nodo mais próximo ao padrão de treinamento;
        Atualizar os pesos deste nodo e de seus vizinhos;
        se n é múltiplo de y então
            reduzir taxa de aprendizado e área de vizinhança;
        fim
    fim
fim
```

Fim

4.1.2 LARFSOM - Mapa Auto-Organizável com Campo Receptivo Adaptativo Local

O Mapa Auto-organizável com Campo Receptivo Adaptativo Local (LARFSOM), diferentemente da versão original do SOM, tem uma estrutura variante no tempo ARAúJO; RÊGO (2013). O LARFSOM inicia com um mapa de dois nodos e, durante o treinamento, insere

novos nodos e novas conexões entre eles no mapa. A regra de inserção de nodos é baseada na similaridade. Se o mapa possui algum vetor de pesos associado a um nodo suficientemente semelhante ao padrão de treinamento apresentado, esse mapa não necessita de um novo protótipo para representar esse padrão. Mas, caso o mapa não possua um nodo com vetor de pesos suficientemente semelhante ao padrão de treinamento apresentado, esse mapa necessita de um novo protótipo para representar adequadamente tal padrão. Um problema que surge é *como definir similaridade entre o padrão de entrada e os protótipos do mapa*. Para resolver isso, o LARFSOM usa "uma similaridade determinada através de um limiar de ativação, calculado com respeito a um valor de campo receptivo de vizinhança local"ARAúJO; COSTA (2009).

O LARFSOM funciona da seguinte forma: quando um padrão de entrada p é apresentado, a rede procura os dois nodos com melhores correspondências com p, isto é, os dois nodos mais próximos do padrão de entrada segundo uma medida de distância. Logo em seguida, a rede calcula o grau de ativação do nodo vencedor para saber se será necessário se inserir um novo nodo ou atualizará os pesos do vencedor. Esse processo de treinamento se encerra quando a rede converge e é determinado por um critério de convergência. Por fim, são filtrados todos os nodos desconectados.

Mais detalhadamente, a Figura 4.2 apresenta o diagrama de fluxo com subprocedimentos utilizados na aprendizagem. Detalhes do procedimento de treinamento do LARFSOM serão apresentados a seguir.

O primeiro passo é estabelecer os valores iniciais dos seguintes parâmetros:

- Taxa de aprendizagem final (ρ_f)
- Modulador da taxa de aprendizagem (ε)
- Limiar de atividade (a_T)
- Quantidade de vitórias do nodo i (d_i)
- Máxima quantidade de vitórias de um nodo (d_m)
- Erro mínimo (e_m)
- Número inicial de nodos já conectados ($N = 2$)

Os nodos iniciais são gerados com valores de pesos aleatórios.

Logo em seguida, é selecionado, aleatoriamente, um padrão de entrada ($\boldsymbol{\xi}$) e são determinados os dois nodos mais próximos ao vetor de entrada.

$$d(\boldsymbol{w}_{s_1}, \boldsymbol{\xi}) \leq d(\boldsymbol{w}_{s_2}, \boldsymbol{\xi}) \leq d(\boldsymbol{w}_i, \boldsymbol{\xi}), \forall i \in N \tag{4.5}$$

onde $d(\boldsymbol{w}, \boldsymbol{\xi}) = \|\boldsymbol{w} - \boldsymbol{\xi}\|^2$ e N é o conjunto de todos os nodos. O nodo mais próximo é identificado por s_1 e o segundo, por s_2. Também, é incrementado o contador de vitórias do vencedor s_1:

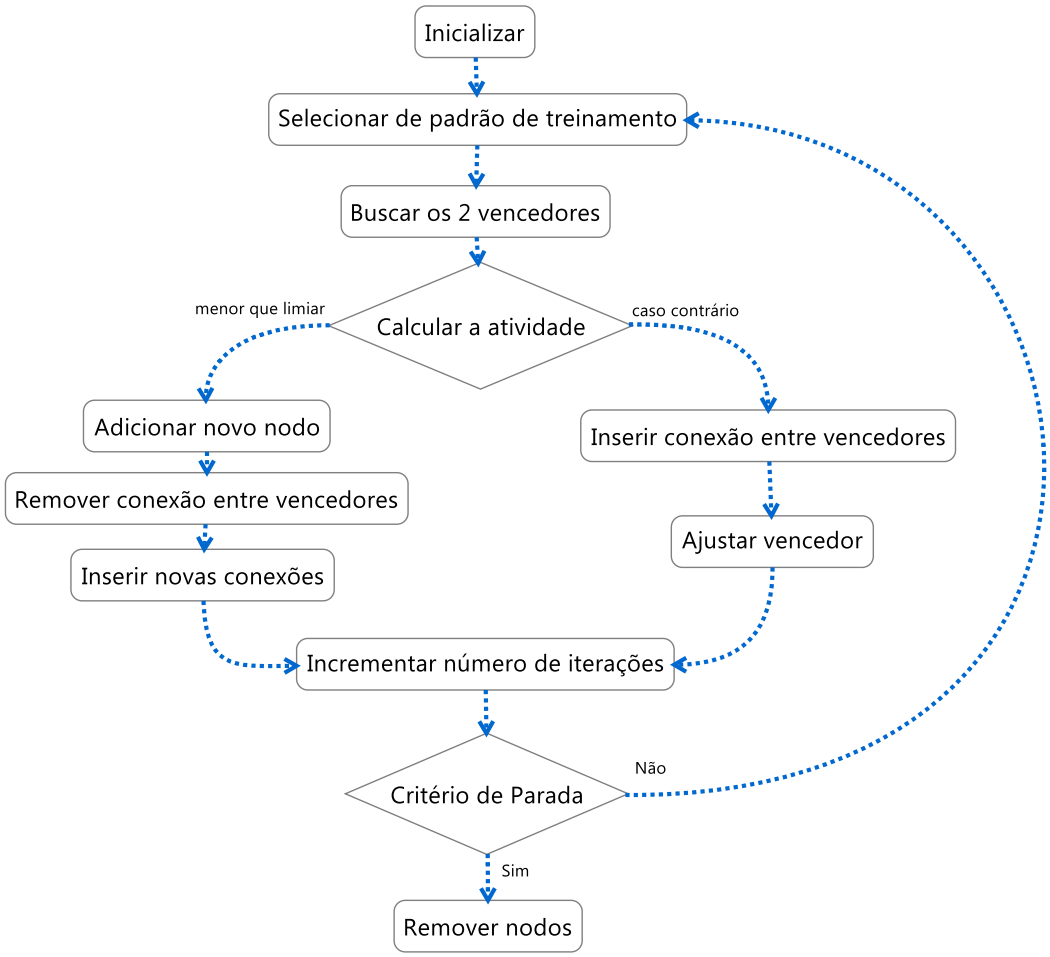

Figura 4.2: Diagrama de fluxo do procedimento de aprendizagem no LARFSOM.

$$d_{s_1} = d_{s_1} + 1 \tag{4.6}$$

Após encontrar os dois vencedores, é calculado o nível de atividade do vencedor s_1. Primeiramente, é dimensionado o campo receptivo de s_1:

$$r_{s_1} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{w}_{s_1}[i] - \boldsymbol{w}_{s_2}[i])^2} \tag{4.7}$$

onde n é a dimensão do vetor de entrada. Em seguida, é calculada a atividade de s_1 por meio de uma *função de base radial* (Eq. 4.8).

$$a_{s_1} = \frac{exp(-\|\boldsymbol{\xi} - \boldsymbol{w}_{s_1}\|)}{r_{s_1}} \tag{4.8}$$

Se $a_{s_1} < a_T$, então os seguinte passos são realizados:

- Adicionar um novo nodo com o vetor de pesos $\boldsymbol{w}_n = \boldsymbol{\xi}$
- Remover a conexão entre os vencedores s_1 e s_2 (caso exista)

- Calcular as distâncias: $d(\boldsymbol{w}_{s_1}, \boldsymbol{w}_n), d(\boldsymbol{w}_{s_2}, \boldsymbol{w}_n)$ e $d(\boldsymbol{w}_{s_1}, \boldsymbol{w}_{s_2})$
- Inserir as duas conexões com menores distâncias

Se $a_{s_1} \geq a_T$, então uma conexão entre os vencedores s_1 e s_2 é inserida (caso não exista) e é atualizado o nodo vencedor s_1:

$$\Delta \boldsymbol{w}_{s_1} = \rho \times (\boldsymbol{\xi} - \boldsymbol{w}_{s_1}) \tag{4.9}$$

onde,

$$\rho = \begin{cases} \varepsilon \times \rho_f^{\frac{d_i}{d_m}}, & d_i \leq d_m \\ \varepsilon \times \rho_f, & d_i > d_m \end{cases} \tag{4.10}$$

Lembrando que d_i é a quantidade de vitórias do nodo i e d_m é a máxima quantidade de vitórias de um nodo. Após, é incrementado o número de iterações e é verificado o critério de parada. O critério de parada é definido pela Equação 4.11 ou por atingir o número máximo de iterações.

$$e = \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{w}_i(t) - \boldsymbol{w}_i(t+1)\|^2 \leq e_m = 10^{-4} \tag{4.11}$$

O valor de $e_m = 10^{-4}$ é sugerido pelo o artigo original do LARFSOM ARAúJO; COSTA (2009). Caso não pare, retorna ao passo de selecionar padrão de entrada. Caso pare, é feito um procedimento para remover todos os nodos desconectados.

O procedimento de treinamento do LARFSOM é apresentado em Pseudocódigo 4.2.

4.2 Controles Neurais

Nessa seção, são explicadas as duas soluções propostas de controle desenvolvidas nesse trabalho. Primeiramente, é mostrado, em detalhes, um sistema de controle inteligente utilizando o SOM. Nessa primeira solução, é usado o esquema de módulos preditor/controlador, semelhante ao SOAC. A principal diferença dessa solução para as outras é ser um controle baseado em um SOM mais próximo do convencional e utiliza a abordagem de aprendizagem por realimentação de erro para ajuste dos controladores.

Em seguida, é descrita a segunda proposta, que é um sistema de controle de estrutura topológica variante no tempo. Essa estrutura variante é a característica marcante que diferencia das outras soluções da literatura. O esquema utilizado para confecção desse segundo controle foi a modelagem inversa direta.

Ambos sistemas de controle foram projetados para atender a dois requisitos importantes para sistema em ambientes como o ambiente humano. São eles: (1) uma rápida resposta a mudanças inesperadas no sistema controlado; (2) e a aquisição de comportamento geral a partir de uma amostra pequena de exemplos. O primeiro requisito é atendido nos dois sistemas de

Algoritmo 4.2: Treinamento da rede LARFSOM

Dados: $S_{treinamento} = \{\boldsymbol{x}_i\}, (i = 1, ..., I)$ conjunto de treinamento
Resultado: $LARFSOM = \{\boldsymbol{w}_k\}, (k = 1, .., K)$ mapa de nodos
Início
  $\rho_f \leftarrow$ taxa de aprendizagem final; $\varepsilon \leftarrow$ modulador da taxa de aprendizagem;
  $a_T \leftarrow$ limiar de corte de ativação; $d_m \leftarrow$ número máximo de vitórias;
  $e_{min} \leftarrow$ erro mínimo;
  $Mapa \leftarrow$ Inicialização(); *//mapa com 2 nodos e conectados*;
  para $w_k \in Mapa$ **faça**
    $w_k \leftarrow$ valores aleatórios entre $[0,499; 0,501]$; *//Inicializar pesos*;
    $d_k \leftarrow 0$; *//Número de vitórias*;
  fim
  $n = 0$;
  $e = \infty$;
  para $e \leq e_{min}$ || $n < N$ **faça**
    para cada $\boldsymbol{x}_i$ *em* $S_{treinamento}$ **faça**
      $w_{s_1} \leftarrow$ Selecionar o nodo mais próximo ao padrão de treinamento;
      $w_{s_2} \leftarrow$ Selecionar o 2º nodo mais próximo ao padrão de treinamento;
      $d_{s_1} = d_{s_1} + 1$; *//Incrementar contador de vitórias do vencedor*;
      Inserir uma conexão entre s_1 e s_2 se ela não existir;
      $l \leftarrow$ comprimento do vetor $\boldsymbol{w}$;
      $r_{s_1} = \sqrt{\sum_{i=1}^{l} (\boldsymbol{w}_{s_1}[i] - \boldsymbol{w}_{s_2}[i])^2}$;
      //Calcular o campo receptivo de s_1;
      $a_{s_1} = \dfrac{exp(-\|\boldsymbol{x}_i - \boldsymbol{w}_{s_1}\|)}{r_{s_1}}$; *//Calcular a atividade de* s_1;
      //Inserir um nodo se a ativação do vencedor estiver abaixo do limiar;
      se $a_{s_1} < a_T$ **então**
        $\boldsymbol{w}_{novo} \leftarrow \boldsymbol{x}_i$; *//Inserir novo nodo*;
        Remover conexões entre s_1 e s_2;
        Calcular distâncias entre *novo*, s_1 e s_2 Inserir as duas conexões com menores distâncias;
      fim
      senão
        $\rho = \begin{cases} \varepsilon \times \rho_f^{\frac{d_i}{d_m}}, & d_i \leq d_m \\ \varepsilon \times \rho_f, & d_i > d_m \end{cases}$; *//Caso contrário, atualizar os pesos do vencedor*;
        $\Delta \boldsymbol{w}_{s_1} = \rho \times (\boldsymbol{x}_i - \boldsymbol{w}_{s_1}(t))$;
        $\boldsymbol{w}_{s_1}(t+1) = \boldsymbol{w}_{s_1}(t) + \Delta \boldsymbol{w}_{s_1}$;
      fim
    fim
    $e = \frac{1}{K} \sum_{i=1}^{K} \|\boldsymbol{w}_i(t) - \boldsymbol{w}_i(t+1)\|^2$;
  fim
Fim
Remover os nodos desconectados;

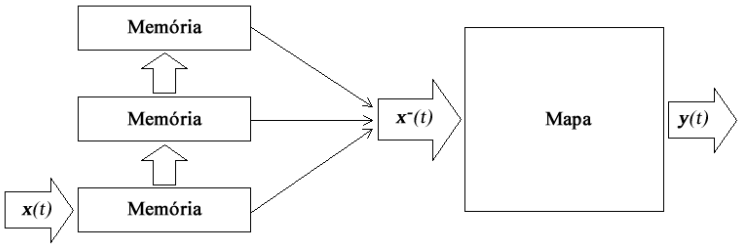

Figura 4.3: O Mapa Auto-organizável treinado com padrões de entrada concatenados. No exemplo, três padrões de entrada são concatenados para forma o vetor de entrada da rede. Adaptado da Figura 2 de KANGAS (1990).

controle através da modelagem de modelos locais. As duas soluções conseguem satisfazer o segundo requisito através da capacidade de generalização do SOM e do LARFSOM. Na segunda proposta, o segundo requisito também é atendido por causa do uso do algoritmo genético.

4.2.1 Controle SOM

Para proposta de controle utilizando o SOM, foi usado o mesmo esquema do controle SOAC, o modelo preditor/controle. Para isso, o vetor de peso $\boldsymbol{w}$ de cada nodo é dividido em duas partes $\boldsymbol{w}^{in}$ e $\boldsymbol{w}^{out}$. A parte $\boldsymbol{w}^{in}$ é responsável pela identificação da configuração atual da planta. Enquanto $\boldsymbol{w}^{out}$ é responsável por gerar um comando de controle adequado. Essa modelagem é baseada na Memória Associativa Temporal por Quantização Vetorial (*vector-quantized temporal associative memory* - VQTAM) BARRETO; ARAúJO (2001, 2004).

Para identificar a planta, o SOM tem que aprender os diferentes comportamentos do sistema. Isso é feito diferentemente do SOAC. No controle SOM, a identificação é feita sob a ótica de modelos locais, em que o espaço de entrada é particionado em regiões de operações menores sobre as quais são construídos modelos de menor complexidade. Isto é, para uma determinada configuração de planta aprendida pela rede SOM, há vários nodos representando as diferentes regiões de operações dessa configuração. No SOAC, existe uma única função em cada módulo, que modela o comportamento do sistema dinâmico para uma determinada configuração. Logo, a divisão do espaço de possibilidades, no SOAC, é feita particionando possíveis configurações de planta. Enquanto no SOM, é feita a divisão de possíveis configurações de planta e regiões de operações.

Na ótica de dividir os espaço de entrada, a entrada real da rede é construída a partir de uma concatenação dos vetores de entrada (Figura 4.3). Essa concatenação dos padrões de entrada é chamada de janela de tempo. Essa janela contém uma sequência de padrões associados temporalmente. O tamanho da janela p é definido previamente. No problema de controle, pode-se definir, em termos formais, da seguinte maneira:

$$\boldsymbol{x}_{in}^{-}(t) = [\boldsymbol{x}(t), u(t), \boldsymbol{x}(t-1), u(t-1), \cdots, \boldsymbol{x}(t-p+1), u(t-p+1)] \qquad (4.12)$$

onde $[\boldsymbol{x}(t), u(t)]$ é o vetor característico para analisar o comportamento da planta sendo $\boldsymbol{x}(t)$ o estado para planta e $u(t)$ o comando de controle aplicado a planta. p é o tamanho da janela de concatenação.

No treinamento, tem-se, associado a cada $\boldsymbol{x}_{in}^{-}(t)$, um vetor $\boldsymbol{x}_{out}(t)$, que corresponde aos parâmetros da planta $\boldsymbol{p}$ que gerou o comportamento $\boldsymbol{x}_{in}^{-}(t)$. O padrão de treinamento $\boldsymbol{x}_{in}^{-}(t)$ é usado para ajustar a parte $\boldsymbol{w}^{in}$, enquanto $\boldsymbol{x}_{out}(t)$ é utilizado para adaptar a parte $\boldsymbol{w}^{out}$. Na fase de treinamento, a determinação do nodo vencedor, representado por k^*, emprega ou $\boldsymbol{x}_{in}^{-}$, detalhado na Equação 4.13, ou $\boldsymbol{x}_{out}$, detalhado na Equação 4.14. Para a etapa de execução, é usado somente a competição pelo $\boldsymbol{x}_{in}^{-}$, pois, normalmente, não há a informação da configuração da planta. Formalmente, tem-se:

$$k^*(t) = \underset{k}{arg\,min} \left\| \boldsymbol{x}_{in}^{-}(t) - \boldsymbol{w}_k^{in}(t) \right\| \tag{4.13}$$

ou

$$k^*(t) = \underset{k}{arg\,min} \left\| \boldsymbol{x}_{out}(t) - \boldsymbol{w}_k^{out}(t) \right\| \tag{4.14}$$

Então, os ajustes no nodo k é feito da seguinte forma:

$$\boldsymbol{w}_k^{in}(t+1) = \boldsymbol{w}_k^{in}(t) + \alpha(t) h_{bmm,k}(t) \left[\boldsymbol{x}_{in}(t) - \boldsymbol{w}_k^{in}(t) \right] \tag{4.15}$$

$$\boldsymbol{w}_k^{out}(t+1) = \boldsymbol{w}_k^{out}(t) + \alpha(t) h_{bmm,k}(t) \left[\boldsymbol{x}_{out}(t) - \boldsymbol{w}_k^{out}(t) \right] \tag{4.16}$$

Na etapa de execução, essa proposta de controle SOM é bastante semelhante ao SOAC. É utilizada, também, a aprendizagem por realimentação de erro para o treinamento do controle. Novamente, a aprendizagem aqui é *incremental*. Para cada nodo k, há um vetor de pesos ${}^{c}\boldsymbol{w}^{k}$, que define uma função ${}^{c}\boldsymbol{f}^{k}$ (Equação 4.17).

$$\boldsymbol{u}^{k}(t) = {}^{c}\boldsymbol{f}^{k}({}^{c}\boldsymbol{w}^{k}, \hat{\boldsymbol{x}}(t)) \tag{4.17}$$

onde $\hat{\boldsymbol{x}}(t)$ é o vetor do estado desejado para a planta.

O comando de controle final é resultado da soma da saída do controle retroalimentado convencional (cfc) com o comando de controle gerado pelo nodo vencedor. Como dito anteriormente, o vencedor é determinado pela Equação 4.13.

$$\boldsymbol{u}(t) = \boldsymbol{u}^{k^*}(t) + \boldsymbol{u}^{cfc}(t) \tag{4.18}$$

A maneira de ajuste para cada ${}^{c}\boldsymbol{w}^{k}$ é similar ao SOAC. É empregada a mesma regra de ajuste descrita na Equação 3.17. Na Figura 4.4, é apresentado um nodo da rede SOM com um bloco representado a identificação do sistema (*bloco do preditor*) e um outro bloco representado o controlador (*bloco do controlador*). No bloco do preditor, é exemplificado uma janela de tempo de tamanho $p = 2$.

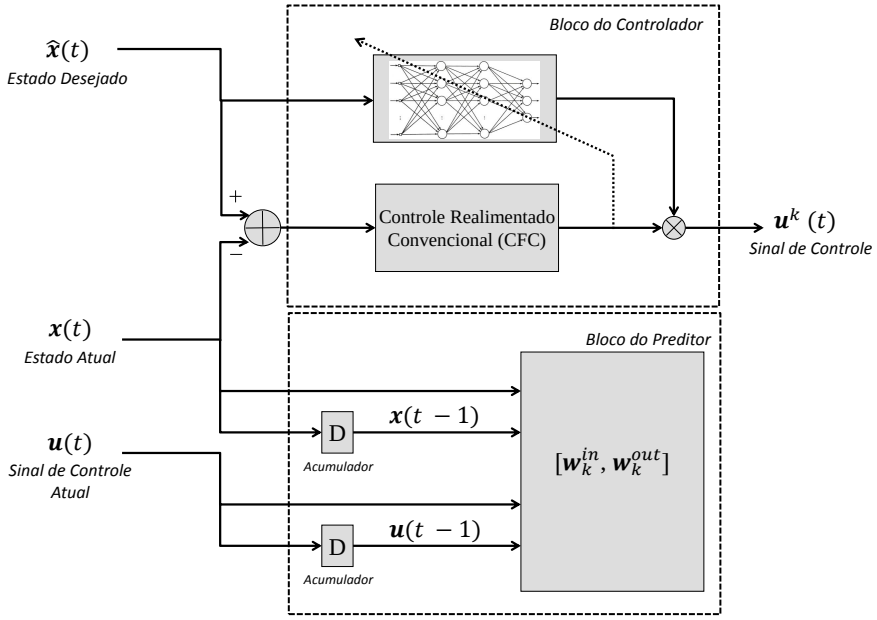

Figura 4.4: Nodo do SOM modificado para problema do controle com aprendizagem por realimentação de erro.

4.2.2 Controle LARFSOM

Nessa segunda proposta, tem-se uma solução de controle no esquema de modelagem inversa direta, um candidato natural quando se usa redes neurais. Na Figura 2.6, tem-se a abordagem computacional para aprender o modelo inverso do objeto controlado. Nessa abordagem, o objeto controlado sobre o estado $\boldsymbol{x}(t)$ recebe um comando de força $u(t)$ e responde com a trajetória resultante $y(t)$. Para sistema observáveis, é possível obter o vetor de estado $\boldsymbol{x}(t)$ em qualquer instante de tempo t a partir da informação de saída do sistema $y(t)$ e do estado inicial $\boldsymbol{x}(0)$.

Estrutura do Controle

Como foi dito anteriormente, esse segundo controle é confeccionado sobre uma adaptação da rede LARFSOM, caracterizada por um fracionamento nos vetores de pesos. O $\boldsymbol{w}$ é dividido em duas partes, $\boldsymbol{w}_c$ e $\boldsymbol{w}_p$. O vetor $\boldsymbol{w}_c$ consiste de um vetor com o seguinte formato: $[\boldsymbol{x}_w(t), u_w(t), \tilde{\boldsymbol{x}}_w(t+\Delta t)]$, onde $\boldsymbol{x}_w$ é o vetor de pesos correspondente ao estado atual, $u_w(t)$ é o comando de controle aplicado e $\tilde{\boldsymbol{x}}_w(t+\Delta t)$ é a estimativa de próximo estado após ser aplicado $u_w(t)$. Já o vetor $\boldsymbol{w}_p$ consiste dos parâmetros estimados do sistema dinâmico, considerado para gerar os estados do sistema. Por exemplo, no experimento do sistema massa-mola-amortecedor, $\boldsymbol{w}_p = [b,k]$, onde b é o coeficiente de amortecimento e k é a constante da mola. Pode haver diferentes configurações do sistema determinadas por valores distintos de b e k.

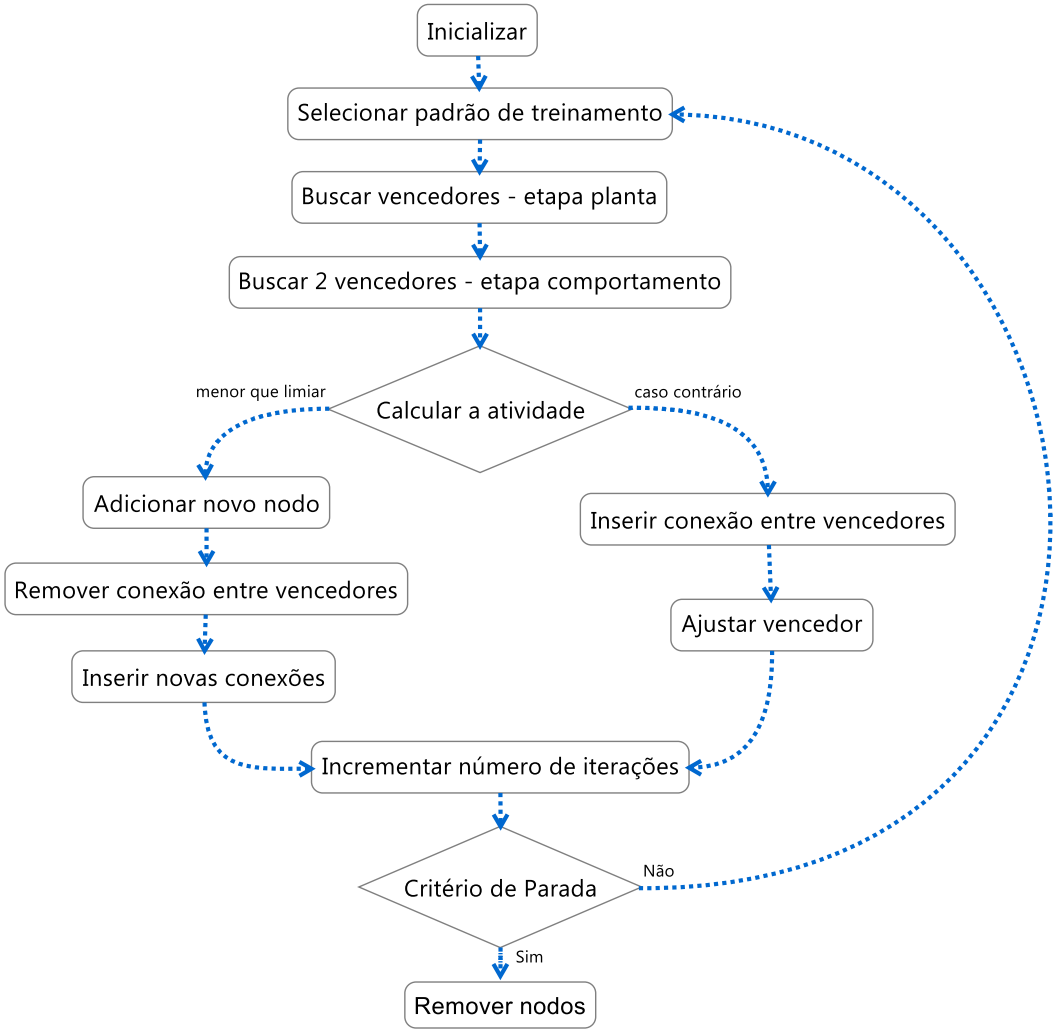

Figura 4.5: Diagrama de fluxo do procedimento de aprendizagem no LARFSOM adaptado para controle.

Procedimento de Aprendizagem do Controle

O procedimento de treinamento dos pesos da rede também é semelhante ao do LARFSOM original. A principal mudança está na procura dos nodos com melhores correspondências. Como o vetor de peso é divido em duas partes, a busca pelos nodos vencedores também foi alterada. Essa busca é dividida em duas etapas. Em um primeiro momento, é feita uma busca dos $\boldsymbol{w}_p$ que melhor correspondem às componentes dos parâmetros do sistema do padrão de entrada. Pode haver mais de um nodo com $\boldsymbol{w}_p$ iguais e com $\boldsymbol{w}_c$ distintos. Dentre esse conjunto de $\boldsymbol{w}_c$ selecionado na primeira busca, é feita a segunda busca. Essa segunda busca é igual ao do LARFSOM.

A Figura 4.5 representa o diagrama de fluxo com subprocedimentos utilizados na aprendizagem. Agora, é explicado cada subprocedimento em detalhes.

O primeiro passo é estabelecer os valores iniciais dos seguintes parâmetros:

- Taxa de aprendizagem final (ρ_f)

- Modulador da taxa de aprendizagem (ε)
- Limiar de atividade (a_T)
- Quantidade de vitórias do nodo i (d_i)
- Máxima quantidade de vitórias de um nodo (d_m)

São inicializados 2 nodos para cada configuração dos parâmetros do sistema usados no treinamento. Isto é, insere-se 2 nodos com a parte $\boldsymbol{w}_p$ iguais para cada configuração do sistema usado no treinamento. Já a parte $\boldsymbol{w}_c$ são preenchidos com valores de pesos aleatórios.

Logo em seguida, é selecionado aleatoriamente um padrão de entrada ($\boldsymbol{\xi} = [\boldsymbol{\xi}_c, \boldsymbol{\xi}_p]$) onde $\boldsymbol{\xi}_c$ corresponde a uma amostra de comportamento da planta $[\boldsymbol{x}_t(t), u_t(t), \boldsymbol{x}_t(t+\Delta t)]$ e $\boldsymbol{\xi}_p$, aos parâmetros da planta. Na primeira busca, são selecionados todos os nodos com a parte $\boldsymbol{w}_p$ igual a $\boldsymbol{\xi}_p$. A partir desse conjunto, são determinados os dois nodos mais próximos ao vetor $\boldsymbol{\xi}_c$.

$$d(\boldsymbol{w}_{c_{s_1}}, \boldsymbol{\xi}_c) \leq d(\boldsymbol{w}_{c_{s_2}}, \boldsymbol{\xi}_c) \leq d(\boldsymbol{w}_{c_i}, \boldsymbol{\xi}_c), \forall i \in N_{\boldsymbol{\xi}_p} \tag{4.19}$$

onde $d(\boldsymbol{w}_c, \boldsymbol{\xi}_c) = \|\boldsymbol{w}_c - \boldsymbol{\xi}_c\|^2$ e $N_{\boldsymbol{\xi}_p}$ é conjunto de todos os nodos com $\boldsymbol{w}_p = \boldsymbol{\xi}_p$. O nodo mais próximo é identificado por s_1 e o segundo, por s_2. Também, é incrementado o contador de vitórias do vencedor s_1: $d_{s_1} = d_{s_1} + 1$.

Após encontrar os dois vencedores, é calculado o nível de atividade do vencedor s_1. Primeiro, é dimensionado o campo receptivo de s_1:

$$r_{s_1} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{w}_{s_1}[i] - \boldsymbol{w}_{s_2}[i])^2} \tag{4.20}$$

onde n é a dimensão do vetor de entrada. Em seguida, é calculada a atividade de s_1 por meio de uma *função de base radial*:

$$a_{s_1} = \frac{exp(-\|\boldsymbol{\xi} - \boldsymbol{w}_{s_1}\|)}{r_{s_1}} \tag{4.21}$$

Se $a_{s_1} < a_T$, então os seguinte passos são realizados:

- Adicionar um novo nodo com o vetor de pesos $\boldsymbol{w}_n = \boldsymbol{\xi}$
- Remover a conexão entre os vencedores s_1 e s_2 (caso exista)
- Calcular as distâncias: $d(\boldsymbol{w}_{s_1}, \boldsymbol{w}_n), d(\boldsymbol{w}_{s_2}, \boldsymbol{w}_n)$ e $d(\boldsymbol{w}_{s_1}, \boldsymbol{w}_{s_2})$
- Inserir as duas conexões com menores distâncias

Se $a_{s_1} \geq a_T$, então uma conexão entre os vencedores s_1 e s_2 é inserida (caso não exista) e é atualizado o nodo vencedor s_1 pela seguinte equação:

$$\Delta \boldsymbol{w}_{s_1} = \rho \times (\boldsymbol{\xi} - \boldsymbol{w}_{s_1}) \tag{4.22}$$

onde,

$$\rho = \begin{cases} \varepsilon \times \rho_f^{\frac{d_i}{d_m}}, & d_i \leq d_m \\ \varepsilon \times \rho_f, & d_i > d_m \end{cases} \tag{4.23}$$

Após, é incrementado o número de iterações e é verificado o critério de parada. O critério de parada é atingir o número máximo de iterações. Caso não pare, retorna ao passo de selecionar padrão de entrada. Caso pare, é feito um procedimento para remover todos os nodos desconectados.

Procedimento de Execução do Controle

Na etapa de execução, o vetor de pesos de cada nodo $\boldsymbol{w}$ é constituído apenas da componente $\boldsymbol{w}_c$. A parte $\boldsymbol{w}_p$ é desprezada. Sendo assim, tem-se que $\boldsymbol{w} = \boldsymbol{w}_c = [\boldsymbol{x}_w(t), u_w(t), \tilde{\boldsymbol{x}}_w(t+\Delta t)]$.

Para um instante de tempo t, tem-se o estado atual $\boldsymbol{x}(t)$ e o próximo estado desejado $\boldsymbol{x}_d(t+\Delta t)$. Para melhor entendimento, é representado por $\boldsymbol{\zeta} = [\boldsymbol{x}(t), \boldsymbol{x}_d(t+\Delta t)]$. Então, para gerar o comando de controle para esse instante de tempo, é feita uma busca do nodo que possui a maior similaridade de $\boldsymbol{x}_w(t)$ com $\boldsymbol{x}(t)$ e $\tilde{\boldsymbol{x}}_w(t+\Delta t)$ com $\boldsymbol{x}_d(t+\Delta t)$.

$$d_e(\boldsymbol{w}_{k^*}, \boldsymbol{\zeta}) \leq d_e(\boldsymbol{w}_i, \boldsymbol{\zeta}), \forall i \in N \tag{4.24}$$

onde $d_e(\boldsymbol{w}_i, \boldsymbol{\zeta}) = \|\boldsymbol{x}_{w_i}(t) - \boldsymbol{x}(t)\|^2 + \|\tilde{\boldsymbol{x}}_{w_i}(t+\Delta t) - \boldsymbol{x}_d(t+\Delta t)\|^2$ e N é conjunto de todos os nodos. O nodo mais próximo é identificado por k^*.

Por fim, o comando de controle aplicado a planta é dado por:

$$u = u_{w_{k^*}} - \boldsymbol{k}_a^T \cdot \boldsymbol{E}_a - \boldsymbol{k}_d^T \cdot \boldsymbol{E}_d \tag{4.25}$$

onde $u_{w_{k^*}}$ é comando de controle armazenado no nodo vencedor, $\boldsymbol{E}_a$ e $\boldsymbol{E}_d$ são o erros associados ao estado atual e desejado, respectivamente (Equações 4.26 e 4.27) e $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ são vetores transpostos ponderadores dos erros atual e desejado, respectivamente.

$$\boldsymbol{E}_a = \boldsymbol{x}_{w_{k^*}}(t) - \boldsymbol{x}(t) \tag{4.26}$$

$$\boldsymbol{E}_d = \tilde{\boldsymbol{x}}_{w_{k^*}}(t+\Delta t) - \boldsymbol{x}_d(t+\Delta t) \tag{4.27}$$

Ambos $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ são ajustados por um algoritmo genético descrito no Apêndice A. O algoritmo genético desenvolvido encontra uma solução aproximadamente ótima ou ótima para $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ através de um processo evolutivo. É utilizada a técnica *hold-out* como critério de parada desse processo evolutivo.

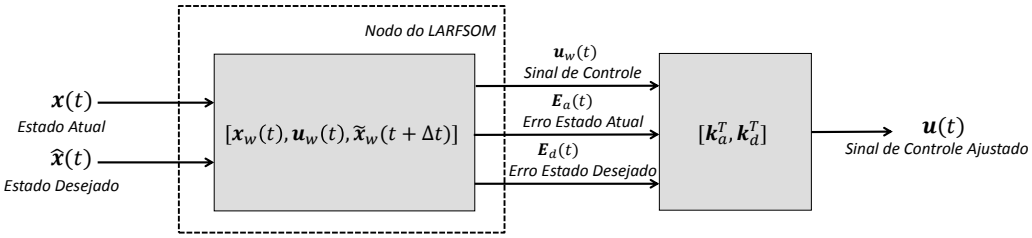

Figura 4.6: Nodo do LARFSOM modificado para problema do controle para modelagem inversa direta.

Na Figura 4.6, é apresentado um nodo da rede LARFSOM. Também é mostrado o bloco de ajuste do comando de controle do nodo.

4.3 Discussão Final e Contribuições

Um ponto importante a ser discutido é a existência de dois possíveis modos de competição do controle SOM na etapa de treinamento. Há duas formas de competição, pois a convencional do VQTAM (a Equação 4.13) nem sempre é conveniente de ser usada. A competição usando $\boldsymbol{w}^{in}$ tende a modelar mais a partição das regiões de operação e sintetizar as configurações de plantas aprendidas, isto é, haverá mais divisões das regiões de operações enquanto haverá uma tendência que vários nodos representem uma "média" das configurações de treinamento. Sendo assim, haverá uma menor variedade de configurações e isso poderá comprometer o requisito de adquirir um comportamento geral a partir de uma amostra pequena de exemplos. Já a segunda (a Equação 4.14) realiza justamente o oposto. A rede neural dará maior prioridade na divisão das possíveis configurações da planta enquanto a partição das regiões de interesse será mais grosseira.

Em resumo, a principal contribuição do modelo de controle SOM é a capacidade de particionar em regiões de operações menores além da partição de possíveis configurações de plantas. Isso pode gerar um maior número de nodos para uma determinada configuração de planta com cada nodo sendo especializado na sua região de operação. Já isso não ocorre no SOAC. Da maneira que o SOAC é projetado, haverá somente um módulo especializado para uma determinada configuração de planta. Essa característica do controle projetado com SOM pode proporcionar um poder de generalização maior que o do SOAC. Outro ponto de contribuição é que essa divisão de regiões de operação pode ser com uma prioridade na possíveis regiões de operação ou nas configurações observadas no treinamento. Isso dependerá da escolha da forma de competição na etapa de treinamento.

Já a contribuição mais relevante do modelo de controle LARFSOM é a possibilidade de um ajuste automático de partições de regiões de interesse através do crescimento adaptativo do número de nodo. Dessa maneira, o LARFSOM, além de apresentar a mesma propriedade do controle SOM que foi dissertado no paragrafo anterior, vai adaptar o número de regiões de

operação. A remoção de nodos também é importante para particionar adequadamente. Esse passo vai possibilitar a exclusão de regiões não significantes verificadas na fase treinamento.

Outra notória contribuição do controle LARFSOM é a regra de ajuste do comando de controle do nodo vencedor. Através dela e do ajuste adequado dos ponderadores $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ pelo algoritmo genético, é possível ter um poder de generalização maior do que a do controle SOM.

5

Resultados Experimentais - Sistema Mecânico Massa-Mola-Amortecedor

Este capítulo se destina à apresentar e discutir os resultados obtidos pelos modelos apresentados e propostos nesse trabalho. Para esse capítulo, os modelos são avaliados para o problema de controle do sistema massa-mola-amortecedor. O sistema massa-mola-amortecedor é um problema experimental de controle muito comum GU; PETKOV; KONSTANTINOV (2005) que pode fornecer uma visão preliminar da execução e análise dos controladores. Esse problema assume uma dinâmica de segunda ordem e apresenta uma natureza estável em malha aberta OGATA (2004).

As validações serão feitas, basicamente, através da capacidade de rastreamento de uma trajetória desejada da posição da massa. Para esse fim, são feitas análises de gráficos e do valor quadrático médio do erro (RMSE - *root mean square error*).

Este capítulo está dividido da seguinte forma: Primeiramente, é demonstrada a modelagem matemática do sistema massa-mola-amortecedor. Depois são apresentados os resultados obtidos pelos PDA, SOAC, SOM e LARFSOM. Por fim, é feita uma análise comparativa dos métodos experimentados.

5.1 Descrição do Experimento - Sistema Massa-Mola-Amortecedor

Nessa seção, é descrito em detalhes o experimento de controle do sistema massa-mola-amortecedor. Na Figura 5.1, há uma representação visual do sistema real. O sistema massa-mola-amortecedor é modelado matematicamente através de uma equação diferencial de segunda ordem, apresentada em Equação 5.1.

$$m\ddot{x}(t) + b_i\dot{x}(t) + k_i x(t) = F(t) \tag{5.1}$$

onde m, b_i e k_i são a massa do carrinho $[kg]$, o coeficiente de amortecimento $[\frac{kg}{s}]$ e a elasticidade da mola $[\frac{kg}{s^2}]$, respectivamente. Os sub-escritos indicam os valores para uma determinada configuração da planta. Nesse experimento, foram usadas 9 configurações da planta para

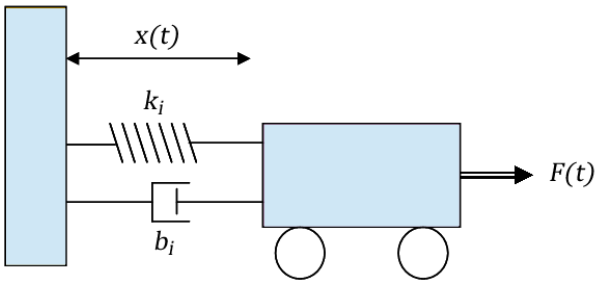

Figura 5.1: Sistema mecânico massa-mola-amortecedor.

treinamento e 6 para testar. Uma das configurações foi usada tanto para treinar quanto para testar. A massa do objeto é constante e igual a 1 kg. Enquanto que os outros dois parâmetros (b_i e k_i) podem assumir os seguintes valores $[2,6,10]$, gerando assim as 9 configurações usadas para treinar. Os valores dos parâmetros b_i e k_i usados para teste não foram explicitamente descritos no artigo original do SOAC MINATOHARA; FURUKAWA (2011), utilizado como base para esse trabalho de mestrado. Os valores aproximados das configurações de teste do trabalho original do SOAC estão apresentados na Tabela 5.1.

Tabela 5.1: Valores dos parâmetros b e k usado para testar

	b_i	k_i
$\boldsymbol{p}_A$	6	6
$\boldsymbol{p}_B$	6,1	6,8
$\boldsymbol{p}_C$	4,3	6,5
$\boldsymbol{p}_D$	5	4
$\boldsymbol{p}_E$	8,7	4
$\boldsymbol{p}_F$	8,8	9

Na Figura 5.2, são mostradas as 9 configurações usadas para treinamento e as 6 configurações usadas para teste do trabalho MINATOHARA; FURUKAWA (2011). Os 9 objetos $(\boldsymbol{p}_i, (i = 1,2,...,9))$ foram usados para treinamento. Enquanto, os 6 objetos $(\boldsymbol{p}_i, (i = A,B,...,F))$ foram usados para teste.

Para o conjunto de treinamento, foram amostrados uniformemente 2001 possibilidades de estado de sistema e comando de controle e aplicada cada uma nas diferentes configurações de planta de treinamento. Dessa maneira, o conjunto de treinamento é composto por tuplas $\{$*estado atual, comando de controle*, *próximo estado, configuração da planta*$\}$ e o total de tuplas é $9 * 2001 = 18009$. Na Tabela 5.2, tem a faixa de amostragem de cada variável. Essas faixas foram selecionadas a partir de observações feitas do comportamentos dessas variáveis quando o sistema foi controlado por um controle PID.

Para o treinamento das redes SOM e LARFSOM, esse conjunto de treinamento foi normalizado para que todas as variáveis ficasse entre 0 e 1. Essa normalização não foi aplicada para os experimentos envolvendo o SOAC porque não é mencionado, em nenhum momento,

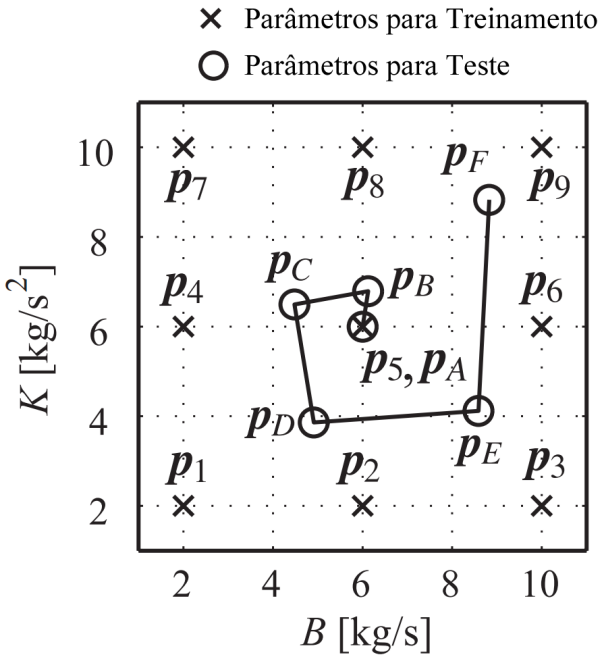

Figura 5.2: Configurações do objeto controlado. Figura adaptada de MINATOHARA; FURUKAWA (2011).

Tabela 5.2: Faixas de amostragens das variáveis

Variável	Min	Max
x (posição)	$-0,6$	$0,6$
v (velocidade)	-4	4
u (Controle)	-2000	2000

normalização dos dados no artigo original MINATOHARA; FURUKAWA (2011).

Para o teste do sistema, foi usada uma trajetória aleatória de 30 segundos e a cada 5 segundos é alterada a configuração da planta de teste. A ordem das configurações de teste sempre é $[\boldsymbol{p}_A, \boldsymbol{p}_B, \boldsymbol{p}_C, \boldsymbol{p}_D, \boldsymbol{p}_E, \boldsymbol{p}_F]$. A trajetória desejada foi gerada a partir de um processo aleatório de *Ornstein-Uhlenbeck* GARDINER et al. (1985). A equação diferencial estocástica (EDE) para esse processo é dada por:

$$dS_t = \lambda(\mu - S_t)dt + \sigma dW_t \tag{5.2}$$

onde λ é taxa de reversão à média, μ é a média e σ é a volatilidade. W_t denota um processo de Wiener.

A Equação 5.3 é uma solução exata para EDE apresentada.

$$S_{t+i} = S_i e^{-\lambda\delta} + \mu(1 - e^{-\lambda\delta}) + \sigma\sqrt{\frac{1 - e^{-2\lambda\delta}}{2\lambda}}N_{0,1} \tag{5.3}$$

onde δ é o tamanho do passo de tempo para a amostragem e $N_{0,1}$ é um número aleatório seguindo

uma distribuição normal com média igual a 0 e desvio-padrão igual a 1.

Os parâmetros usados para Equação 5.3 estão apresentados na Tabela 5.3.

Tabela 5.3: Valores dos parâmetros usado no processo aleatório *Ornstein-Uhlenbeck*

Parâmetro	Valor
λ	3
μ	0
σ	0,5
δ	0,001

Todas as simulações foram conduzidas usando o método de Runge-Kutta BUTCHER (1987) de quarta ordem com tamanho de passo $h = 0,001$ [seg]. O desempenho do controle foi avaliado pelo valor de erro RMSE entre a trajetória da saída do sistema e a trajetória desejada. A Equação 5.4 expressa o erro RMSE.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{x}(t) - x(t))^2}{n}} \tag{5.4}$$

onde $x(t)$ é a posição da massa ao longo do tempo, $\hat{x}(t)$ é a posição desejada, e n é o número de amostras.

5.2 Resultados PDA

Nessa seção, são mostrados os resultados obtidos no experimento massa-mola-amortecedor utilizando um controle convencional. O controle retro-alimentado convencional (cfc) usado foi o Controlador Proporcional Derivativo Acelerativo (PDA). Os parâmetros usados foram ${}^{cfc}\boldsymbol{W} = [k_x, k_{\dot{x}}, k_{\ddot{x}}] = [5\ 10\ 0,5]$. Valores retirados do artigo original do SOAC MINATOHARA; FURUKAWA (2011). Na Figura 5.3, tem-se um resultado do rastreamento do caminho desejado controlado pelo PDA.

A principal característica do resultado obtido usando o PDA é que há um grande erro associado (RMSE = 0,14569). Pode se notar isso no gráfico, principalmente, nos picos da curva desejada. Nesses trechos, o PDA não consegue mandar, para a planta, o comando necessário para rastear adequadamente.

5.3 Resultados SOAC

Nessa seção, são mostrados os resultados obtidos no experimento massa-mola-amortecedor utilizando o SOAC. As entradas dos preditores são a posição x, a velocidade v e o comando de controle u. Cada preditor tem como saída, uma predição da aceleração. As entradas dos controles são a posição desejada x_d, a velocidade desejada v_d e a aceleração desejada a_d. Cada controle

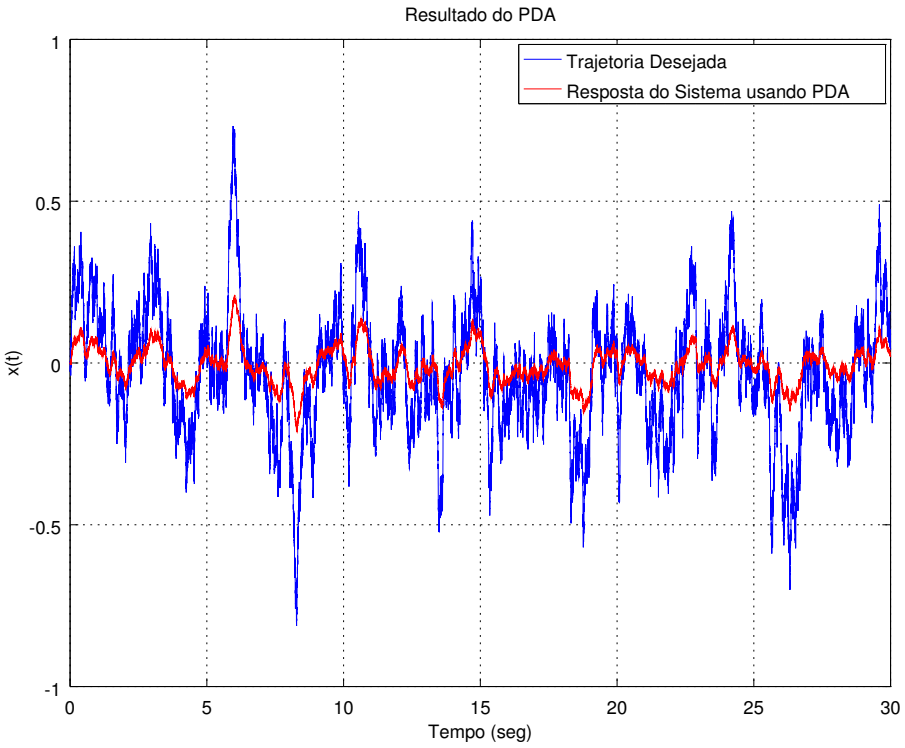

Figura 5.3: Resultado do PDA para o sistema mecânico massa-mola-amortecedor. Nesse rastreamento, o valor do erro RMSE foi de 0,14569.

tem como saída, um comando de controle direto. Tanto o preditor quanto o controle consistem de uma rede linear. O SOAC foi configurado com um mapa bidimensional de 49 (7x7) módulos. Foi escolhido 49 (7x7) módulos, pois foi a configuração que teve melhor desempenho no trabalho original do SOAC MINATOHARA; FURUKAWA (2011). Todos os outros parâmetros usados na configuração do SOAC estão apresentados na Tabela 5.4. Os parâmetros foram escolhidos empiricamente.

Tabela 5.4: Valores dos parâmetros usados no SOAC

Parâmetro	Valor
N (critério de parada)	20
ε	0.1
σ_0	10
σ_{inf}	1
τ	80
η (preditor)	0,0002
η (controle)	0,000001

Na Figura 5.4, tem-se o melhor resultado de rastreamento da trajetória desejada controlado pelo SOAC.

Visualmente, o SOAC obteve um resultado bom. Os rasteamento foi próximo da trajetória

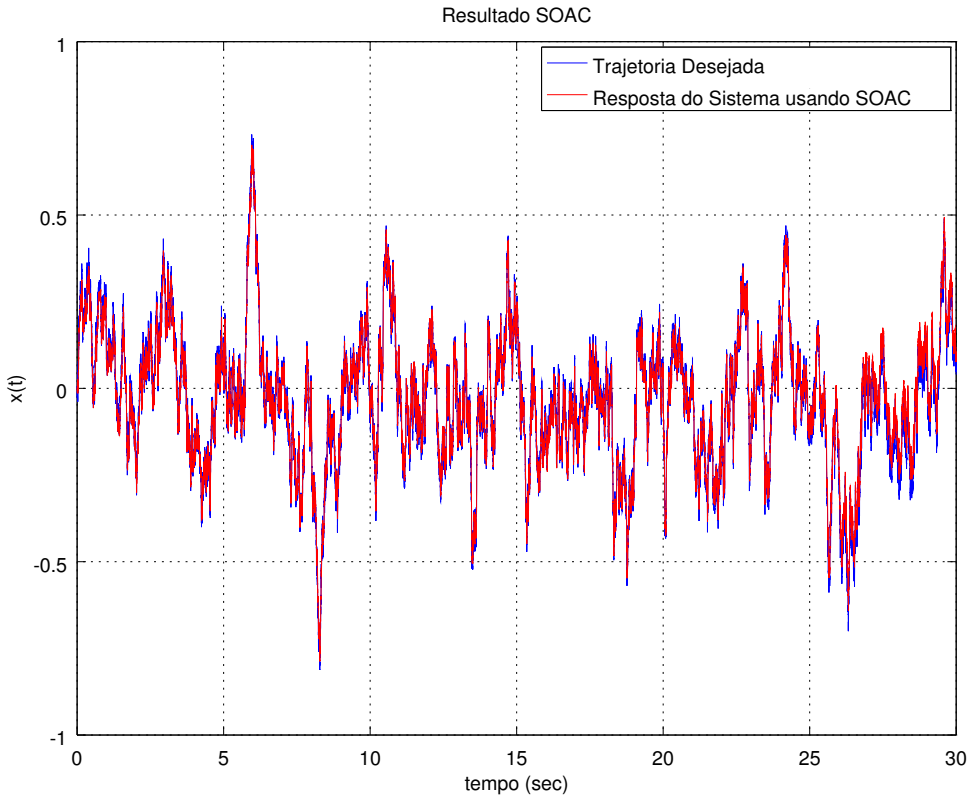

Figura 5.4: Resultado do SOAC para o sistema mecânico massa-mola-amortecedor. Nesse rastreamento, o valor do erro RMSE foi de 0,021073.

desejada. A média e o desvio-padrão do erro RMSE de 10 execuções foram $0,021801 \pm 6,4058 \cdot 10^{-4}$. O principal fato ocorrido no resultado do SOAC é que foi aprendido, continuamente, um comando de correção para o comando do PDA. Isto é, o SOAC foi capaz de corrigir o comando de controle do PDA adicionando um ajuste aprendido de modo *incremental*.

5.4 Resultados SOM

Nessa seção, são mostrados os resultados obtidos no experimento massa-mola-amortecedor utilizando o SOM. Recapitulando, o vetor de peso do nodo k ($\boldsymbol{w}_k$) é dividido em duas partes $\boldsymbol{w}^{in}$ e $\boldsymbol{w}^{out}$, e associado ao nodo tem o vetor ${}^c\boldsymbol{w}^k$, que define a função $\boldsymbol{f}$ (Equação 4.17). O $\boldsymbol{w}^{in}$ corresponde a parte que representa a identificação da serie temporal da planta e $\boldsymbol{w}^{out}$ corresponde a uma estimativa dos parâmetros da planta. Foi escolhido um tamanho de janela $p = 2$. Então, $\boldsymbol{x}_{in}^-$ consiste de $[\boldsymbol{x}(t), u(t), \boldsymbol{x}(t-1), u(t-1)]$, considerando $\boldsymbol{x}(t)$ como sendo a posição $x(t)$, a velocidade $v(t)$ e a aceleração $a(t)$, e o $u(t)$, comando de controle. ${}^c\boldsymbol{w}^k$ consiste de um conjunto de pesos para uma rede linear, semelhante aos controles na rede SOAC. A rede SOM foi ajusta para ser um mapa 9x9. A competição usada na etapa de treinamento foi usando a parte $\boldsymbol{x}_{in}^-$ (Equação 4.13). Todos os outros parâmetros da SOM estão na Tabela 5.5. Os parâmetros foram escolhidos empiricamente.

Tabela 5.5: Valores dos parâmetros usados no SOM

Parâmetro	Valor
N (critério de parada)	50
σ_0	70
σ_{inf}	1
$\tau(\sigma)$	5
α_0	0.9
α_∞	0.0001
$\tau(\alpha)$	30
η (controle)	0,000001

Na Figura 5.5, tem-se o melhor resultado de rastreamento da trajetória desejada controlado pelo SOM.

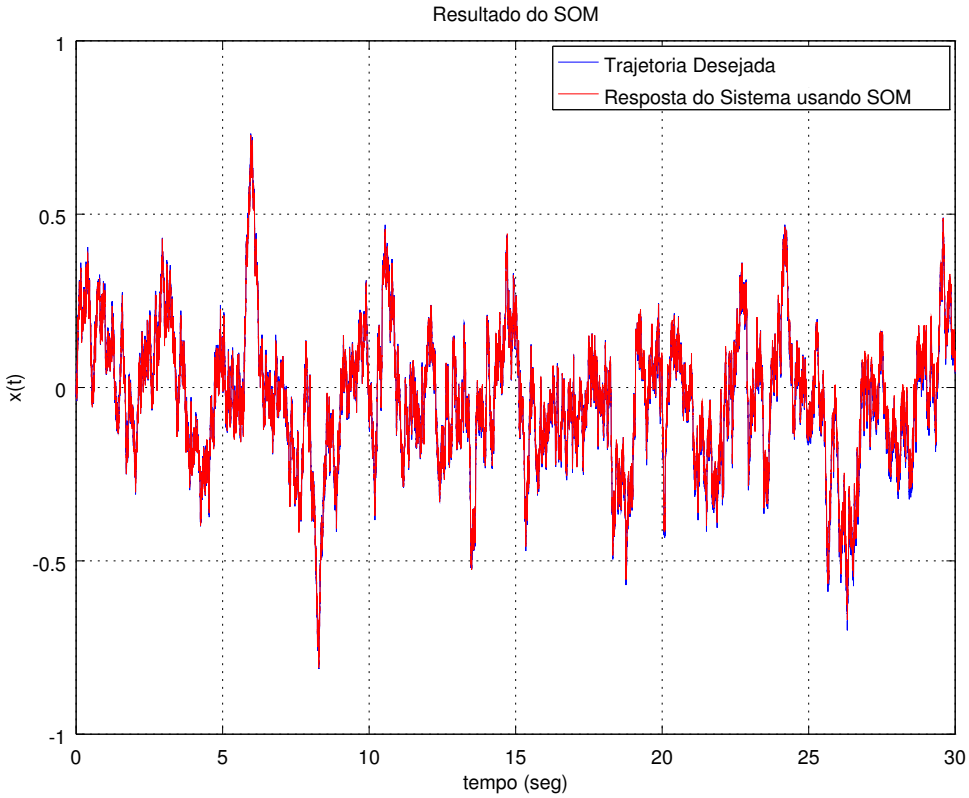

Figura 5.5: Resultado do SOM para o sistema mecânico massa-mola-amortecedor. Nesse rastreamento, o valor do erro RMSE foi de 0,018233.

Visualmente, o SOM obteve um bom resultado assim como o SOAC. Os rasteamentos foram muito próximo da trajetória desejada. A média e o desvio-padrão do erro foram de $0,018322 \pm 4,8662 \cdot 10^{-5}$. O experimento foi executado 10 vezes. Três dos resultados foram semelhantes ao apresentado na Figura 5.5. Pode-se afirmar que o SOM aprendeu, continuamente, semelhante ao SOAC, um comando de correção para o comando do PDA.

5.5 Resultados LARFSOM

Nessa seção, são mostrados os resultados obtidos no experimento massa-mola-amortecedor utilizando o LARFSOM.

Em cada experimento, primeiramente, foi executado o treinamento da rede LARFSOM. Logo em seguinte, é realizado o ajuste dos $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ através do algoritmo genético (Apêndice A). Como o experimento é executado 10 vezes, os $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ encontrados em um experimento anterior são usados na inicialização do algoritmo genético do próximo experimento. Um individuo da primeira população representa $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ do experimento anterior. Isso ocorre em todos experimentos com exceção do primeiro. Na Tabela 5.6, tem-se os parâmetros usados no treinamento da rede LARFSOM. Os parâmetros foram escolhidos empiricamente.

Na etapa de execução, cada vetor de peso ($\boldsymbol{w}$) é constituído da seguinte maneira: $[x_w(t), v_w(t), a_w(t), u_w(t), \tilde{x}_w(t+\Delta t), \tilde{v}_w(t+\Delta t), \tilde{a}_w(t+\Delta t)]$. A parte que corresponde à estimativa da configuração da planta $\tilde{\boldsymbol{p}}_w$ é descartada. Por fim, foi avaliado o desempenho da instância de controle LARFSOM.

Tabela 5.6: Valores dos parâmetros usados no LARFSOM

Parâmetro	Valor
N (critério de parada)	50
ρ_f	0,05
ε	0,3
d_m	25
a_T	1,0

Na Figura 5.6, tem-se o melhor resultado de rastreamento da trajetória desejada controlado pelo LARFSOM.

Visualmente, o LARFSOM obteve um bom resultado. Os rasteamentos foram muito próximos das trajetórias desejadas. A média e o desvio-padrão do erro foram de $0,015682 \pm 1,1924 \cdot 10^{-4}$. O LARFSOM com o ajuste de pesos utilizando um algoritmo genético conseguiu um controle mais refinado, mesmo não usando um comando de controle convencional durante a etapa de execução.

5.6 Comparação dos Métodos

Todos os quatros métodos foram avaliados sobre a mesma trajetória desejada de 30 segundos e sempre mudando a configuração da planta de teste a cada 5 segundos. Na Figura 5.7, há a apresentação dos resultados obtidos pelos métodos que possuem alguma etapa de aprendizagem. Esse tipo de gráfico da estatística descritiva é chamado de *boxplot*. Ao todo, foram executadas 10 tentativas de aprendizagens. No eixo vertical, tem-se o valor de erro RMSE. Enquanto no eixo

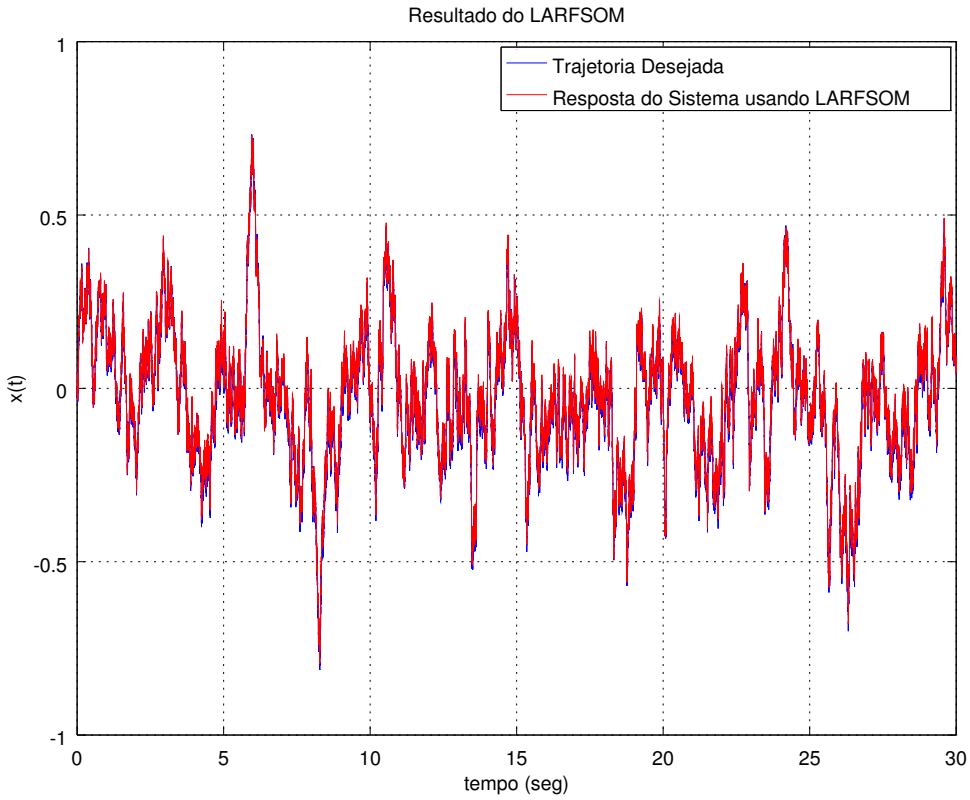

Figura 5.6: Resultado do LARFSOM para o sistema mecânico massa-mola-amortecedor. Nesse rastreamento, o valor do erro RMSE foi de 0,015587.

horizontal, tem-se os métodos analisados. A linha vermelha representa a mediana dos resultados e a "caixa" representa 50% dos resultados obtidos. As retas horizontais menores superiores e inferiores representam os valores máximos e mínimos observados, respectivamente. Poderia haver pontos isolados no gráfico, que são considerados valores atípicos (*outliers*), mas não houve nesses experimentos.

Nesse experimento, todos os 3 métodos neurais de controle foram capazes de responder adequadamente após mudanças nos parâmetros do sistema. Como também, conseguiram controlar o sistema mesmo para parâmetros não apresentados na etapa de treinamento. O LARFSOM foi o método que apresentou melhor performance em relação ao erro RMSE. O principal fato que possibilitou o LARFSOM ter essa performance foi a sua capacidade de generalização. Tal ação de generalizar está associada, principalmente, a regra de ajuste de comando de controle do nodo vencedor (Equação 4.25). O comando de controle final é determinado a partir do comando de controle aprendido pelo nodo vencedor menos uma ponderação de dois erros. O primeiro erro é calculado entre o estado atual da planta e o estado atual representado pelo nodo vencedor. O outro erro é o estado desejado e estado desejado representado pelo nodo vencedor. Essa adaptação do comando controle, através dos dois erros, só é adequada devido a capacidade do algoritmo genético em aprender valores apropriados de $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ para a ponderação.

O SOM teve um bom desempenho e conseguiu superar o SOAC. O projeto do controle

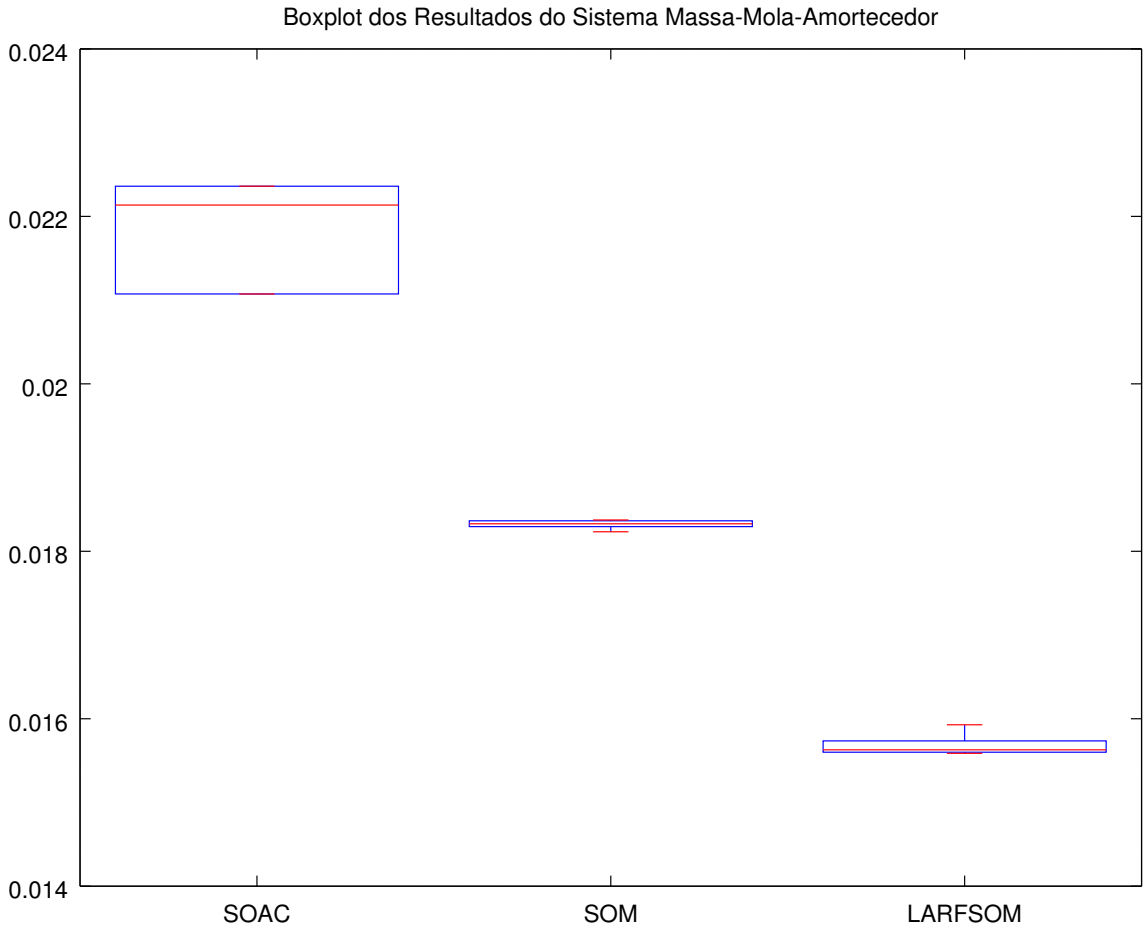

Figura 5.7: Resultados dos experimentos envolvendo os métodos SOAC, SOM e LARFSOM para o sistema massa-mola-amortecedor.

SOM é bastante semelhante ao do SOAC. A principal diferença está no modo de identificação do comportamento da planta atual. Devido a essa diferença, o controle SOM pode identificar mais adequadamente o comportamento e efetivamente ajustar o comando de controle do PDA.

6

Resultados Experimentais - Pêndulo Invertido

A capacidade do SOAC, SOM e LARFSOM também foi avaliada em uma simulação de controle do problema do pêndulo invertido. O pêndulo invertido é um dos problemas mais fundamentais da teoria de controle não-linear e nesse últimos 50 anos, tem sido o sistema mais usado para analisar e comparar os resultados de projetos de controle para problemas não-lineares BOUBAKER (2013). Esse experimento foi confeccionado especialmente para verificar o poder de generalização intrínseco aos modelos. Como foi visto nos Capítulo 3 e 4, esse poder de generalização é herdado da estrutura do SOM. Para o controle LARFSOM, além da arquitetura dos Mapas Auto-organizáveis, o poder de generalização está pertinente à função de ajuste do comando de controle final (Equação 4.25).

Primeiramente, será explicado o conhecimento relativo a modelagem física do problema do pêndulo invertido. Depois, é demonstrada a configuração do experimento. Logo após, são mostrados detalhes paramétricos de cada modelo e os resultados obtidos nas simulações. Por último, é feita uma discussão sobre os resultados das simulações.

6.1 Modelagem da Dinâmica de um Pêndulo Invertido

Nessa seção, é descrita, em detalhes, a modelagem da dinâmica de um pêndulo invertido. Também é mostrado como se deu a linearização desse problema. Por último, são apresentadas as matrizes da modelagem por variáveis de estado.

De maneira simples, o problema do pêndulo invertido consiste em controlar um pêndulo invertido fixado em um carrinho motorizado que pode se movimentar sobre um trilho. O objetivo do controle é conservar, no estado invertido, o pêndulo através de aplicações de forças no carrinho, mesmo quando há perturbações aplicadas ao sistema ou há mudanças na posição desejada do carrinho. Um exemplo do mundo real que se relaciona diretamente com este sistema pêndulo invertido é o controle de altitude de um foguete na decolagem NASA - National Aeronautics and Space Administration (s.d.). No caso, objetivo do problema de controle é manter o foguete

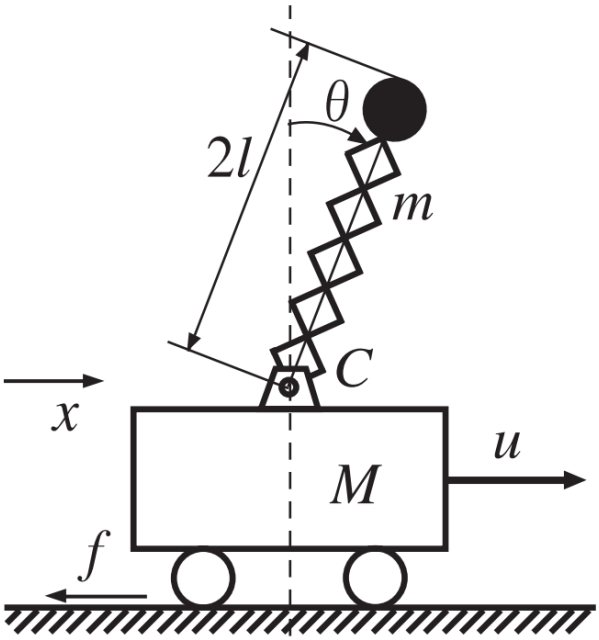

Figura 6.1: Esquema do sistema mecânico do pêndulo invertido. Figura retirada de MINATOHARA; FURUKAWA (2011).

na posição vertical. Também há analogias com problemas, tais como, controle de um braço robótico, posicionamento de guindastes especiais e controle de dedos robóticos RIZK; KRUT; DOMBRE (2007).

Para modelar matematicamente, foi utilizado o esquema de forças apresentado na Figura 6.1. Nesse caso, foi considerado uma versão bidimensional do problema. Sendo assim, o pêndulo só poderá se mover no plano xy mostrado na Figura 6.2. Para esse sistema, a saída de controle é a força u, que move o carrinho na horizontal, e os resultados são observados através dos valores da posição angular do pêndulo θ e da posição horizontal do carrinho x. Para esse experimentos, foram usado valores dos parâmetros do sistema apresentados na Tabela 6.1.

Tabela 6.1: Valores dos parâmetros usado no sistema do pêndulo invertido

Parâmetro	Valor
Massa do carrinho M	$5,0\ [kg]$
Coeficiente de atrito (pêndulo) C_p	$0,0004\ [\frac{kgm^2}{s}]$
Coeficiente de atrito (carrinho) f	$10,0\ [\frac{kg}{s}]$
Aceleração da gravidade g	$9.8\ [\frac{m}{s^2}]$
Ganho a	25

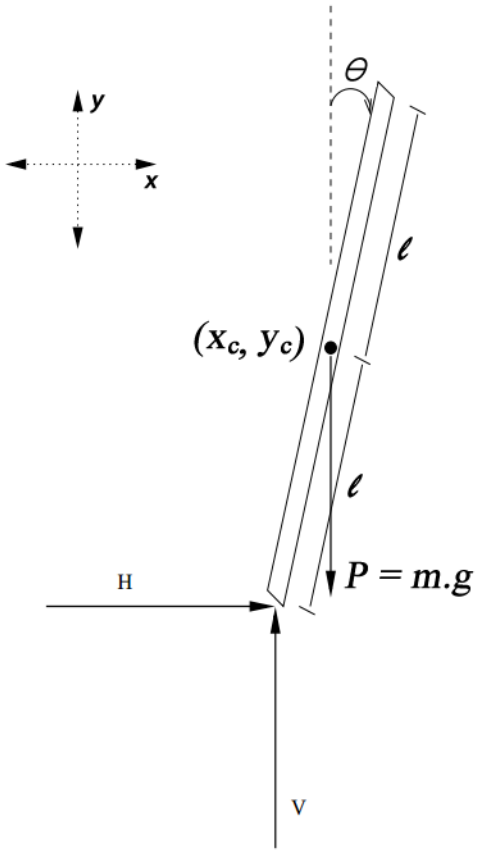

Figura 6.2: Diagrama de corpo livre do pêndulo.

6.1.1 Análise das Forças e Sistema de Equações

Analisando a Figura 6.2, pode-se obter a seguinte expressão para o centróide do pêndulo (Equações 6.1 e 6.2):

$$x_c = x + lsen(\theta); \tag{6.1}$$

$$y_c = lcos(\theta); \tag{6.2}$$

Somando as forças no diagrama de corpo livre do carro na direção horizontal, pode-se obter a seguinte equação de movimento:

$$M\ddot{x} + b\dot{x} + H = a * u \tag{6.3}$$

Para encontrar uma expressão para H, pode-se fazer a soma das forças no diagrama de corpo livre do pêndulo na direção horizontal.

$$H = m\frac{\mathrm{d}^2}{\mathrm{d}t^2}(x + lsen(\theta)) = m\ddot{x} + ml\ddot{\theta}cos(\theta) - ml\dot{\theta}^2 sen(\theta) \tag{6.4}$$

Substituindo H da Equação 6.3 pela Equação 6.4, pode-se obter a seguinte expressão:

$$(M+m)\ddot{x}+ml\ddot{\theta}cos(\theta)-ml\dot{\theta}^2sen(\theta)+f\dot{x}=a*u \tag{6.5}$$

Para obter a segunda equação de movimento para este sistema, soma-se as forças verticais no pêndulo. Sendo assim, pode-se obter a seguinte equação:

$$V-mg=m\frac{\mathrm{d}^2}{\mathrm{d}t^2}(x+lsen(\theta))=-ml\ddot{\theta}sen(\theta)-ml\dot{\theta}^2cos(\theta) \tag{6.6}$$

Fazendo a soma dos momentos de inércia no pêndulo, obtém-se a seguinte expressão:

$$Vlsen(\theta)-Hlcos(\theta)=I\ddot{\theta}+C\dot{\theta} \tag{6.7}$$

onde I é o termo de momento de inércia e é dada pela expressão $I=\frac{ml^2}{3}$.

Combinando as Equações 6.4, 6.6 e 6.7 para eliminar os termos V e H, obtém-se a seguinte expressão:

$$ml\ddot{x}cos(\theta)+(I+ml^2)\ddot{\theta}-mglsen(\theta)+C\dot{\theta}=0 \tag{6.8}$$

As duas equações usadas para modelar o movimento do pêndulo invertido são as Equações 6.5 e 6.8.

6.1.2 Linearização

Como as Equações 6.5 e 6.8 são não-lineares, foi feita uma linearização em torno de $\theta=0$. Pode-se fazer esse processo devido ao fato desse sistema apresentar um comportamento próximo ao linear em torno desse ponto. Então, considerando que θ tem pequenas variações em torno de zero, pode-se usar as seguintes aproximações: $sen(\theta)=\theta$, $cos(\theta)=1$, $\theta^2=0$, $\dot{\theta}^2=0$ e $\theta\dot{\theta}=0$. Sendo assim, as Equações 6.5 e 6.8 podem ser expressas da seguinte forma:

$$(M+m)\ddot{x}+ml\ddot{\theta}+f\dot{x}=a*u \tag{6.9}$$

$$ml\ddot{x}+(I+ml^2)\ddot{\theta}-mgl\theta+C\dot{\theta}=0 \tag{6.10}$$

6.1.3 Variável de Estados

As equações linearizadas do movimento descritas anteriormente também podem ser representadas na forma de espaço de estado. Para isso, elas são reorganizadas em uma série de equações diferenciais de primeira ordem. Uma vez que as equações são lineares, elas poderão, então, ser colocadas na forma de matrizes, como descrito na Equação 6.11.

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = A \cdot \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix} + B \cdot u(t) \tag{6.11}$$

Após manipulações algébricas nas Equações 6.9 e 6.10, os valores obtidos para as matrizes A e B foram os seguintes:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m^2 gl^2}{Den} & \frac{-(I+ml^2)b}{Den} & \frac{-mlc}{Den} \\ 0 & \frac{mgl(M+m)}{Den} & \frac{-mlb}{Den} & \frac{(M+m)C}{Den} \end{bmatrix} \tag{6.12}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{I+ml^2}{Den} \\ \frac{ml}{Den} \end{bmatrix} \tag{6.13}$$

$$Den = I(M+m) + Mml^2 \tag{6.14}$$

6.2 Descrição Experimento

Nessa seção, são explicados os detalhes da execução do experimento do pêndulo invertido. Nesse experimento, foram usadas 9 configurações da planta para treinamento e 9 para testar. Duas das configuração foram usadas tanto para treinar quanto para testar. Novamente, os valores dos parâmetros m_i e l_i usados para teste não foram explicitamente descritos no artigo original do SOAC MINATOHARA; FURUKAWA (2011). Os valores aproximados das configurações de teste do trabalho original do SOAC estão apresentados na Tabela 6.2.

Tabela 6.2: Valores dos parâmetros m e l usado para testar

	m_i	l_i
$\boldsymbol{p}_A$	1	1,2
$\boldsymbol{p}_B$	0,9	1,245
$\boldsymbol{p}_C$	0,8	1,134
$\boldsymbol{p}_D$	0,92	0,915
$\boldsymbol{p}_E$	1,82	0,915
$\boldsymbol{p}_F$	1,48	1,257
$\boldsymbol{p}_G$	1,244	1,71
$\boldsymbol{p}_H$	0,648	1,749
$\boldsymbol{p}_I$	0,2	1,2

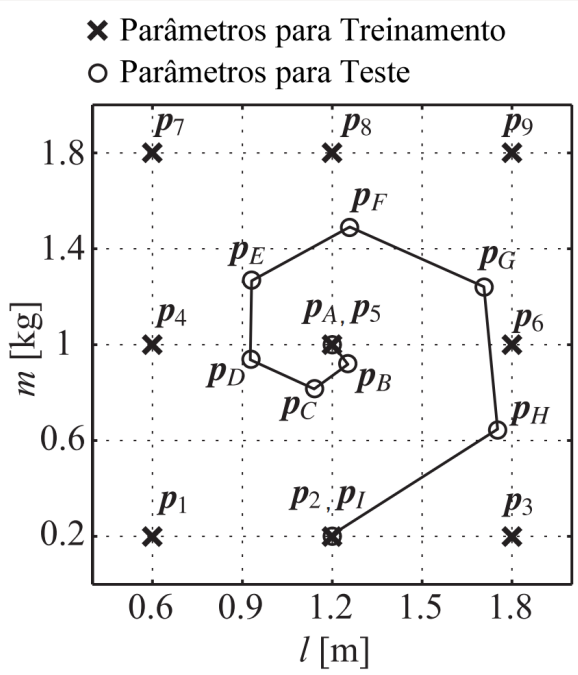

Figura 6.3: Parâmetros da planta usados para treinamento e teste. Figura adaptada de MINATOHARA; FURUKAWA (2011).

Na Figura 6.3, são mostradas as configurações paramétricas do sistema usadas para o treinamento e teste nesse experimento. Na execução do teste, é feita uma mudança da posição desejada do carrinho. Os possíveis valores desejados são $[1,-1,0]$ e a mudança ocorre a cada 10 segundos. O estado inicial é $\boldsymbol{x}_0 = [0,0,0,0]$ e o primeiro valor desejado (posição do carrinho) é $x = 1$. Também ocorre a mudança dos valores das configurações da planta de teste a cada 10 segundos. A ordem das configurações de teste foi $[\boldsymbol{p}_A, \boldsymbol{p}_B, \boldsymbol{p}_C, \boldsymbol{p}_D, \boldsymbol{p}_E, \boldsymbol{p}_F, \boldsymbol{p}_G, \boldsymbol{p}_H, \boldsymbol{p}_I]$. As simulações foram executadas usando o método de Runge-Kutta de 4ª ordem com tamanho de passo $h = 0,01$ [seg].

Para construir o conjunto de treinamento, foram amostrados uniformemente 2001 possibilidades de estado de sistema e comando de controle e aplicada cada uma nas diferentes configurações de planta de treinamento. Dessa maneira, o conjunto de treinamento é composto por tuplas {*estado atual, comando de controle*, *próximo estado, configuração da planta*} e o total de tuplas é $9 * 2001 = 18009$. Na Tabela 6.3, tem a faixa de amostragem de cada variável. Essas faixas foram selecionadas a partir de observações feitas do comportamentos dessas variáveis quando o sistema foi controlado por controles LQR.

Para o treinamento das redes SOM e LARFSOM, esse conjunto de treinamento foi normalizado para que todas as variáveis ficasse entre 0 e 1. Essa normalização não foi aplicada para os experimentos envolvendo o SOAC porque não é mencionado, em nenhum momento, normalização dos dados no artigo original MINATOHARA; FURUKAWA (2011).

Tabela 6.3: Faixas de amostragens das variáveis

Variável	Min	Max
x (posição do carrinho)	$-0,6$	$0,6$
θ (ângulo do pêndulo)	$-0,6$	$0,6$
$\dot{x}$ (velocidade)	-4	4
$\dot{\theta}$ (velocidade do ângulo)	-4	4
u (Controle)	-2000	2000

6.3 Resultados dos Experimentos

Nessa seção, são descritos os detalhes de cada modelo avaliado e são apresentados os resultados obtidos nas simulações do experimento do pêndulo invertido.

6.3.1 Resultados LQR

Nessa subseção, são mostrados os resultados obtidos no experimento do Pêndulo Invertido usando o Regulador Linear Quadrático (LQR). O LQR é um tipo de controle com realimentação proposto por Kalman. O LQR é o projeto de controle ótimo para os sistemas lineares com índice de performance quadrático. Mais precisamente, é gerada uma lei de ganhos de realimentação, que otimiza o índice, através da solução da equação diferencial de Riccati LEWIS (2010). Maiores detalhes sobre regulador linear quadrático, veja o apêndice B.

Foram feitos experimentos com o LQR para fins de comparação. Em um primeiro momento, é mostrado um resultado ideal. Nesse caso, há o conhecimento da configuração corrente da planta de teste. Dessa maneira, é gerado um LQR específico para cada configuração da planta. É ideal porque na pratica, normalmente, não se conhece precisamente a configuração da planta. Na Figura 6.4, é apresentado o resultado dessa simulação.

Também, foi feito um outro experimento usando um único LQR. Dessa vez, foi escolhida uma configuração de planta de treinamento e foi gerado um controle LQR especifico para essa configuração. Foi selecionada a configuração de treinamento mais central, no caso, $\boldsymbol{p}_5$. Esse controle foi usado para todas as configurações de teste. Para melhor entendimento do texto, vai-se denominar esse controle de $LQR_{\boldsymbol{p}_5}$. Na Figura 6.5, é apresentado o resultado desse experimento.

Como pode ser observado, o controle $LQR_{\boldsymbol{p}_5}$ respondeu com instabilidade. Então, foram feitos outros experimentos para tentar identificar quais configurações de teste estavam apresentando um comportamento instável. O resultado foi que para as configurações de teste P_C, P_D, P_E, P_G e P_H, o $LQR_{\boldsymbol{p}_5}$ apresentou um comportamento instável. O experimento sem os casos de teste que o $LQR_{\boldsymbol{p}_5}$ falha está apresentado na Figura 6.6.

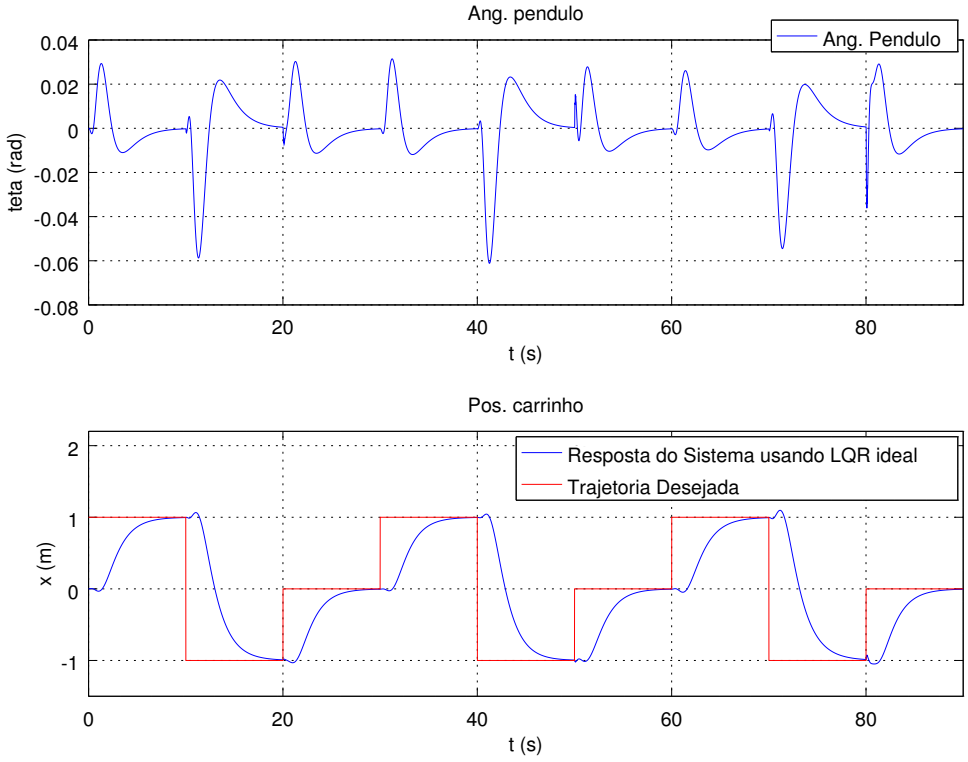

Figura 6.4: Resultado da simulação do LQR para um caso ideal de conhecimento total das configurações da planta. Os valores de massa e comprimento do pêndulo são mudados a cada 10 segundos.

6.3.2 Resultados SOAC

Nessa subseção, são mostrados os resultados obtidos no experimento do Pêndulo Invertido usando o SOAC. Mas, primeiramente, são descritos os detalhes do SOAC para o experimento do pêndulo invertido.

As entradas para o preditor são o estado atual $\boldsymbol{x}(t)$ e o comando de controle $u(t)$. A saída do preditor $\tilde{\boldsymbol{x}}(t+\Delta t)$ é uma predição do estado $\boldsymbol{x}(t+\Delta t)$ onde $\Delta t = 0,01$. Nesse caso, foi utilizado uma rede linear como modelo para a predição. No que diz a respeito dos controladores, foram usados os controles ótimos preparados previamente. Foi confeccionado para cada módulo um regulador linear quadrático (LQR). O número de módulos é 9x9 = 81, o mesmo usado no artigo original do SOAC. Os parâmetros usados na rede SOAC estão apresentados na Tabela 6.4. Na fase de aprendizagem, o treinamento é feito somente a respeito dos preditores. Na fase de execução, é usado somente o comando de controle gerado pelo vencedor (BMM). No caso, o BMM é determinado baseado no erro entre os valores preditos $\tilde{\boldsymbol{x}}(t)$ e os sinais observados de $\boldsymbol{x}(t)$.

Um resultado obtido na fase de aprendizagem é mostrado na Figura 6.7. Consiste do valor do vetor da configuração do sistema $\tilde{\boldsymbol{p}}^k$ (massa e comprimento do pêndulo) que cada módulo do SOAC representa. No caso, esses vetores aprendidos pelo SOAC estão representados

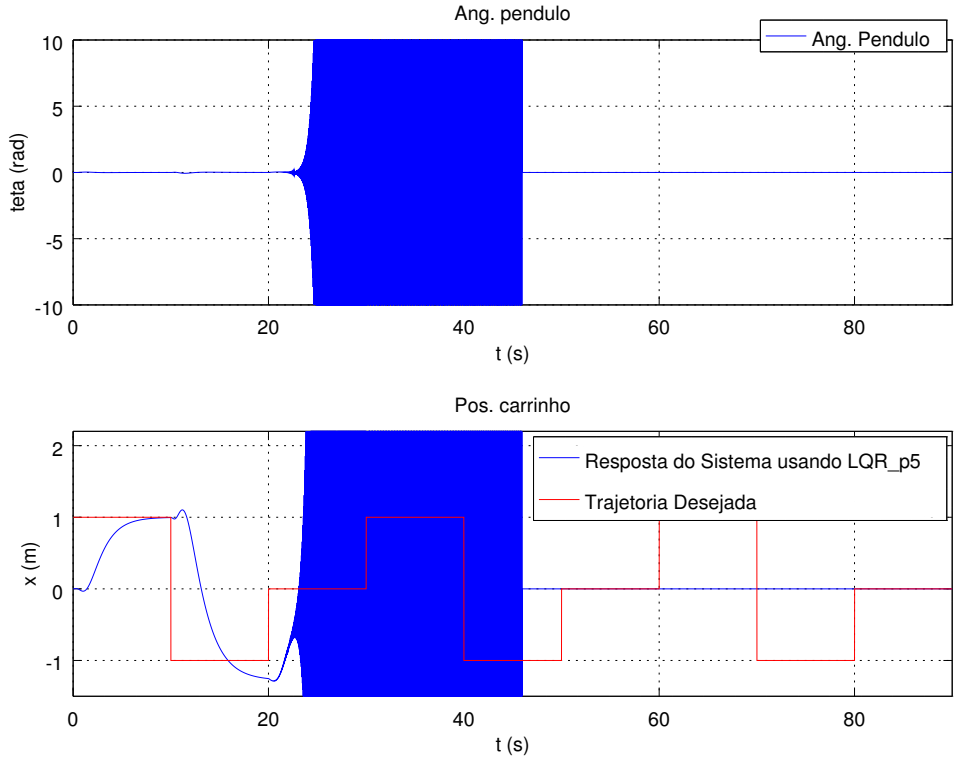

Figura 6.5: Resultado da simulação do $LQR_{\boldsymbol{p}_5}$. No inicio, há uma resposta estável. Porém, o sistema fica instável a partir da configuração de teste P_C. Em torno dos 46 segundos, o sistema começa apresentar valores fora da possibilidade numérica suportada.

Tabela 6.4: Valores dos parâmetros usados no SOAC.

Parâmetro	Valor
N (critério de parada)	300
ε	0.01
σ_0	9
σ_{inf}	1
τ	50
η (preditor)	0.001
η (controle)	Não há aprendizagem no controle

por asteriscos ($*$) azuis. Os xis ($\times$) em vermelho foram usados para treinamento. Já os pontos ($\bullet$) em verde foram utilizados para teste. O controle LQR projetado para cada módulo é feito de modo a otimizar a função de desempenho para a configura de sistema que o módulo representa.

Na Figura 6.8, é apresentado o resultado da simulação do controle SOAC para o exemplo apresentado na Figura 6.7. Esse experimento foi executado 15 vezes. Em todas as execuções, SOAC tem resultados similares ao que está apresentado na Figura 6.8.

Outros experimentos foram realizados para identificar a melhor versão treinada do SOAC.

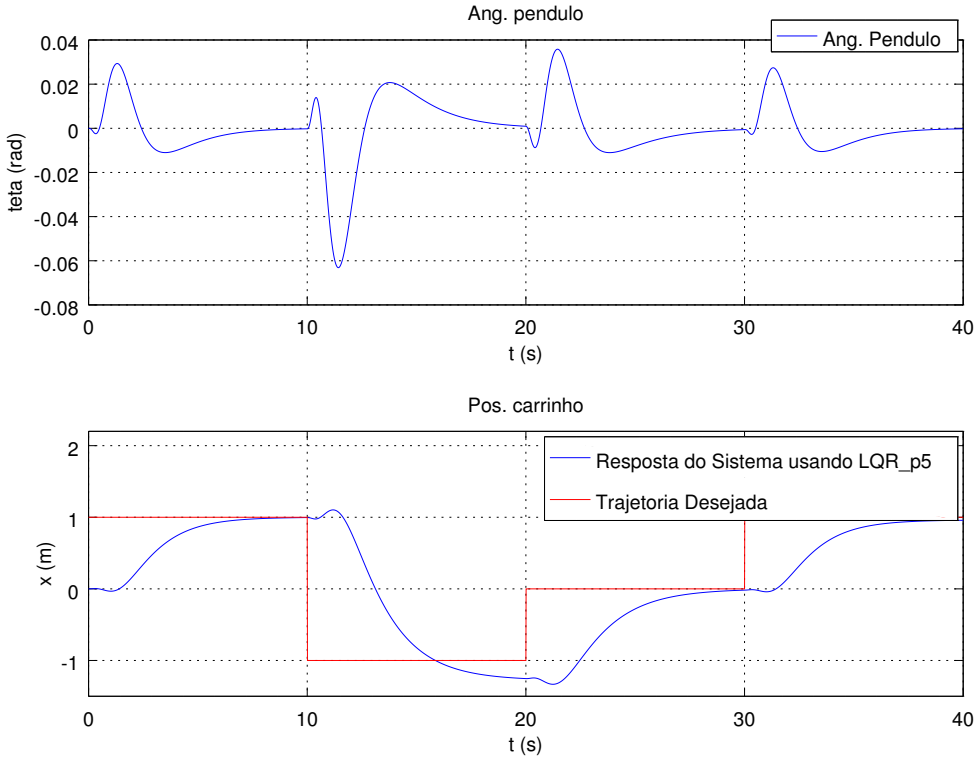

Figura 6.6: Resultado da simulação do $LQR_{\boldsymbol{p}_5}$ sem as configurações P_C, P_D, P_E, P_G e P_H no conjunto de teste.

Para essa instancia, o SOAC não consegui controlar adequadamente para as configurações de teste P_D e P_E. Na figura 6.9, é mostrado o resultado da simulação do controle SOAC sem as configurações que não convergem.

Também foi feita uma mudança para analisar a aprendizagem em relação ao espaço de controles aprendidos. Para isso, foi modificada a forma de seleção do módulo vencedor (BMM). Anteriormente, foi descrito que a escolha do BMM seria com base no erro do preditor. No caso, não levaria em conta informações do comprimento e massa do pêndulo correntes do teste. Nesse experimento, foram levados em conta as informações de comprimento e massa do pêndulo da configuração de teste para seleção do módulo vencedor. Sendo assim, foi descartado o sinal de erro do preditor.

Foi utilizado o $\tilde{\boldsymbol{p}}^k$, que consiste do vetor de configuração aprendido por cada módulo (asteriscos em azul na Figura 6.7), para a escolha do BMM. Então, o módulo vencedor $\tilde{\boldsymbol{p}}^{k^*}$ será o $\tilde{\boldsymbol{p}}^k$ mais próximo do $\boldsymbol{p}_i$ de teste corrente. A medida de distância foi a Euclidiana.

$$\tilde{\boldsymbol{p}}^{k^*} = \underset{k}{agr\,min} \left\| \tilde{\boldsymbol{p}}^k - \boldsymbol{p}_i \right\| \tag{6.15}$$

onde $\boldsymbol{p}_i$ consisti da configuração da planta, isto é, o vetor com o comprimento e massa do pêndulo para o i-ésimo conjunto de teste. O resultado para esse experimento pode ser observado na

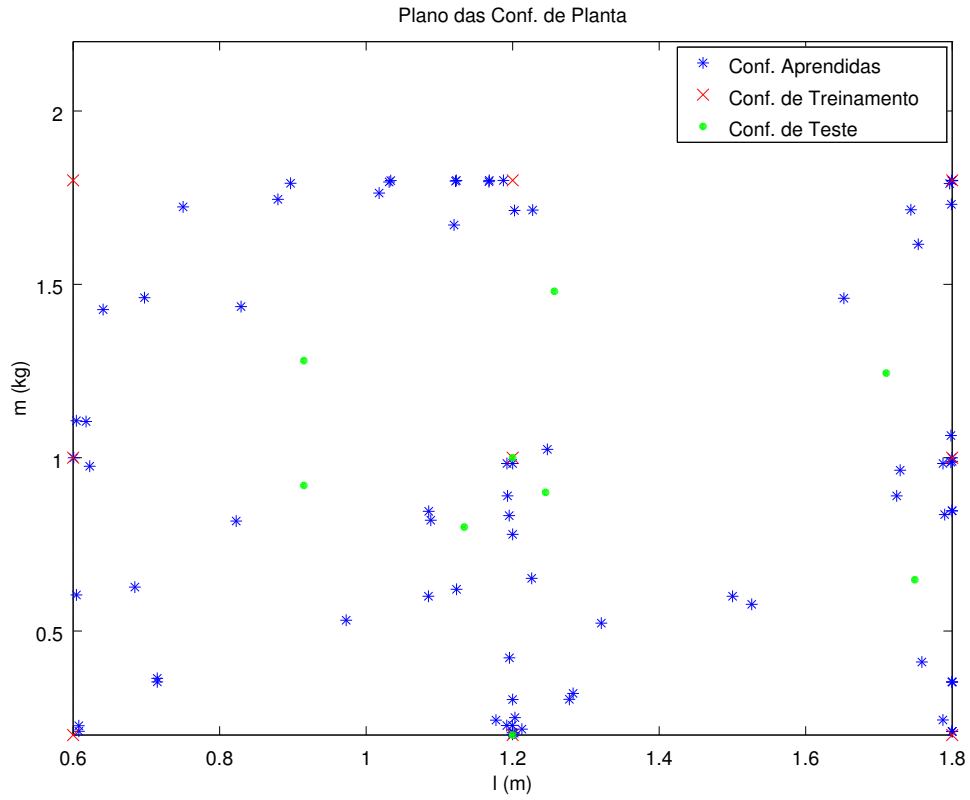

Figura 6.7: Resultado do aprendizado do SOAC no espaço de configurações de planta. Nas abscissas, tem-se os valores do comprimento do pêndulo em metros. Nas ordenadas, tem-se os valores da massa do pêndulo em quilogramas.

Figura 6.10. Das 15 execuções desse experimento, 9 obtiveram exito em controlar o pêndulo e 6 não apresentaram um resultado satisfatório. Os resultados não satisfatórios são similares ao apresentado na Figura 6.8.

6.3.3 Resultados SOM

Nessa subseção, são mostrados os resultados obtidos no experimento do Pêndulo Invertido usando o SOM. Mas, primeiramente, são descritos os detalhes do SOM para o experimento do pêndulo invertido.

Cada vetor de peso ($\boldsymbol{w}$) é dividido em duas partes: $\boldsymbol{w}^{in}$ e $\boldsymbol{w}^{out}$. O $\boldsymbol{w}^{in}$ representa o comportamento da serie temporal da planta para uma janela de tempo específica e $\boldsymbol{w}^{out}$ corresponde a uma estimativa dos parâmetros da planta. No caso, após a etapa de aprendizagem, $\boldsymbol{w}^{out}$ é usado para gerar o controle LQR para cada nodo da rede SOM. A janela de tempo selecionada nesse experimentos tem tamanho $p = 2$. Então, $\boldsymbol{x}_{in}^{-}$ consiste de $[\boldsymbol{x}(t), u(t), \boldsymbol{x}(t-\Delta t), u(t-\Delta t)]$, sendo $\boldsymbol{x}(t)$ o vetor de estado da planta e $u(t)$, o comando de controle. O SOM foi ajustado para ser um mapa 10x10. Todos os outros parâmetros da SOM estão na Tabela 6.5.

Para o experimento pêndulo invertido usando o SOM, foi escolhida a regra de competição na fase de aprendizagem dada pela Equação 4.14 que utiliza os parâmetros da configura planta

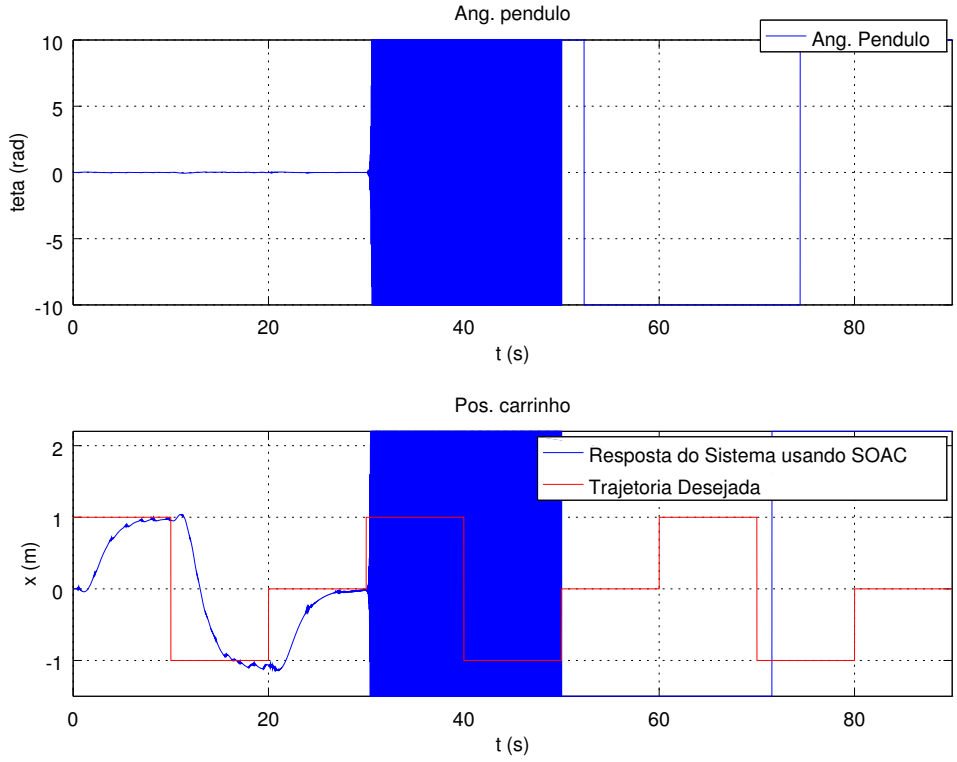

Figura 6.8: Resultado do SOAC. A partir da configuração P_D, o sistema apresentou um comportamento instável e isso reflete no controle das outras configurações de teste.

de treinamento $\boldsymbol{x}_{out}$. Nessa competição ao invés de usar $\boldsymbol{w}_{in}$ para a seleção do nodo vencedor na fase de aprendizagem, usa $\boldsymbol{w}_{out}$. Com essa escolha, há um foco maior na generalização de configurações de plantas ao invés de regiões de operações. Sendo assim, pode-se aprender mais padrões de $\boldsymbol{w}_{out}$. Na Figura 6.11, pode-se verificar as estimativas de $\tilde{\boldsymbol{p}}^k$ aprendidas pela rede SOM. Os $\tilde{\boldsymbol{p}}^k$ aprendidos pelo SOM estão representados por asteriscos ($*$) azuis. Já na Figura 6.12, é apresentado o resultado da simulação desse controle SOM.

6.3.4 Resultados LARFSOM

Nessa subseção, são mostrados os resultados obtidos no experimento do Pêndulo Invertido usando o LARFSOM. Também, são descritos detalhes do LARFSOM para o experimento do pêndulo invertido.

Em cada experimento, primeiramente, foi executado o treinamento da rede LARFSOM e logo em seguinte o ajuste dos $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ usando o algorítimo genético do Apêndice A. Na etapa de execução, cada vetor de peso ($\boldsymbol{w}$) é constituído da seguinte maneira: $[\boldsymbol{x}(t), u(t), \tilde{\boldsymbol{x}}(t+\Delta t)]$, sendo $\boldsymbol{x}(t)$ o vetor de estado da planta, $u(t)$, o comando de controle e $\tilde{\boldsymbol{x}}(t+\Delta t)$, a estimativa de próximo estado. A parte que corresponde à configuração da planta $\tilde{\boldsymbol{p}}_w$ é descartada. Por fim, foi verificado o resultado da instância de controle LARFSOM. Na Tabela 6.6, são mostrados

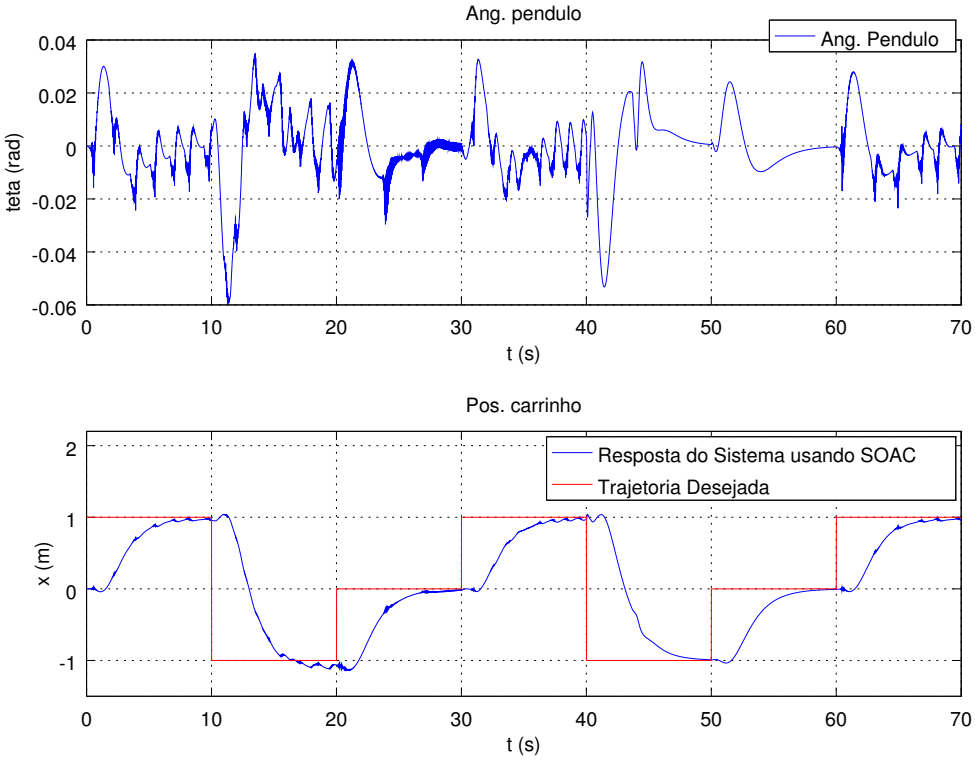

Figura 6.9: Resultado do SOAC para uma simulação sem as configurações de teste P_D e P_E.

os parâmetros usados no treinamento da rede LARFSOM. Já na Figura 6.13, é apresentado o resultado da simulação do controle LARFSOM.

6.4 Considerações Finais

Essa seção é dedicada a uma discussão sobre os resultados obtidos no experimento do Pêndulo Invertido. Os resultados desses experimentos foram analisados pelos gráficos dos resultados na simulação do controle e também sobre o mapa de configurações aprendidas.

O controle SOAC não obteve um resultado satisfatório. Porém, esse resultado insatisfatório não ocorre em todas as configurações. No melhor caso, o SOAC falhou para duas configurações de teste. Também foram feitos experimentos em um SOAC ideal, isto é, um controle SOAC com acesso a informação da configuração de teste. O resultado foi positivo e bastante semelhante ao controle ideal usando LQRs. Esse último fato sugere que a instabilidade do controle SOAC ocorre devido a falhas na identificação da planta de teste. As redes lineares de predição não conseguiram discriminar corretamente as configurações de planta a partir do comportamento delas.

O controle SOM obteve um resultado bom, porém com instantes de leve oscilações e com *offsets* entre o valor desejado e o valor da posição do carrinho. Esses momentos de oscilações

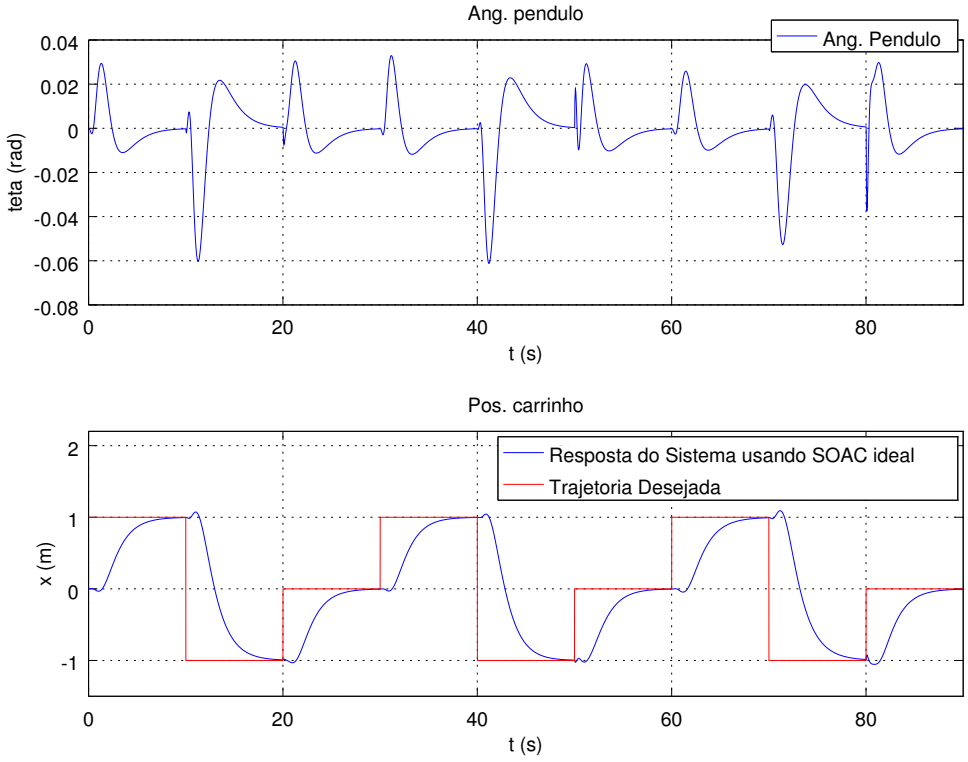

Figura 6.10: Resultado do SOAC com modificação na seleção do vencedor. Resultado bastante semelhante ao resultado do LQR ideal (Figura 6.4).

se apresentaram semelhantes ao que ocorre entre os instantes 60 segundos e 70 segundos da Figura 6.12. E os *offsets* se apresentaram semelhantes ao que acontece no instante de 20 segundos e 50 segundos da Figura 6.12. Essas falhas não comprometeram o comportamento geral do experimento, diferentemente do que acontece no controle SOAC. Nos momentos de oscilações, a escolha da configuração de planta a partir da região de operação não é suficientemente adequada para ter um controle do sistema com resultados análogos ao LQR ideal. Porém, o conjunto de protótipos de configurações do sistema aprendido pelo SOM foi superior ao do SOAC. Esse fato importante deu a controle SOM a capacidade de controlar o sistema sem graves momentos de instabilidades ou oscilações como os que ocorreram com o SOAC. Os protótipos construídos pelo SOM se distribuem mais fidedignamente no espaço de configurações. Enquanto os protótipos aprendidos pelo SOAC se distribuem mais desorganizadamente com uma tendência de irem mais para a borda da região delimitada pelo conjunto de treinamento. Isto também sugere que, nesse experimento, a capacidade de aprender um comportamento geral a partir de um pequeno conjunto de amostras é melhor no controle SOM do que no SOAC.

O sistema teve um comportamento subamortecido quando foi controlado pelo LARFSOM. O principal motivo desse fato é que o módulo do comando de força do controle LARFSOM não diminui tanto quanto LQR a medida que a resposta do sistema vai se aproximando do valor desejado. As vezes acontece o contrário. Ao invés de diminuir o módulo do comando de controle,

Tabela 6.5: Valores dos parâmetros usados no SOM.

Parâmetro	Valor
N (critério de parada)	150
σ_0	70
σ_{inf}	0.5
$\tau(\sigma)$	0.61
α_0	0.01
α_∞	0.0001
$\tau(\alpha)$	30

Tabela 6.6: Valores dos parâmetros usados no LARFSOM

Parâmetro	Valor
N (critério de parada)	50
ρ_f	0,05
ε	0,3
d_m	25
a_T	3,0

o LARFSOM aumenta. Mas, isso ocorre com uma menor frequência. Outro ponto que deve ser observado é que o algoritmo genético em todas as vezes parou pela regra do número máximo de gerações. Enquanto, no experimento do sistema massa-mola-amortecedor, o algoritmo genético quase sempre parou pelo critério de validação. Isso mostra que a construção dos protótipos pelo LARFSOM ainda pode ser melhorada visto que o algoritmo genético tem uma função de ajuste fino. Mas, mesmo assim, é interessante comentar que o comportamento do sistema controlado pelo LARFSOM não teve um comportamento instável.

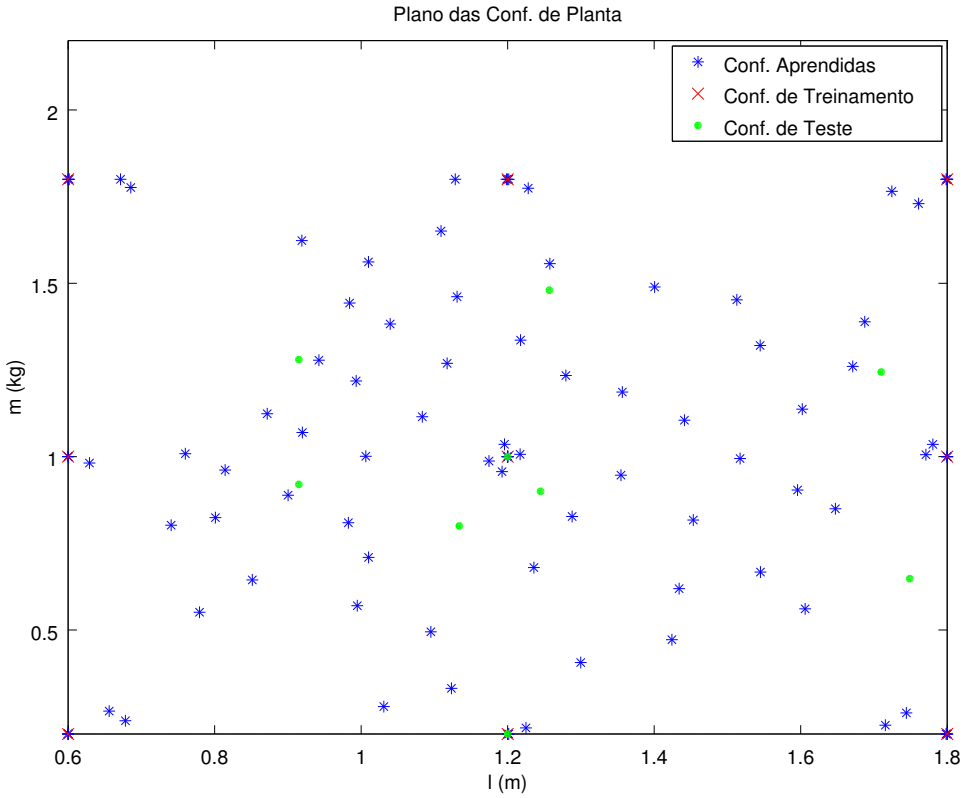

Figura 6.11: Resultado da aprendizagem do SOM.

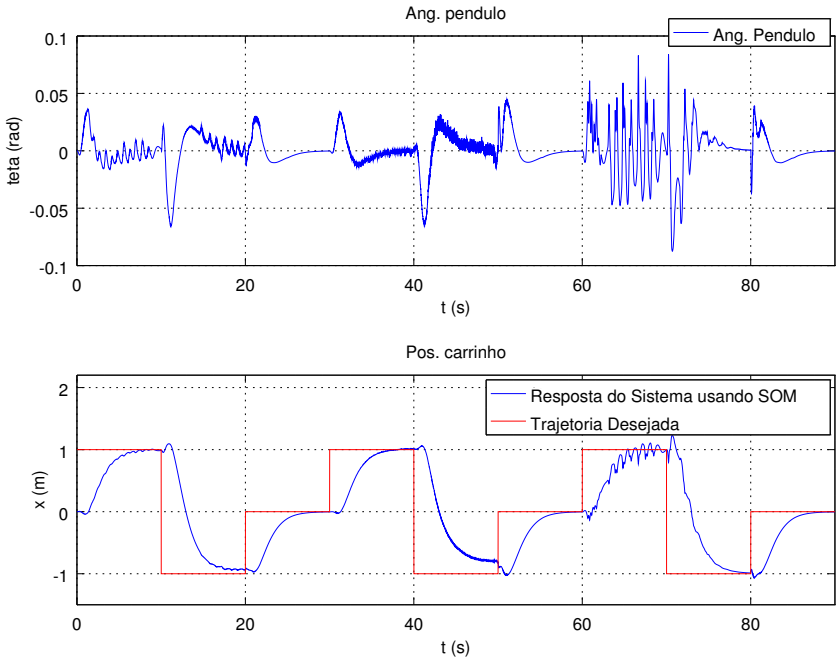

Figura 6.12: Resultado do SOM.

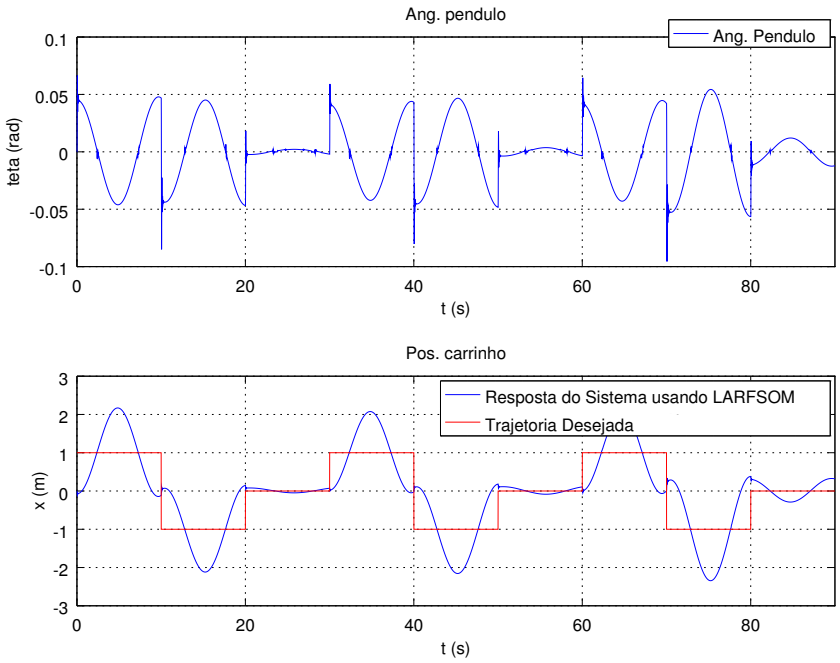

Figura 6.13: Resultado do LARFSOM.

7

Conclusão

Durante esse trabalho de mestrado, foram propostas duas soluções de estratégias de controle inspiradas em Mapas Auto-Organizáveis para sistemas de controle lineares e não-lineares. Uma delas foi concebida sobre o Mapa Auto-Organizável adaptado para identificação do comportamento do sistema. Enquanto a segunda proposta foi confeccionado a partir de uma variante da rede SOM chamada Mapa Auto-organizável com Campo Receptivo Adaptativo Local (LARFSOM) com a característica de ter uma estrutura variante no tempo. A inspiração de projetar controle a partir de Mapas Auto-Organizáveis, chamados nessa dissertação de Controles Auto-Organizáveis, veio de alguns fatores presentes nessas redes. O primeiro deles é aprender sem supervisor. A aprendizagem não-supervisionada pode simplificar o controle e, também, pode permitir uma aprendizagem incremental. Outro fator é que a rede SOM é adequada para o esquema de múltiplos controles. Múltiplos controladores têm sido usados com sucesso para projetar leis de controles capazes de ter respostas adequadas mesmo que haja mudanças repentinas no objeto controlado WOLPERT; KAWATO (1998); MINATOHARA; FURUKAWA (2011) e os mapas podem ser utilizados para determinar e organizar autonomamente o conjunto de módulos de controles. O último fator é o poder de generalização que tem o SOM e suas variantes. Essa característica dá a possibilidade de esquematizar o controle para ser robusto o suficiente e adquirir o comportamento geral a partir de um pequena amostra de exemplos de treinamento.

Assim, foi estudada a subclasse de controle inteligente composta de controles neurais do tipo Mapas Auto-organizáveis. Foram identificadas as características relevantes desse tipo de controle para o problema de sistema de controles. Também, foram apresentadas as limitações dessa subclasse de soluções de controle. De início foi feito um levantamento de métodos auto-organizáveis para controle. Dentre os modelos estudados foi selecionado o SOAC para uma estudo detalhado e implementação. O SOAC consiste de um esquema de múltiplos módulos em que cada módulo tem um par preditor/controlador. O preditor tem o objetivo de identificar o comportamento do sistema. O controlador a ser utilizado é definido a cada instante e é escolhido através do preditor que melhor identifica o comportamento do sistema. O SOAC foi construído sobre uma variante da rede SOM chamada rede modular SOM (mnSOM).

A partir desse estudo, foram projetados dois controles auto-organizáveis. Na primeira proposta, foi utilizada também a arquitetura de múltiplos módulos. Em cada nodo da rede tem uma parte do vetor de pesos dedicada para reconhecimento do sistema e outra, para representar o controlador. Dessa forma, os Mapas Auto-organizáveis foram adaptados para identificação da dinâmica do sistema. Nesse ponto, o projeto de identificação foi inspirado pela Memória Associativa Temporal por Quantização Vetorial (VQTAM). Já a segunda proposta é inspirada na variante da rede SOM chamada Mapa Auto-organizável com Campo Receptivo Adaptativo Local (LARFSOM). A rede LARFSOM tem as seguintes características: Número adaptativo de nodos; topologia variável; criação de novos nodos a medida que não há protótipos semelhantes ao padrão de entrada; similaridade decidida através de um limiar calculado a partir de uma função de base radial; campo receptivo calculado por meio da distância dos dois nodos mais próximos; taxa de aprendizagem dependendo do número de vitória do nodo; liberdade na construção da vizinhança; e remoção dos nodos não-significativos ao final do treinamento. Este segundo controle foi esquematizado através modelagem inversa direta. Cada nodo da rede LARFSOM também foi adaptado para aprender o comportamento do sistema dinâmico. Para isso, o nodo aprende uma estimativa de estado do sistema, de comando de controle e de futuro estado do sistema após a aplicação da estimativa de comando de controle. Na etapa de execução, é selecionado o comando de controle que tem melhor estimativas de estado do sistema e estado desejado (futuro) do sistema. O comando selecionado é aplicado após um ajuste, que é feito a partir dos erros entre as estimativas e o valor real do sistema. Esse ajuste é ponderado por pesos que foram encontradas por um algoritmo genético.

Foram realizados experimentos em simulações de dois sistemas de controle. Um deles foi o sistema Massa-Mola-Amortecedor. Este sistema se caracteriza por ser linear, de segunda ordem e por ter uma natureza estável em malha aberta. O outro foi o sistema Pêndulo Invertido, um sistema mais complexo do que o primeiro, conceituado como um dos problemas mais fundamentais da teoria de controle não-linear. Ideal para estudos de sistema de natureza instável. Todos os dois sistemas são usado para analisar e comparar os resultados de projetos de controle para problemas lineares (o primeiro) e não-lineares (o segundo). Em ambas simulações, os parâmetros dos sistemas variaram ao longo do tempo. Isso foi feito para provocar um comportamento mais complexo nesses sistemas dinâmicos e verificar a capacidade dos controles em responder a mudanças repentinas. Todos os experimentos foram executados com os controles auto-organizáveis, e também, com métodos convencionais para efeito de comparação. O método convencional usado no sistema Massa-Mola-Amortecedor foi o controle PDA. O método convencional usado no experimento do Pêndulo Invertido foi o controle ótimo LQR. Os resultados obtidos no experimento Massa-Mola-Amortecedor foram analisados graficamente e pelo erro RMSE. Enquanto os resultados do experimento Pêndulo Invertido foram analisados graficamente. Neste último experimento, também foi feita uma comparação da aprendizagem de configurações de sistema entre os métodos SOAC e controle SOM.

Resultados do experimento sistema Massa-Mola-Amortecedor mostraram uma superio-

ridade no desempenho de rastrear o valor desejado dos métodos propostos. O destaque nessas simulações foi para o controle LARFSOM que obteve a menor média do RMSE, 0,015682. Enquanto a média do RMSE do controle SOM foi igual a 0,018322 e a do SOAC foi 0,021801. O método convencional utilizado para comparação foi o controle PDA. Esse método de controle não conseguiu rastear adequadamente e seu erro RMSE foi significantemente mais alto. O RMSE do PDA foi 0,14569. Esses resultados demonstraram que as soluções propostas podem ser usadas para sistemas lineares variantes no tempo.

Já os experimentos do sistema Pêndulo Invertido mostraram uma limitação do controle LARFSOM. Os resultados do LARFSOM apresentaram um comportamento subamortecido. O LARFSOM apresentou dificuldades em identificar corretamente a região de operação para escolha adequada dos comandos de controles. Um fator que implicou nesse mal desempenho do LARFSOM foi a maneira de indicar o valor desejado. No primeiro experimento, o valor da posição desejada era dada por uma função continua. Enquanto no segundo, era dada por uma função discreta. Sendo assim, sugere-se que o LARFSOM é mais adequado para planejadores de trajetórias desejadas contínuas. Já os resultados do SOM foram superiores. O SOM apresentou melhor desempenho de rasteamento e a aprendizagem de protótipos de configurações foi melhor do que a do SOAC. As configurações aprendidas pelo SOM foram distribuídas mais uniformemente. Já o SOAC não identificou corretamente em algumas configuração de teste na simulação. Acarretou em escolhas inadequadas de controladores locais levando a estados instáveis no sistema. Mas, ao testar um SOAC, com prévio conhecimento das configurações de teste, os resultados são bastante semelhantes ao do controle ótimo ideal. O método LQR ideal, com conhecimento das configurações de teste, obteve o melhor desempenho. Mas, o LQR projetado a partir de uma configuração específica de treinamento não conseguiu controlar pêndulo invertido e o sistema é levado a estados instáveis.

Para trabalhos futuros, considera-se investigar melhorias no controle LARFSOM para rastrear o valor desejado no sistema Pêndulo Invertido. Uma pesquisa a ser feita é como melhorar sua operação mesmo para valores desejados regidos por funções discretas. Também se sugere que realize o desenvolvimento de um controle baseado no LARFSOM, mas na abordagem de aprendizagem por realimentação do erro. Uma ideia é inserir novos módulos a medida que a qualidade dos controles locais não atingir um desempenho esperado. É interessante um estudo dos controles SOAC, SOM e LARFSOM em outras plataformas de experimentos. Pretende-se explorar a capacidade desses controles em um sistema de controle de posição de dedos de uma mão robótica. É importante, também, avaliar o desempenho destes controles sobre os aspectos de tempo e recursos necessário para processamento. Isto é visando uma aplicação destes controles em sistema de tempo real embarcado em um sistema robótico.

Referências

ALBUS, J. A theory of intelligent systems. In: INTELLIGENT CONTROL, 1990. PROCEEDINGS., 5TH IEEE INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 1990. p.866–875 vol.2.

ALBUS, J. Outline for a theory of intelligence. **Systems, Man and Cybernetics, IEEE Transactions on**, [S.l.], v.21, n.3, p.473–509, 1991.

ARAúJO, A. F. R.; RÊGO, R. L. M. E. Self-organizing Maps with a Time-varying Structure. **ACM Comput. Surv.**, New York, NY, USA, v.46, n.1, p.7:1–7:38, July 2013.

ARAúJO, A. R.; COSTA, D. C. Local adaptive receptive field self-organizing map for image color segmentation. **Image and Vision Computing**, [S.l.], v.27, n.9, p.1229–1239, 2009.

ÅSTRÖM, K. J.; HÄGGLUND, T. The future of PID control. **Control Engineering Practice**, [S.l.], v.9, n.11, p.1163 – 1175, 2001. PID Control.

ÅSTRÖM, K.; WITTENMARK, B. **Adaptive Control**. [S.l.]: Addison-Wesley, 1995. (Addison-Wesley Series in Electrical Engineering).

BARRETO, G. Time Series Prediction with the Self-Organizing Map: a review. In: HAMMER, B.; HITZLER, P. (Ed.). **Perspectives of Neural-Symbolic Integration**. [S.l.]: Springer Berlin Heidelberg, 2007. p.135–158. (Studies in Computational Intelligence, v.77).

BARRETO, G.; ARAúJO, A. F. R. A self-organizing NARX network and its application to prediction of chaotic time series. In: NEURAL NETWORKS, 2001. PROCEEDINGS. IJCNN '01. INTERNATIONAL JOINT CONFERENCE ON. **Anais...** [S.l.: s.n.], 2001. v.3, p.2144–2149 vol.3.

BARRETO, G.; ARAúJO, A. F. R. Identification and control of dynamical systems using the self-organizing map. **Neural Networks, IEEE Transactions on**, [S.l.], v.15, n.5, p.1244–1259, Sept 2004.

BARRETO, G.; ARAúJO, A.; RITTER, H. Self-Organizing Feature Maps for Modeling and Control of Robotic Manipulators. **Journal of Intelligent and Robotic Systems**, [S.l.], v.36, n.4, p.407–450, 2003.

BERNSHTEIN, N. A. **The co-ordination and regulation of movements**. [S.l.]: Pergamon Press, 1967.

BOUBAKER, O. The Inverted Pendulum Benchmark in Nonlinear Control Theory: a survey. **Int J Adv Robotic Sy**, [S.l.], v.10, n.233, 2013.

BUESSLER, J. et al. Multiple self-organizing maps to facilitate the learning of visuo-motor correlations. In: SYSTEMS, MAN, AND CYBERNETICS, 1999. IEEE SMC'99 CONFERENCE PROCEEDINGS. 1999 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1999. v.3, p.470–475.

BUTCHER, J. C. **The numerical analysis of ordinary differential equations**: runge-kutta and general linear methods. [S.l.]: Wiley-Interscience, 1987.

CHO, J. et al. Modeling and inverse controller design for an unmanned aerial vehicle based on the self-organizing map. **Neural Networks, IEEE Transactions on**, [S.l.], v.17, n.2, p.445–460, 2006.

CHO, J. et al. Quasi-sliding mode control strategy based on multiple-linear models. **Neurocomputing**, [S.l.], v.70, n.4, p.960 – 974, 2007.

CHUANG, C.-H. et al. Application of Self-Organizing Map (SOM) for Cerebral Cortex Reconstruction. **International Journal of Computational Intelligence Research**, [S.l.], v.3, n.1, p.26–30, 2007.

COITON, Y. et al. A neural network model for the intersensory coordination involved in goal-directed movements. **Biological cybernetics**, [S.l.], v.66, n.2, p.167–176, 1991.

LEVINE, W. S. (Ed.). **Control system advanced methods**. 2 ed..ed. [S.l.]: CRC Press, 2010. p.19–1–19–13. (Electrical Engineering Handbook). 3.

DAYUAN, G.; ZHENG, M.; HAI, Z. Modeling and control of unmanned aerial vehicle using self-organizing map multiple models. In: CONTROL AND DECISION CONFERENCE (CCDC), 2013 25TH CHINESE. **Anais...** [S.l.: s.n.], 2013. p.4177–4182.

FENG, G. A Survey on Analysis and Design of Model-Based Fuzzy Control Systems. **Fuzzy Systems, IEEE Transactions on**, [S.l.], v.14, n.5, p.676–697, Oct 2006.

GARDINER, C. W. et al. **Handbook of stochastic methods**. [S.l.]: Springer Berlin, 1985. v.3.

GE, M.; CHIN, M.-S.; WANG, Q.-G. An extended self-organizing map for nonlinear system identification. In: DECISION AND CONTROL, 1999. PROCEEDINGS OF THE 38TH IEEE CONFERENCE ON. **Anais...** [S.l.: s.n.], 1999. v.1, p.1065–1070 vol.1.

GOMI, H.; KAWATO, M. Neural network control for a closed-loop System using Feedback-error-learning. **Neural Networks**, [S.l.], v.6, n.7, p.933 – 946, 1993.

GOPALAN, N.; DEISENROTH, M.; PETERS, J. Feedback error learning for rhythmic motor primitives. In: ROBOTICS AND AUTOMATION (ICRA), 2013 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.1317–1322.

GU, D.-W.; PETKOV, P. H.; KONSTANTINOV, M. M. Robust Control of a Mass-Damper-Spring System. In: **Robust Control Design with MATLAB®**. [S.l.]: Springer London, 2005. p.101–162. (Advanced Textbooks in Control and Signal Processing).

HAYKIN, S. S. **Neural Networks and Learning Machines**. [S.l.]: Prentice Hall, 2009. p.790 – 842. (Neural networks and learning machines).

HAYKIN, S. S. **Neural Networks and Learning Machines**. [S.l.]: Prentice Hall, 2009. p.425 – 470. (Neural networks and learning machines).

HOLLAND, J. H. **Adaptation in natural and artificial systems**: an introductory analysis with applications to biology, control, and artificial intelligence. [S.l.]: U Michigan Press, 1975.

HOU, Z.-S.; WANG, Z. From model-based control to data-driven control: survey, classification and perspective. **Information Sciences**, [S.l.], v.235, n.0, p.3 – 35, 2013. Data-based Control, Decision, Scheduling and Fault Diagnostics.

HUNT, K. et al. Neural networks for control systems—A survey. **Automatica**, [S.l.], v.28, n.6, p.1083 – 1112, 1992.

JONES, M.; VERNON, D. Using neural networks to learn hand-eye co-ordination. **Neural Computing & Applications**, [S.l.], v.2, n.1, p.2–12, 1994.

KANGAS, J. Time-delayed self-organizing maps. In: NEURAL NETWORKS, 1990., 1990 IJCNN INTERNATIONAL JOINT CONFERENCE ON. **Anais...** [S.l.: s.n.], 1990. p.331–336.

KASKI, S.; KANGAS, J.; KOHONEN, T. Bibliography of self-organizing map (SOM) papers: 1981–1997. **Neural computing surveys**, [S.l.], v.1, n.3&4, p.1–176, 1998.

KATAYAMA, M.; KAWATO, M. Learning Trajectory and Force Control of an Artificial Muscle Arm by Parallel-hierarchical Neural Network Model. In: CONFERENCE ON ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 3, 1990., San Francisco, CA, USA. **Proceedings...** Morgan Kaufmann Publishers Inc., 1990. p.436–442. (NIPS-3).

KAWATO, M. Feedback-error-learning neural network for supervised motor learning. **Advanced neural computers**, [S.l.], v.6, n.3, p.365–372, 1990.

KAWATO, M. Computational schemes and neural network models for formation and control of multijoint arm trajectory. **Neural networks for control**, [S.l.], p.197–228, 1990.

KAWATO, M.; FURUKAWA, K.; SUZUKI, R. A hierarchical neural-network model for control and learning of voluntary movement. **Biological cybernetics**, [S.l.], v.57, n.3, p.169–185, 1987.

KAWATO, M.; GOMI, H. A computational model of four regions of the cerebellum based on feedback-error learning. **Biological cybernetics**, [S.l.], v.68, n.2, p.95–103, 1992.

KAWATO, M.; GOMI, H. The cerebellum and VOR/OKR learning models. **Trends in Neurosciences**, [S.l.], v.15, n.11, p.445–453, 1992.

KIHL, H. et al. Neural network based hand-eye positioning with a transputer-based system. In: HIGH-PERFORMANCE COMPUTING AND NETWORKING. **Anais...** [S.l.: s.n.], 1995. p.281–286.

KIM, C.-H.; IM, S.-M.; LEE, J.-J. Adaptive control of robot manipulators using an incremental fuzzy system. In: MECHATRONICS AND AUTOMATION, 2008. ICMA 2008. IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2008. p.66–71.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological cybernetics**, [S.l.], v.43, n.1, p.59–69, 1982.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, [S.l.], v.78, n.9, p.1464–1480, Sep 1990.

KOHONEN, T. An Overview of SOM Literature. In: **Self-Organizing Maps**. [S.l.]: Springer Berlin Heidelberg, 2001. p.347–371. (Springer Series in Information Sciences, v.30).

KOHONEN, T. Applications. In: **Self-Organizing Maps**. [S.l.]: Springer Berlin Heidelberg, 2001. p.263–310. (Springer Series in Information Sciences, v.30).

LEVINE, W. S. (Ed.). **Control system advanced methods**. 2.ed. [S.l.]: CRC Press, 2010. p.53–1–53–25. (Electrical Engineering Handbook).

LAFLAMME, S.; SLOTINE, J. J. E.; CONNOR, J. J. Wavelet network for semi-active control. **Journal of Engineering Mechanics**, [S.l.], v.137, n.7, p.462–474, Jul 2011.

LAFLAMME, S.; SLOTINE, J. J. E.; CONNOR, J. J. Self-organizing input space for control of structures. **Smart Materials and Structures**, [S.l.], v.21, n.11, p.115015, NOV 2012.

LEVINE, W. S. (Ed.). **Control system advanced methods**. 2.ed. [S.l.]: CRC Press, 2010. p.32–1–36–19. (Electrical Engineering Handbook). 3.

LEVINE, W. S. (Ed.). **Control system advanced methods**. 2.ed. [S.l.]: CRC Press, 2010. p.25–1–25–35. (Electrical Engineering Handbook).

LIAN, R.-J. Intelligent controller for robotic motion control. **Industrial Electronics, IEEE Transactions on**, [S.l.], v.58, n.11, p.5220–5230, Nov 2011.

LIAN, R.-J. Grey-prediction self-organizing fuzzy controller for robotic motion control. **Information Sciences**, [S.l.], v.202, p.73–89, Oct 2012.

LIAN, R.-J. Enhanced Adaptive Self-Organizing Fuzzy Sliding-Mode Controller for Active Suspension Systems. **Industrial Electronics, IEEE Transactions on**, [S.l.], v.60, n.3, p.958–968, Mar 2013.

LIAN, R.-J.; HUANG, C.-N. Self-organizing fuzzy radial basis function neural-network controller for robotic motion control. **International Journal of Innovative Computing, Information and Control**, [S.l.], v.8, n.3A, p.1955–1971, Mar 2012.

LIU, X.-L.; DUAN, G.-R.; TEO, K.-L. Optimal soft landing control for moon lander. **Automatica**, [S.l.], v.44, n.4, p.1097 – 1103, 2008.

LOCATELLI, A. **Optimal Control**: an introduction. [S.l.]: Springer, 2001. p.1 – 5.

MAMDANI, E. Application of fuzzy algorithms for control of simple dynamic plant. **Electrical Engineers, Proceedings of the Institution of**, [S.l.], v.121, n.12, p.1585–1588, December 1974.

MAMDANI, E.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International Journal of Man-Machine Studies**, [S.l.], v.7, n.1, p.1 – 13, 1975.

MINATOHARA, T.; FURUKAWA, T. Self-organizing adaptive controllers: application to the inverted pendulum. In: WORKSHOP ON SELF-ORGANIZING MAPS (WSOM05), FRANCE, 5. **Proceedings...** [S.l.: s.n.], 2005.

MINATOHARA, T.; FURUKAWA, T. The self-organizing adaptive controller. **International Journal of Innovative Computing, Information and Control (IJICIC)**, [S.l.], v.7, n.4, p.1933–1947, 2011.

MIYAMOTO, H. et al. Feedback-error-learning neural network for trajectory control of a robotic manipulator. **Neural Networks**, [S.l.], v.1, n.3, p.251–265, 1988.

MOSHOU, D.; RAMON, H. Extended Self-Organizing Maps with Local Linear Mappings for Function Approximation and System Identification. In: WORKSHOP ON SELF-ORGANIZING MAPS. **Proceedings...** [S.l.: s.n.], 1997.

MURRAY-SMITH, R.; JOHANSEN, T. A. **Multiple Model Approaches to Modelling and Control**. [S.l.]: Taylor and Francis, London, 1997.

MURRAY-SMITH, R.; JOHANSEN, T. **Multiple Model Approaches to Nonlinear Modelling and Controles**. [S.l.]: CRC press, 1997.

NARENDRA, K. S.; BALAKRISHNAN, J. Adaptive control using multiple models. **Automatic Control, IEEE Transactions on**, [S.l.], v.42, n.2, p.171–187, 1997.

NARENDRA, K. S.; BALAKRISHNAN, J.; CILIZ, M. K. Adaptation and learning using multiple models, switching, and tuning. **Control Systems, IEEE**, [S.l.], v.15, n.3, p.37–51, 1995.

National Aeronautics and Space Administration. **Rocket Stability**. Acessado em: 26-12-2013, `http://exploration.grc.nasa.gov/education/rocket/rktstab.html`.

NISE, N. S. **Control Systems Engineering**. 6.ed. [S.l.]: John Wiley & Sons, Incorporated, 2011. p.1 – 31.

OGATA, K. **System Dynamics**. [S.l.]: Pearson/Prentice Hall, 2004. p.53 – 100.

OGATA, K. **Engenharia de controle moderno**. [S.l.]: Pearson Prentice Hall, 2011. p.1 – 18.

OGATA, K. **Engenharia de controle moderno**. [S.l.]: Pearson Prentice Hall, 2011. p.749 – 789.

LEVINE, W. S. (Ed.). **Control system advanced methods**. 2 ed..ed. [S.l.]: CRC Press, 2010. p.54–1–54–12. (Electrical Engineering Handbook). 3.

PRABHU, S. M.; GARG, D. P. Artificial neural network based robot control: an overview. **Journal of Intelligent and Robotic Systems**, [S.l.], v.15, n.4, p.333–365, 1996.

PRINCIPE, J. C.; WANG, L. Non-linear time series modeling with Self-Organization Feature Maps. In: NEURAL NETWORKS FOR SIGNAL PROCESSING [1995] V. PROCEEDINGS OF THE 1995 IEEE WORKSHOP. **Anais...** [S.l.: s.n.], 1995. p.11–20.

PRINCIPE, J. C.; WANG, L.; MOTTER, M. A. Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control. **Proceedings of the IEEE**, [S.l.], v.86, n.11, p.2240–2258, 1998.

PROCYK, T.; MAMDANI, E. A linguistic self-organizing process controller. **Automatica**, [S.l.], v.15, n.1, p.15 – 30, 1979.

PSALTIS, D.; SIDERIS, A.; YAMAMURA, A. A multilayered neural network controller. **IEEE control systems magazine**, [S.l.], v.8, n.2, p.17–21, 1988.

RAINEY III, L. W. **Digital Control of the Utah/MIT Dexterous Hand**: initial evaluation and analysis. 1990. Dissertação (Mestrado em Ciência da Computação) — Faculty of the School of Engineering of the Air Force Institute of Technology.

RITTER, H. J.; MARTINETZ, T. M.; SCHULTEN, K. J. Topology-conserving maps for learning visuo-motor-coordination. **Neural networks**, [S.l.], v.2, n.3, p.159–168, 1989.

RIZK, R.; KRUT, S.; DOMBRE, E. Grasp-stability analysis of a two-phalanx isotropic underactuated finger. In: INTELLIGENT ROBOTS AND SYSTEMS, 2007. IROS 2007. IEEE/RSJ INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.3289–3294.

SABAHI, K.; GHAEMI, S.; PEZESHKI, S. Application of type-2 fuzzy logic system for load frequency control using feedback error learning approaches. **Applied Soft Computing**, [S.l.], v.21, p.1–11, AUG 2014.

SHAO, S. Fuzzy self-organizing controller and its application for dynamic processes. **Fuzzy Sets and Systems**, [S.l.], v.26, n.2, p.151 – 164, 1988. Fuzzy Control.

SUH, S.-H. et al. PID Control System. In: **Theory and Design of CNC Systems**. [S.l.]: Springer London, 2008. p.157–185. (Springer Series in Advanced Manufacturing).

TEWARI, A. **Advanced Control of Aircraft, Spacecraft and Rockets**. [S.l.]: Wiley, 2011. p.195 – 275. (Aerospace Series).

THAMPI, G. et al. Adaptive inverse control using SOM based multiple models. In: PORTUGUESE CONF. AUTOMATIC CONTROL. **Proceedings...** [S.l.: s.n.], 2002. p.278–282.

TOKUNAGA, K.; FURUKAWA, T. Modular network SOM. **Neural Networks**, [S.l.], v.22, n.1, p.82 – 90, 2009.

WALTER, J.; RITER, H.; SCHULTEN, K. Nonlinear prediction with self-organizing maps. In: NEURAL NETWORKS, 1990., 1990 IJCNN INTERNATIONAL JOINT CONFERENCE ON. **Anais...** [S.l.: s.n.], 1990. p.589–594 vol. 1.

WALTER, J.; RITTER, H. Rapid learning with parametrized self-organizing maps. **Neurocomputing**, [S.l.], v.12, n.2, p.131–153, 1996.

WOLPERT, D. M.; KAWATO, M. Multiple paired forward and inverse models for motor control. **Neural Networks**, [S.l.], v.11, n.7, p.1317–1329, 1998.

WONG, R. H. Parametric Design of Coupled Adaptive Self-Organizing Sliding-Mode Fuzzy Controller in a Valve-Controlled Folding Machine. In: APPLIED MECHATRONICS AND ANDROID ROBOTICS. **Anais...** [S.l.: s.n.], 2013. p.92–95. (Applied Mechanics and Materials, v.418).

YIN, H. The Self-Organizing Maps: background, theories, extensions and applications. In: FULCHER, J.; JAIN, L. (Ed.). **Computational Intelligence**: a compendium. [S.l.]: Springer Berlin Heidelberg, 2008. p.715–762. (Studies in Computational Intelligence, v.115).

ZELLER, M.; SHARMA, R.; SCHULTEN, K. Topology representing network for sensor-based robot motion planning. In: WORLD CONGRESS ON NEURAL NETWORKS, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.100–103.

ZELLER, M.; SHARMA, R.; SCHULTEN, K. Motion planning of a pneumatic robot using a neural network. **Control Systems, IEEE**, [S.l.], v.17, n.3, p.89–98, 1997.

ZELLER, M.; WALLACE, K. R.; SCHULTEN, K. Biological visuo-motor control of a pneumatic robot arm. In: INTELLIGENT ENGINEERING SYSTEMS THROUGH ARTIFICIAL NEURAL NETWORKS. PROCEEDINGS OF THE ARTIFICIAL NEURAL

NETWORKS IN ENGINEERING CONFERENCE, NEW YORK. **Anais...** [S.l.: s.n.], 1995. v.5, p.645–650.

Apêndice

A

Algoritmo Genético

Nesse apêndice, é descrito o algoritmo genético utilizado para ajustar os pesos de $\boldsymbol{k}_a^T$ e $\boldsymbol{k}_d^T$ da Equação 4.25. Primeiramente, é abordado um contexto sobre algoritmos genético. Logo em seguida, é definido e detalhado o algoritmo genético projetado. Por fim, é apresentado o diagrama de fluxo do algoritmo genético.

Algoritmos Genéticos

Algoritmos Genéticos (AG) é uma classe de algoritmos usados para encontrar soluções em problemas de otimização. É também a classe de Algoritmos Evolucionários mais conhecida. Os AGs foram concebidos por Holland HOLLAND (1975) que se inspirou fortemente na ideia darwiniana da origem das espécies. Holland fundamentou que o funcionamento dos AGs consiste das seguintes etapas: Inicialização da população, avaliação dos indivíduos, seleção dos pais, cruzamento, mutação, atualização da população e critério de parada. Na Figura A.1, é apresentado o diagrama de fluxo do funcionamento de um algoritmo genético convencional.

Inicialmente, o AG cria um conjunto de soluções. Cada solução é um indivíduo e todas as soluções formam a população. Tais soluções são submetidas ao processo de evolução e novas soluções vão surgindo. O processo de evolução é constituído pelas seguintes etapas:

Avaliação dos indivíduos: Avaliação da aptidão do indivíduo, isto é, o quão boa é a solução.

Seleção de pais: Seleção de indivíduos para a reprodução. Essa seleção é estocástica e a probabilidade de cada indivíduo é proporcional à aptidão dele.

Cruzamento: Características de indivíduos são selecionadas e recombinadas para gerar novos indivíduos.

Mutação: Características dos novos indivíduos são alteradas aleatoriamente.

Atualização da População: Os novos indivíduos são inseridos na população.

Critério de parada: Verificação do critério de finalização do processo de evolução.

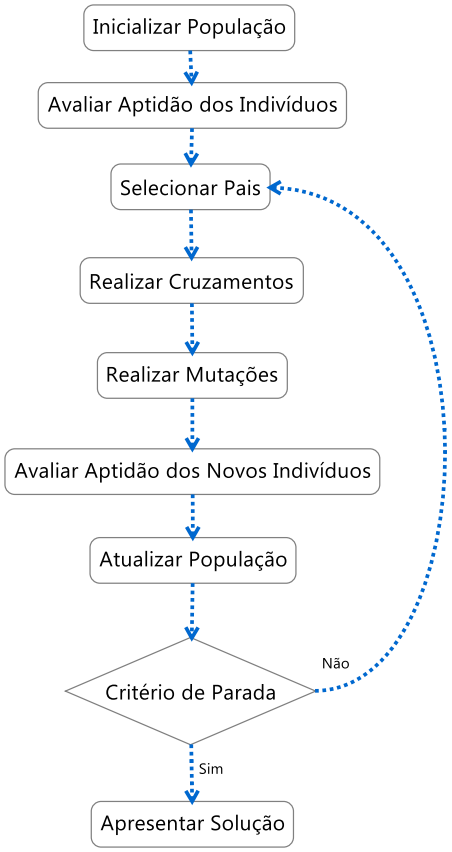

Figura A.1: Diagrama de fluxo do funcionamento do algoritmo genético.

AG Implementado

Agora, é demonstrado, em detalhes, o algoritmo genético projetado. Primeiro, é definida a representação do indivíduo. Em seguida, são mostrados a mutação e o cruzamento. Depois, são apresentados o modelo de população e a seleção de pais. Por fim, são descritos o modo de avaliação da aptidão e o critério de parada.

Representação: A representação usada foi um vetor de valores reais. Os primeiros elementos do vetor representam $\boldsymbol{k}_a^T$ e os últimos, $\boldsymbol{k}_d^T$ ($[\boldsymbol{k}_a^T, \boldsymbol{k}_d^T]$).

Mutação: Há duas mutações que podem ocorrer. Cada uma com 12% de probabilidade, mas nunca acontece as duas simultaneamente no mesmo indivíduo. As duas mutações consistem de alterar aleatoriamente o valor de cada alelo. A diferença está na aleatoriedade. Uma usa uma distribuição uniforme entre $[-1,1]$, e outra, uma distribuição normal com $\mu = 0$ e $\sigma = 1$. Na equação A.1, tem-se a formalização da mutação.

$$[x'_1, \cdots, x'_n] \longleftarrow [x_1, \cdots, x_n] + \alpha * [d_1, \cdots, d_n] \tag{A.1}$$

onde $[x'_1, \cdots, x'_n]$ é o indivíduo modificado pela mutação, $[x_1, \cdots, x_n]$ é o indivíduo selecionado para mutação, α é uma valor aleatório $[0,1]$ e d_i é outro valor aleatório que pode ser da distribuição uniforme ou da normal.

Recombinação: A recombinação usada é a *recombinação aritmética completa*. Consiste de uma soma ponderada dos alelos de um pai com o outro (Equação A.2).

$$filho = \alpha * \boldsymbol{x} + (1 - \alpha) * \boldsymbol{y} \tag{A.2}$$

onde $\boldsymbol{x}$ e $\boldsymbol{y}$ são os pais selecionados para cruzar e α é uma valor aleatório $[0,1]$.

Modelo de população: Foi escolhido o modelo geracional (*generational model*). Nesse modelo, a cada geração, toda a população é substituída pela sua prole, "a nova geração". Sendo assim, em uma população de tamanho γ, são selecionados 2 * γ pais para o cruzamento, em cada geração. Foi utilizado $\gamma = 30$ em todos os experimentos. A população inicial é composta por indivíduos criados aleatoriamente. Para o experimento do sistema massa-mola-amortecedor, cada indivíduo foi construído da seguinte forma: $\boldsymbol{x}_i = [e_1, -1 * e_2, e_3, -1 * e_4, e_5, -1 * e_6]$, onde e_i é um número aleatório de uma distribuição exponencial com $\lambda = 1$. Foi utilizado esse tipo de aleatoriedade porque se observou que as melhores soluções para esse problema tinham o formato $[+,-,+,-,+,-]$. Para o experimento do Pêndulo Invertido, cada indivíduo foi construído da seguinte forma: $\boldsymbol{x}_i = [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8]$, onde e_i é um número aleatório de uma distribuição uniforme entre $-1{,}5$ e $-1{,}5$.

Seleção de Pais: a escolha dos pais é feita através da *seleção por torneio*. Na seleção usada, eram sorteados dois indivíduos da população com reposição e o mais bem avaliado, com maior aptidão, era escolhido como um dos pais. São realizados 2 * γ torneios. Após todos torneios, os pais são casados (pareados) dois a dois, na ordem dos torneios.

Avaliação: A avaliação é feita da seguinte forma: É aplicado $[\boldsymbol{k}_a^T, \boldsymbol{k}_c^T]$ do indivíduo no problema alvo. Avalia o sistema de controle ao rastrear uma determinada trajetória desejada. A trajetória desejada é a mesma para todos indivíduos de todas as gerações e é diferente da trajetória desejada usada nos testes apresentados nos capítulos dos experimentos. Após a simulação, é calculado o erro entre a trajetória da saída do sistema e a desejada. O erro mensurado é RMSE, expresso na Equação 5.4. Então, a aptidão do indivíduo é o inverso do erro calculado.

Critério de Parada: Foi aplicada a técnica *hold-out* como critério de parada. Para isso, há duas trajetórias desejadas além da de teste. Uma trajetória é a de treinamento e outra, a de validação. Em cada população, é verificada se encontrou uma nova melhor solução levando em consideração todas as gerações. Caso melhore k vezes, em seguida, a solução para a trajetória de treinamento e não melhore para a trajetória de validação, é finalizado o processo de evolução e a solução que teve melhor desempenho para a trajetória de validação é apresentada como a solução. Há também outro critério de parada, que é o número máximo de gerações $n_{geracoes}$. Foi utilizado nos experimentos $k = 10$ e $n_{geracoes} = 100$.

A Figura A.2 apresenta o fluxo do funcionamento do AG implementado.

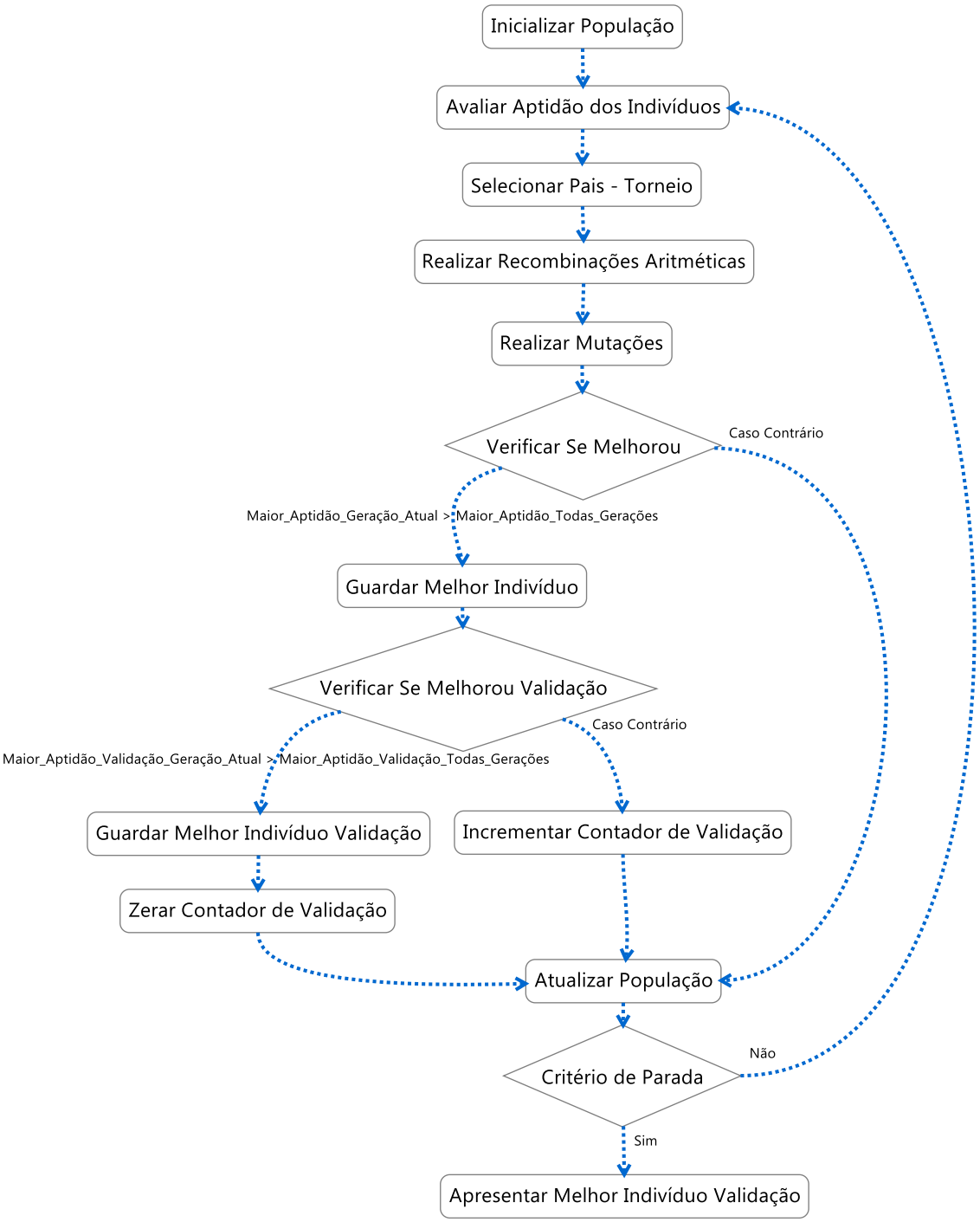

Figura A.2: Diagrama de fluxo do funcionamento do algoritmo genético implementado.

B

Geração de Coeficientes de Realimentação

Esse apêndice é dedicado a apresentar a geração de coeficiente para o controle usado no pêndulo invertido. Para esse experimento, em alguns momentos, foi utilizado um controle chamado *regulador quadrático linear (LQR)* LEWIS (2010). Mais precisamente, foi aplicado o controle LQR para problemas de rastreamento visto que há mudanças na posição desejada do carrinho. Para o problema de rastreamento, o LQR também é conhecido por rastreador LQ.

O LQR é um controle realimentado proposto por Kalman. O LQR é o projeto de controle ótimo para sistemas representados por equações diferenciais lineares (Eq. B.2) com índice de performance quadrático (Eq. B.5). Mais precisamente, é gerada uma lei de ganhos de realimentação, que otimiza o índice, através da solução da equação diferencial de Riccati LEWIS (2010).

O funcionamento do regulador LQ é manter o estado do sistema próximo de zero para garantir estabilidade. Outro problema fundamental no projeto de sistemas é controlar um sistema de que a saída apresente um comportamento específico, isto é, siga uma trajetória de referência diferente de zero $r(t)$. Um exemplo é no problema de pêndulo invertido feito nos experimentos. A posição desejada do carrinho muda ao longo do tempo. Esse tipo de problema é chamado de problema de *rastreamento* ou *servodesign* LEWIS (2010). Para essa finalidade a lei de controle do LQR é alterada. A questão fundamental aqui é que para o rastreamento ótimo alguns termos diretos (*feedforward*) devem ser adicionados à entrada do controle no circuito fechado realimentado LQR básico que vai garantir a estabilidade do sistema.

Considerando o sistema linear representado pela equação B.2, com $z(t)$, dado pela equação B.3, sendo a saída do sistema que deve rastrear um dado sinal de referência $r(t)$. O LQR também pode controlar para o caso que $z(t)$ e $r(t)$ são vetores, isto é, o caso de rastreamento de múltiplas saídas. A entrada do controle é dada por:

$$u = -Kx + v \tag{B.1}$$

onde $v(t)$ é um sinal de realimentação requerido para o desempenho de um bom rastreamento. K são os ganhos de retroalimentação. Tanto K quanto $v(t)$ são determinados de modo que o erro de rastreamento (Equação B.4) seja pequeno. A Tabela B.1 apresenta as equações necessárias para

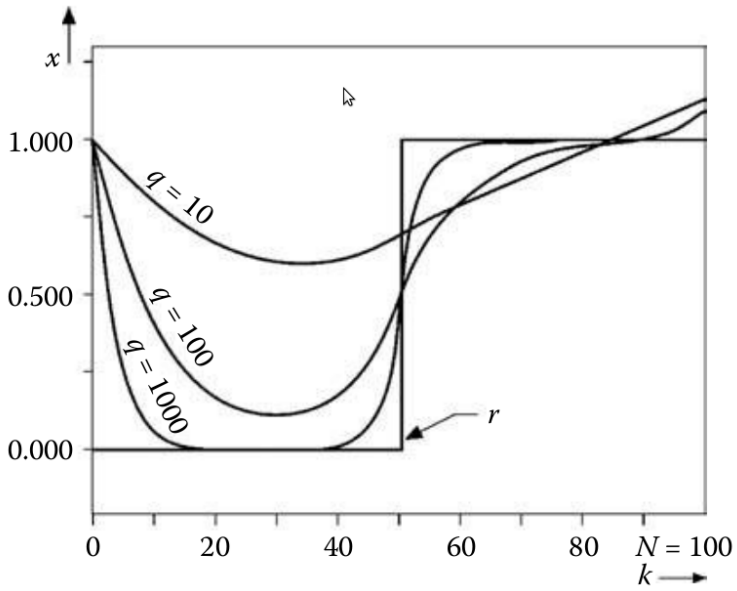

Figura B.1: Resposta antecipada do rastreador LQ ótimo.

implementação desse controle ótimo para o problema de rastreamento.

A estrutura de realimentação é basicamente a mesma do LQR apresentado em LEWIS (2010). Mas, há também um controlador PI que contém o erro de rastreamento. A presença do sinal de erro de rastreamento $e(t)$ no controle PI tem o efeito da adição na alimentação de entrada de um termo de velocidade do erro $v(t)$ para o sinal de controle. O ganho de rastreador de K e sinal $v(t)$ são determinados pela abordagem ótima LQ, dadas pelas equações apresentadas na Tabela B.1. O ganho de realimentação K é determinado da mesma maneira como no caso LQ tipo regulador.

O sinal de alimentação de entrada é calculado usando o sistema dinâmico (Equação B.8que é chamado o sistema adjunto. Na realidade, o rastreador LQ ótimo é não-causal, porque há necessidade do conhecimento de valores futuros da entrada de referência $r(t)$. Esse valores futuros são utilizados para calcular $w(t)$. As ramificações desta natureza não-causal do rastreador ótimo encontram-se ilustrados na Figura B.1, que mostra a resposta de rastreador ótima para um sistema de controle de peso escalar utilizando R = 1 e diferentes valores do coeficiente de erro de saída P. Pode ser observado que o sistema começa a alterar seu comportamento antes da referência $r(t)$, de modo que o sistema antecipa as alterações em $r(t)$. Este comportamento antecipatório é uma característica importante do rastreador ótimo.

Tabela B.1: Controle Rastreador Linear Quadrático

Modelagem do sistema

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}, \quad t \geq t_0, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{B.2}$$

Saída do sistema

$$\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x} \tag{B.3}$$

Erro de rastreamento

$$\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{z} \tag{B.4}$$

Índice de desempenho

$$J(t_0) = \frac{1}{2}\boldsymbol{e}^T(T)\boldsymbol{P}\boldsymbol{e}(T) + \frac{1}{2}\int_{t_0}^{T}(\boldsymbol{e}^T\boldsymbol{Q}\boldsymbol{e} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})dt \tag{B.5}$$

$$\boldsymbol{P} \geq 0, \quad \boldsymbol{Q} \geq 0, \quad \boldsymbol{R} > 0$$

Controle ótimo para rastreamento
Equação de Riccati

$$-\dot{\boldsymbol{S}} = \boldsymbol{A}^T\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A} - \boldsymbol{S}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{S} + \boldsymbol{C}^T\boldsymbol{Q}\boldsymbol{C}, \quad t \leq T, \quad \boldsymbol{S}(T) = \boldsymbol{C}^T\boldsymbol{P}\boldsymbol{C} \tag{B.6}$$

Ganho de retroalimentação ótimo

$$\boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{S} \tag{B.7}$$

Sistema para frente (Feedforward)

$$-\dot{\boldsymbol{w}} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})^T\boldsymbol{w} + \boldsymbol{C}^T\boldsymbol{Q}\boldsymbol{r}, \quad t \leq T, \quad \boldsymbol{w}(T) = \boldsymbol{C}^T\boldsymbol{P}\boldsymbol{r}(T) \tag{B.8}$$

Controle para frente (Feedforward) mais retroalimentação

$$\boldsymbol{u}(t) = -\boldsymbol{K}(t)\boldsymbol{x}(t) + \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{w}(t) \tag{B.9}$$