Fernando Passold

Controle Neural de Posição e Força em Manipuladores Robóticos

Florianópolis
2003

Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia Elétrica

Controle Neural de Posição e Força em Manipuladores Robóticos

Tese submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Doutor em Engenharia Elétrica

Fernando Passold

Florianópolis, 12 de Dezembro de 2003.

CONTROLE NEURAL DE POSIÇÃO E FORÇA EM MANIPULADORES ROBÓTICOS

Fernando Passold

'Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica, Área de Concentração em *Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Prof. Dr. Ing. Marcelo Ricardo Stemmer
Orientador

Prof. Dr. Jefferson Luiz Brum Marques
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Dr. Ing. Marcelo Ricardo Stemmer
Presidente

Prof. Dr. Luiz Carlos S. Góes

Prof. Dr. Ing. Paulo Martins Engel

Prof. Dr. Sc. Raul Guenther

Prof. Dr. Edson Roberto De Pieri

À Karen que sempre me lembra que posso melhorar como pessoa.

Agradecimentos

Primeiramente gostaria-se de agradecer aos companheiros de labuta na operação com o robô Inter/Scara utilizado neste trabalho: Carlos César Bier, Cesar Augusto Montejumas, Julio Feller Golin e Alejandro Garcia Ramirez. Outros agradecimentos não menos importantes vão para meus colegas pós-graduandos pelos anos de companheirismo e amizade, entre alguns que poderia citar estão o Sumar, Jorge, Jerusa, "C.Bradão", João, Otacílio, Leandro, Michelle, Torrico, Karina, Lau, Felipe, "C-K.oliver', Max, Sônia, Werner, Wilson Costa. Em especial aos colegas Antonio Eduardo Carrilho da Cunha e Marcos Vallim – companheiros de considerações práticas e filosóficas. Agradecimentos também aos professores Marcelo Ricardo Stemmer (pela sua louvável pessoa), prof. Raul Guenther (pela dedicação e excelentes aulas na área de robôs manipuladores) e ao prof. Edson Roberto de Piere (pela paciência demonstrada nas correções que realizou). E por fim, gostaria de agradecer à Universidade de Passo Fundo (entre eles gostaria de agradecer mais fortemente ao prof. Carlos Allan Caballero Peterson) pelo suporte financeiro oferecido para a execução desta tese sem me esquecer também da agência de fomento à pesquisa, CNPq, que contribuiu no primeiro ano de execução desta tese.

Resumo da Tese apresentado à UFSC como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

Controle Neural de Posição e Força em Manipuladores Robóticos

Fernando Passold

Dezembro/2003

Orientador: Dr. Ing. Marcelo Ricardo Stemmer
Área de Concentração: Automação de Sistemas
Palavras-chave: Robôs Manipuladores, Controle de Posição, Controle de Força, Redes Neurais Artificiais
Número de Páginas: 319.

Este trabalho explorou técnicas de inteligência computacional, mais especificamente o uso de redes neurais artificiais no controle de posição e de força de um robô manipulador. Foi basicamente explorada a arquitetura de aprendizagem por retropropagação de erros na qual uma rede neural trabalha em "paralelo" com um controlador convencional. A vantagem no uso desta arquitetura é que a mesma permite aproveitar o controlador convencional normalmente já existente para controle de posição e de força de um robô manipulador sem que seja necessário modificar o algoritmo original do controlador convencional que já vinha sendo empregado. Neste trabalho, resolveu-se cunhar esta arquitetura de controle mesclando um controlador convencional com uma rede neural como "controlador integrado". Este trabalho pode ser dividido em duas partes: experimentação do controlador integrado para controle de 1) posição e 2) de força usando redes neurais multicamadas de *perceptrons* e redes neurais de função de base radial. Como o controle de força normalmente é realizado ao menos num dos graus de liberdade do robô manipulador; nos graus de liberdade restantes, permanece o controle de posição, foi adotada a estrutura de controle híbrido comum da área de controle de robôs manipuladores. Esta técnica implica em realizar controle de força nas direções nas quais o movimento é restringido e controle de posição nas direções ortogonais restantes. Para cada uma destas malhas foi previsto o seu controlador integrado. Estas abordagens foram testadas na prática sobre o robô manipulador Inter presente no Laboratório de Automação Industrial de Universidade Federal de Santa Catarina. Este robô segue a configuração conhecida como SCARA, disponibilizando 4 graus de liberdade e está equipado com um sensor de força. Resultados experimentais são apresentados e conclusões podem ser extraídas tanto da parte de controle de posição quanto da parte de controle de força. O controle de posição foi resolvido com sucesso mas não pôde ser encontrada uma solução conclusiva para o caso do controle integrado de força que exige um estudo mais aprofundado sobre os dados de entrada que devem ser entregues à esta rede neural assim como também, outras arquiteturas para controle integrado de força.

Abstract of Thesis presented to UFSC as partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering

Neural Position and Force Control of Robotics Manipulators

Fernando Passold

December/2003

Advisor: Dr. Ing. Marcelo Ricardo Stemmer
Area of Concentrations: System Automation
Keywords: manipulator, neural networks, position control, force control
Number of pages: 319.

This work has explored computational intelligent technics, specifically the use of artificial neural networks applied to a complex case of control: position and force control of an real manipulator robot. It was applied neural controllers based on the feedback error learning architecture over which a neural network performs in parallel with an conventional controller. The main advantage of this architecture is that it allows the use of the previous conventional controller already existed in the system without require any modification of its internal algorithm. In this work, this combination between neural network plus conventional controller was named "integrate control"'. This work could be divide in two parts: one related to position control and another related with force control, both of them using Multilayer of Perceptrons networks (MLP's) and Radial Basis Function networks (RBF's). As force control is normally applied at least at one degree of freedom of the robot, while in the other degrees remains position control; it was adopted the hybrid force/position control commonly used for manipulators robots. This approach consist in perform force control in restricted directions while perform position control in the ortogonal directions that remains. For each feedback it was developed its integrate control. The technics proposed by this work was experimented over an real scara robot type (4 degrees of freedom) presented in the Laboratory of Industrial Automation of the Federal University of Santa Catarina. It also was equipped with a force sensor. Practical results and conclusions could been extracted for both of the integrate controllers proposed. Different types of conventional controllers were evaluated with the integrate controllers proposed in this work. The integrate positional control performed very successfully but the force integrate control could not be reach conclusive results. It was observed practical problems and more deepen investigation of force control using artificial neural networks remais.

Sumário

Lista de Figuras **xix**

Lista de Tabelas **xxv**

Simbologia Adotada **xxix**

1 Introdução **1**

1.1 Controle Convencional de Força 3

1.2 Controle Avançado de Força 4

1.3 Controle Inteligente de Posição/Força 5

1.4 Objetivos e Contribuições deste trabalho 5

1.5 Organização deste Documento 7

2 Controle de Posição/Força de Robôs Manipuladores **11**

2.1 Introdução 11

2.2 Controle de Posição 13

2.2.1 Controle PD com Compensação de Gravidade 14

2.2.2 PD no Espaço Operacional 15

2.2.3 Controlador por Jacobina Inversa, J_A^{-1} 16

2.2.4 Controladores de Posição no Espaço de Juntas $\times$ Espaço Operacional . 16

2.3 Controle de Força 17

2.3.1 Restrições Naturais e Artificiais 18

2.3.2 Controle de Força por Rigidez 21

2.3.3 Controle de Força por Impedância 24
2.3.4 Controle Híbrido de Posição/Força 27
2.3.5 Controle Híbrido por Impedância 29
2.3.6 Comparação dos métodos convencionais para controle de força 30
2.4 Controladores "Avançados" 30
2.5 Controle Adaptativo de Força 32
2.6 Controle Robusto de Força 36
2.7 Controle de Força por Algoritmos de Aprendizado 38
2.8 Resumo de Controle Avançado de Força 38
2.9 Conclusão 38

3 Controle Inteligente **41**
3.1 Introdução 41
3.2 Redes Neurais em Controle 44
3.2.1 Introdução 44
3.2.2 RNs em Controle (Geral) 47
3.2.3 Redes Multicamadas Perceptron 48
3.2.4 Redes RBF 53
3.2.5 Tabela Resumo de Redes Neurais 56
3.2.6 Detalhes de Implementação com Redes Neurais 56
3.2.7 Estabilidade de neurocontroladores 58
3.2.8 Reconhecimento de Padrões em Controle 58
3.3 Estruturas Típicas de Controladores Neurais 59
3.3.1 Arquitetura de Aprendizado Generalizado 60
3.3.2 Arquitetura de Aprendizado Especializado 60
3.3.3 Controlador Neural *Feedforward* 61
3.3.4 Controlador baseado no aprendizado do erro de realimentação 61
3.3.5 Controlador Auto-Ajustável baseado numa Rede Neural 62
3.3.6 Rede Neural para Identificar Sistemas 62
3.3.7 Controladores Adaptativos e Redes Neurais 66
3.3.8 Resumo de controladores baseados em redes neurais 68
3.4 Aplicações de RNs em Robótica de Manipulador 70

3.4.1 Modelagem da Cinemática Inversa 71
3.4.2 Controlador neural de posição baseado em realimentação do erro para um robô SCARA 73
3.4.3 Experiência de um Controlador (de Posição) Neural Adaptativo, no Espaço Cartesiano 77
3.4.4 Controle Não Linear Baseado em Modelagem usando Redes Neurais . 80
3.4.5 Rede Neural numa estrutura NARMA 81
3.4.6 Proposta de um controlador neural robusto de posição 82
3.4.7 Controle de Força/Posição de um Robô SCARA usando RNs 84
3.5 Controladores *Fuzzy* 92
3.5.1 Controlador *Fuzzy* – Introdução 92
3.5.2 Controladores PID *fuzzy* 94
3.5.3 Controlador *Fuzzy*-PI + Convencional D 94
3.5.4 Controlador *Fuzzy*-PD + Convencional I 95
3.5.5 Controlador *Fuzzy*-PD + *Fuzzy*-I 96
3.5.6 Controlador *Fuzzy*-PI + *Fuzzy*-PD 96
3.5.7 Controlador Adaptativo *Fuzzy* com Aprendizado Neural 96
3.5.8 Controle de Força *Fuzzy* para Robôs Industriais 98

4 Descrição do Robô e dos Controladores Propostos 101
4.1 Introdução 101
4.2 Descrição do robô Inter/Scara 101
4.3 Rotina de Controle 107
4.4 Controladores Convencionais de Posição 110
4.4.1 Controlador PD no Espaço de Juntas 110
4.4.2 Controlador PID no Espaço de Juntas 112
4.4.3 Controlador por Modos Deslizantes no Espaço de Juntas 112
4.4.4 Controle por Jacobiana Inversa no Espaço Operacional 113
4.4.5 Controle PD no Espaço Operacional (PD-Op) 113
4.4.6 Controlador PID no Espaço Operacional (PID-Op) 114
4.5 Controlador Integrado de Posição 114
4.5.1 Rede MLP para controle de posição 118

4.5.2 Rede RBF para controle de posição 120
4.6 Controle Híbrido de Posição/Força 125
4.7 Controladores Convencionais de Força 126
4.7.1 Controlador Proporcional de Força 128
4.7.2 Controlador PI de Força 128
4.8 Controladores Integrados de Força 128
4.8.1 Primeira Versão da RN para Controle de Força (RNf-1) 129
4.8.2 Segunda Versão da RN para Controle de Força (RNf-2) 129
4.8.3 Terceira Versão da RN para Controle de Força (RNf-3) 130
4.8.4 Quarta Versão da RN para Controle de Força (RNf-4) 131
4.9 Índices de Desempenho 132
4.9.1 Índices em relação aos Erros de seguimento de trajetória 132
4.9.2 Índices de Desempenho em relação ao Esforço de Controle realizado . 133

5 Resultados dos Controladores de Posição 137
5.1 Introdução 137
5.2 Ajuste do PID 137
5.3 Ajuste das Redes Neurais 138
5.3.1 Parâmetros de aprendizado para redes MLP 139
5.3.2 Parâmetros de aprendizado para redes RBF 140
5.4 Ensaios Propostos 140
5.5 Ensaios com Trajetórias Simples (sem perturbação) 141
5.5.1 Testes com a Trajetória "A" 141
5.5.2 Testes com a Trajetória "B" 166
5.6 Ensaio de Rejeição à Perturbação 192
5.7 Discussão Final 210

6 Resultados para Controle de Força/Posição 215
6.1 Introdução 215
6.2 Resultados dos Testes com Controladores de Posição no Espaço Operacional . 216
6.2.1 Trajetória do Teste 216
6.2.2 Teste do Controlador por Jacobiana Inversa (J_A^{-1}) 217

6.2.3 Testes do Controlador PD-Op . . . 218
6.2.4 Testes do Controlador PID-Op . . . 219
6.2.5 Erros em XY dos Controladores Op . . . 221
6.2.6 Índices de Desempenho dos Controladores Op . . . 222
6.2.7 Discussão sobre Controladores de Posição no Espaço Operacional . . . 222
6.3 Controladores Convencionais de Força . . . 223
6.3.1 Testes Propostos . . . 224
6.3.2 Teste da Estratégia de Contato . . . 224
6.3.3 Teste sobre Borracha . . . 228
6.3.4 Teste sobre Isopor . . . 233
6.3.5 Teste sobre o Plano Inclinado de Madeira . . . 237
6.3.6 Discussão a respeito dos Controladores Convencionais de Força . . . 239
6.4 Controladores Neurais de Força . . . 240
6.4.1 RN para Controle de Força (RNf-1) . . . 240
6.4.2 Reescalonando informação de entrada para redes RNf-1 . . . 249
6.4.3 Conclusões gerais com relação às redes RNf-1 . . . 254
6.4.4 Segunda Versão da RN para Controle de Força (RNf-2) . . . 254
6.4.5 Terceira Versão da RN para Controle de Força (RNf-3) . . . 257
6.4.6 Conclusões preliminares à respeito da rede RNf-3 . . . 259
6.4.7 Quarta Versão da RN para Controle de Força (RNf-4) . . . 260
6.4.8 Conclusões preliminares à respeito da rede NNf-4 . . . 261
6.5 Discussão à respeito dos Controladores Neurais de Força . . . 262
6.6 Discussão Final sobre Controle de Força . . . 264

7 Conclusões e Perspectivas **267**
7.1 Pontos Abordados . . . 267
7.2 Principais Contribuições . . . 269
7.3 Conclusões sobre Controle de Posição Integrado . . . 271
7.4 Conclusões sobre Controle de Força Integrado . . . 273
7.5 Limitações e Pontos não abordados . . . 275
7.6 Sugestões de Trabalhos Futuros . . . 278

A Transformações de Coordenadas **287**

A.1 Do Espaço de Juntas para o Espaço Operacional 287

A.1.1 Cinemática Direta 287

A.1.2 Jacobiano Analítico 288

A.1.3 Derivada do Jacobiano Analítico 289

A.2 Do Espaço Operacional para o Espaço de Juntas 290

A.2.1 Inversa do Jacobiano Analítico 290

A.2.2 Aceleração do Espaço Operacional para o Espaço de Juntas 293

A.3 Simulações 293

B Controladores PID Digitais **295**

B.1 Formas de Implementação 295

B.2 Ajuste de Controladores PID 296

C Algoritmo *back-propagation* expandido com termo *momentum* **297**

C.1 Introdução 297

C.2 O Algoritmo *Back-propagation* 298

Lista de Figuras

2.1 Controlador PD no espaço operacional 16
2.2 Estimando força de contato contra o meio. 18
2.3 Exemplo de movimento restringido 21
2.4 Controle (explícito) de rigidez. 22
2.5 Controle (implícito) de rigidez. 23
2.6 Controle típico por impedância. 25
2.7 Controle por impedância baseado em posição. 26
2.8 Bloco básico do robô manipulador em contato com o meio. 28
2.9 Estrutura básica do controle híbrido de posição/força. 28
2.10 Controle Híbrido por impedância. 29
2.11 Controle Adaptativo genérico de Força. 34
2.12 Exemplo de controlador de força implícito por impedância ativa. 35
2.13 Controle Robusto genérico de Força. 36
2.14 Trajetória do erro para controle robusto. 37
2.15 Algoritmo de aprendizado para controle de força. 38

3.1 Neurônio artificial simplificado do tipo *perceptron*. 49
3.2 Rede MLP de 2 camadas invisíveis. 49
3.3 Funções de ativação típicas de redes MLP. 51
3.4 Estrutura básica de uma rede RBF. 53
3.5 Função Gaussiana. 54
3.6 Arquitetura de aprendizado generalizado para uma RN. 60

3.7 Arquitetura de aprendizado especializado para uma RN. 61
3.8 Controlador neural *feedforward*. 61
3.9 Controlador baseado no aprendizado do erro de realimentação. 61
3.10 Controlador auto-ajustável baseado em rede neural. 62
3.11 RN para identificação de modelo dinâmico de um sistema. 62
3.12 Rede Neural para modelagem por MA. 64
3.13 Rede Neural para modelagem AR. 64
3.14 Controlador a modelo interno (IMC). 66
3.15 Arquitetura baseada na realimentação do erro (FELC). 74
3.16 Arquitetura avançada sendo proposta – EFELC. 76
3.17 MRAC neural. 81
3.18 Controlador adaptativo neural genérico. 82
3.19 Controlador neural de posição proposto por Battistela (1999). 85
3.20 Controlador híbrido neural de força/posição (Battistela(1999)). 89
3.21 Controlador geral por lógica *fuzzy*. 92
3.22 Funções de pertinência para $e[k]$, $de[k]$ e $u[k]$. 93
3.23 Controlador *Fuzzy*-PI + Convencional D. 94
3.24 Controlador *fuzzy*-PD + *fuzzy*-I. 95
3.25 Controlador *Fuzzy*-PD + *Fuzzy*-I. 96
3.26 Controlador *Fuzzy*-PD + *Fuzzy*-PI. 97

4.1 Robô Inter/Scara e seu sistema de coordenadas. 102
4.2 Sensor de força JR3 instalado no robô. 103
4.3 Ferramentas implementadas para o robô. 103
4.4 Figura ilustrando ambiente de operação do XO/2. 105
4.5 Diagrama em blocos do controle em tempo-real feito no robô Inter/Scara. . . 108
4.6 Controladores de posição implementados no robô Inter/SCARA. 109
4.7 Saturador para ação integral excessiva. 112
4.8 Controlador integrado de posição implementado 115
4.9 Composição dos torques finais enviados às juntas do robô. 116
4.10 Mecanismo de "desconexão" da rede em caso de problemas. 116
4.11 Fluxo de dados envolvendo rede MLP para controle de posição. 117

4.12 Fluxo de dados envolvendo rede RBF para controle de posição. 117
4.13 Exemplo de distribuição de funções de base radial para dados da junta 2. . . 122
4.14 Estrutura de rede neural RBF para controle de posição ($m_1 = 5$ neste caso). . . 124
4.15 Controle híbrido implementado no XO/2. 126
4.16 Diagrama em blocos de controle de força explícito 126
4.17 Dados de entrada e de saída da rede RNf-1. 129
4.18 Dados de entrada e de saída para RNf-2. 130
4.19 Dados de entrada e de saída para RNf-3. 131
4.20 Dados de entrada e de saída para RNf-4. 131

5.1 Sintonia dos controladores PID para cada uma das juntas do robô. 139
5.2 Trajetória executada para o Movimento ″A″. 142
5.3 Perfis da trajetória do Movimento ″A″. 144
5.4 Outras trajetória previstas (mas não realizadas). 145
5.5 Erros de seguimento de trajetória para a Junta 0 (Movimento ″A″). 146
5.5 (Cont.) Erros de seguimento de trajetória para a Junta 0 (Movimento ″A″). . . 147
5.6 Erros de seguimento de trajetória para a junta 1 (Movimento ″A″). 148
5.6 (Cont.) Erros de seguimento de trajetória para a Junta 1 (Movimento ″A″). . . 149
5.7 Erros de seguimento de trajetória para a Junta 2 (Movimento ″A″). 150
5.7 (Cont.) Erros de seguimento de trajetória para a Junta 2 (Movimento ″A″). . . 151
5.8 Erros de seguimento de trajetória para a Junta 3 (Movimento ″A″). 152
5.8 (Cont.) Erros de seguimento de trajetória para a Junta 3 (Movimento ″A″). . . 153
5.9 Torques enviados à junta 0 (Movimento ″A″). 154
5.9 (Cont.) Torques enviados à junta 0 (Movimento ″A″). 155
5.10 Torques enviados à junta 1 (Movimento ″A″). 156
5.10 (Cont.) Torques enviados à junta 1 (Movimento ″A″). 157
5.11 Torques enviados à junta 2 (Movimento ″A″). 158
5.11 (Cont.) Torques enviados à junta 2 (Movimento ″A″). 159
5.12 Torques enviados à junta 3 (Movimento ″A″). 160
5.12 (Cont.) Torques enviados à junta 3 (Movimento ″A″). 161
5.13 Trajetória executada para o Movimento ″B″. 167
5.14 Perfis da trajetória do Movimento ″B″. 168

5.15 Erros de seguimento de trajetória para a Junta 0 (Movimento ″B″). 170
5.15 (Cont.) Erros de seguimento de trajetória para a Junta 0 (Movimento ″B″). . . 171
5.16 Erros de seguimento de trajetória para a Junta 1 (Movimento ″B″). 172
5.16 (Cont.) Erros de seguimento de trajetória para a Junta 1 (Movimento ″B″). . . 173
5.17 Erros de seguimento de trajetória para a Junta 2 (Movimento ″B″). 174
5.17 (Cont.) Erros de seguimento de trajetória para a Junta 2 (Movimento ″B″). . . 175
5.18 Erros de seguimento de trajetória para a Junta 3 (Movimento ″B″). 176
5.18 (Cont.) Erros de seguimento de trajetória para a Junta 3 (Movimento ″B″). . . 177
5.19 Torques enviados à junta 0 (Movimento ″B″). 178
5.19 (Cont.) Torques enviados à junta 0 (Movimento ″B″). 179
5.20 Torques enviados à junta 1 (Movimento ″B″). 180
5.20 (Cont.) Torques enviados à junta 1 (Movimento ″B″). 181
5.21 Torques enviados à junta 2 (Movimento ″B″). 182
5.21 (Cont.) Torques enviados à junta 2 (Movimento ″B″). 183
5.22 Torques enviados à junta 3 (Movimento ″B″). 184
5.22 (Cont.) Torques enviados à junta 3 (Movimento ″B″). 185
5.23 Fixação dos cabos de alimentação dos motores do Inter/Scara. 192
5.24 Trajetória executada com a perturbação presente. 193
5.25 Trajetória executada com a perturbação presente. 193
5.26 Perfis da trajetória com a perturbação presente (espaço op.). 194
5.27 Perfis da trajetória com a perturbação presente (espaço de juntas). 195
5.28 Erros de seguimento de trajetória em XY (teste com perturbação). 197
5.28 (Cont.) Erros de seguimento de trajetória em XY (teste com perturbação). . . . 198
5.28 (Cont.) Erros de seguimento de trajetória em XY (teste com perturbação). . . . 199
5.29 Torques enviados à junta 0 referentes ao teste com perturbação. 201
5.29 (Cont.) Torques enviados à junta 0 referentes ao teste com perturbação. 202
5.29 (Cont.) Torques enviados à junta 0 referentes ao teste com perturbação. 203
5.30 Torques enviados à junta 1 referentes ao teste com perturbação. 204
5.30 (Cont.) Torques enviados à junta 1 referentes ao teste com perturbação. 205
5.30 (Cont.) Torques enviados à junta 1 referentes ao teste com perturbação. 206
5.31 Relação custo × benefício dos controladores de posição 214

6.1 Trajetória de teste prevista para os controladores no espaço operacional. . . . 217
6.2 Erros do teste para o controlador por J_A^{-1}. 218
6.3 Gráficos dos erros de seguimento de trajetória para o PD-Op. 219
6.4 Erros de seguimento de trajetória para PID-Op. 220
6.5 Gráficos de $\tilde{x} \times \tilde{y}$ ao longo do tempo. 221
6.6 Testes previsto para os controladores de força. 225
6.7 Trajetória do teste da Estratégia de Contato. 226
6.8 Resultados do teste da Estratégia de Contato. 227
6.9 "Teste do dedinho". 227
6.10 Dados do teste sobre a borracha. 228
6.11 Erros de Posição para o teste sobre a borracha. 229
6.12 Profundidade dos contatos para teste sobre borracha. 230
6.13 Torques aplicados à junta 2 no teste sobre a borracha. 230
6.14 Forças de contato desenvolvidas no teste sobre borracha. 231
6.15 Estimativa da constante de rigidez da borracha. 231
6.16 Dados da segunda trajetória de teste sobre o isopor. 233
6.17 Profundidade dos contatos para teste sobre isopor. 234
6.18 Forças de contato desenvolvidas no teste sobre o isopor. 235
6.19 Torques aplicados à junta 2 no teste sobre o isopor. 236
6.20 Estimativa da constante de rigidez do isopor. 236
6.21 Trajetória executada sobre o plano inclinado de madeira. 237
6.22 Resultados do teste sobre o plano inclinado. 238
6.23 Controle de força sobre madeira (plano inclinado), com $K_p = 50$ e $K_i = 0.5$. . 239
6.24 Trajetória executada sobre a borracha. 241
6.25 Teste do controlador de força integrado RNf-1(MLP) ($\eta = 0.035$ e $\alpha = 0.5$). . . 242
6.26 Fluxo de dados passando pela rede RNf-1(MLP). 243
6.27 Torques gerados pela rede RNf-1(MLP). 243
6.28 Teste do controlador de força integrado RNf-1(MLP) ($\eta = \mathbf{0.05}$ e $\alpha = 0.5$). . . 244
6.29 Resultados teste do controlador RNf-1 (RBF), com $\eta = 0.005$ e $\alpha = 0.5$. 246
6.30 Fluxo de dados sobre a rede RNf-1 (RBF). 247
6.31 Torques gerados no teste do controlador RNf-1 (RBF). 247

6.32 Teste do controlador RNf-1 (MLP), com entradas re-escalonadas. 251
6.33 Resultado do ganho proporcional elevado do PI + RNf-1 (MLP). 252
6.34 Teste do controlador RNf-2. 255
6.35 Torques gerados pelo controlador RNf-2. 256
6.36 Teste do controlador RNf-3. 257
6.37 Estado energético da rede RNf-3. 259
6.38 Resultado do teste com RNf-4, com $lr = 0.02$ e $mom = 0.5$. 260
6.39 Teste com trajetória repetida para controlador RNf-4. 261
6.40 Pacotes de dados de entrada usados para as redes de controle de força. 262

7.1 Exemplo de efetuador final para tarefa de polimento de peças. 279
7.2 Modelo proposto de novo controlador integrado usando RN e baseado em MRAC. 284
7.3 Dados de entrada e de saída para proposta da nova rede RN-fx. 285

A.1 Diagrama de blocos mostrando processamento da cinemática direta 288
A.2 Teste dos blocos de cinemática direta e inversa no MATLAB 294

C.1 Funções de ativação típicas de redes MLP. 300

Lista de Tabelas

2.1 Métodos de controle de força convencionais. 31
2.2 Métodos avançados de controle de força. 39

3.1 Características de redes neurais MLP e RBF. 57
3.2 Exemplo de base de regras *fuzzy*. 93

4.1 Capacidade de carregamento do sensor de força JR3. 103
4.2 Ganhos (originais) do Controlador PD de posição no espaço de juntas. 111
4.3 Valores máximos e mínimos relacionados com as juntas do robô. 118
4.4 Fatores escala empregados com as redes neurais. 119
4.5 Funções gaussianas dos vetores de entrada da rede RBF(5). 123

5.1 Ajuste dos controladores convencionais de posição. 138
5.2 Dados da trajetória executada para Movimento ″A″. 143
5.3 Principais ensaios realizados para o Movimento ″A″. 145
5.4 $\tilde{q}_b$ relativos aos Movimento ″A″. 161
5.5 $MAX\{\tilde{q}\}$ relativos ao Movimento ″A″. 162
5.6 $IAE\{\tilde{q}\}$ referentes ao Movimento ″A″. 162
5.7 $MED\{\tilde{q}\}$ referentes ao teste para Movimento ″A″. 162
5.8 $MSE\{\tilde{q}\}$ referentes ao teste para Movimento ″A″. 163
5.9 $VAR\{\tilde{q}\}$ relacionados ao teste para Movimento ″A″. 163
5.10 $MAX\{\tau\}$ relacionados ao teste para Movimento ″A″. 163
5.11 $MED\{\tau\}$ relacionados ao teste para Movimento ″A″. 164

5.12 $VARi\{\tau\}$ referentes ao teste para Movimento ″A″. 164
5.13 Dados da trajetória executada para Movimento ″B″. 167
5.14 Principais ensaios realizados para o Movimento ″B″. 169
5.15 $\tilde{q}_b$ relativos aos Movimento ″B″. 186
5.16 $MAX\{\tilde{q}\}$ relativos ao Movimento ″B″. 186
5.17 $IAE\{\tilde{q}\}$ referentes ao Movimento ″B″. 186
5.18 $MED\{\tilde{q}\}$ referentes ao teste para Movimento ″B″. 187
5.19 $MSE\{\tilde{q}\}$ referentes ao teste para Movimento ″B″. 187
5.20 $VAR\{\tilde{q}\}$ relacionados ao teste para Movimento ″B″. 188
5.21 $MAX\{\tau\}$ relacionados ao teste para Movimento ″B″. 188
5.22 $MED\{\tau\}$ relacionados ao teste para Movimento ″B″. 188
5.23 $VARi\{\tau\}$ referentes ao teste para Movimento ″B″. 189
5.24 Operações em ponto flutuante necessárias para processar as redes [Flops]. . . 191
5.25 Dados da trajetória com perturbação. 194
5.26 Ensaios relativos à perturbação. 196
5.27 $MSE\{\tilde{d}\}$) encontrado nos testes de rejeição à perturbação. 207
5.28 Tempos de processamento para controladores de posição. 214

6.1 Dados da trajetória usada para teste dos controladores-Op. 217
6.2 Ganhos do PD-Op 218
6.3 Ganhos PD-Op 219
6.4 Indices de desempenho atingidos pelos controladores-Op. 222
6.5 Dados da primeira trajetória do teste sobre o isopor. 233
6.6 Trajetória usada sobre a superfície de borracha. 241
6.7 Fatores escala originalmente empregados para as RNf. 245
6.8 Distribuição das funções gaussianas para a RNf-1 (RBF). 248
6.9 Novos fatores escala empregados para as RNs-força. 249
6.10 Redistribuição das funções gaussianas para as RNf-1 (RBF). 250
6.11 Estados atingidos pela rede RNf-1 (RBF) com entradas re-escalondas. 253
6.12 Resultados obtidos com a trajetória de ida (RNf-2). 256
6.13 Indices de desempenho para trajetória de (b) para (a). 258

B.1 Relações de Ziegler-Nichols para ajuste de controladores PID. 296

Simbologia Adotada

Símbolos Empregados		
Símbolo	Descrição/Observação	Dimensão
n_q	numero de juntas do robô em questão. $n_q = 4$ para o caso de um robô do tipo SCARA (4 graus de liberdade).	*escalar*
q	vetor das posições angulares de cada junta do robô. $q = \left[\begin{array}{cccc} q_0 & q_1 & q_2 & q_3 \end{array} \right]^T = \left[\begin{array}{cccc} \text{rad} & \text{rad} & \text{m} & \text{rad} \end{array} \right]^T$.	n_q
$\dot{q}$	vetor das velocidades angulares de cada junta do robô.	n_q
$\ddot{q}$	vetor das acelerações angulares desenvolvidas pelas juntas do robô.	n_q
q_d	vetor contendo as posições angulares desejadas para cada junta do robô.	n_q
$\tilde{q}$	erro de posicionamento angular das juntas do robô. $\tilde{q} = q_d - q$	n_q
$\dot{q}_d$	vetor contendo as velocidades angulares desejadas para cada junta do robô.	n_q
$\dot{\tilde{q}}$	vetor de erro de velocidade angular das juntas do robô.	
D.O.F	graus de liberdade do robô.	
n_x	número de graus de liberdade do robô em questão (= D.O.F.). $n_x = 4$ para o caso do robô tipo SCARA.	
x	vetor de posições no espaço operacional (cartesiano) do robô. $x = \left[\begin{array}{cccc} x & y & z & \theta \end{array} \right]^T$, onde θ corresponde à orientação final do robô em relação à seu espaço operacional (em [rad])	n_x
$\tilde{x}$	vetor do erro de posição no espaço operacional do robô.	n_x
$\dot{x}$	vetor das velocidades lineares desenvolvidas pelo robô em relação à cada um dos seus D.O.F.	n_x
$\ddot{x}$	vetor das acelerações lineares desenvolvidas pelo robô em relação a cada um de seus D.O.F.	n_x
x_d	vetor das posições desejadas para o robô dentro de seu espaço operacional.	n_x
$\dot{x}_d$	vetor das velocidades lineares desejadas para o robô dentro do seu espaço operacional.	n_x
f_{Sens}	informação bruta vinda do sensor de força do robô. $f_{Sens} = \left[\begin{array}{cccccc} f_x & f_y & f_z & \mu_x & \mu_y & \mu_z \end{array} \right]^T$	6

Continua na próxima página

Continuação da página anterior

Símbolo	Descrição/Observação	Dimensão
	onde: f_x, f_y, f_z são forças nas direções XYZ do sensor; μ_x, μ_y, μ_z são momentos gerados em relação à cada eixo do sensor.	
f_{THR}	limiar de força desejada para manter contato.	1
h	vetor das forças interagindo com robô (no espaço operacional).	3
h_d	vetor da força de contato desejada contra o meio (pressão do contato).	3
$B(q)$	matriz de inércia do robô manipulador.	$n_q \times n_q$
$C(q, \dot{q})$	vetor dos termos de Corioli centrífugos do sistema.	
F	termos relacionados aos atritos envolvidos no sistema.	n_q
$n(q, \dot{q})$	matriz correspondente à compensação dinâmica do robô.	
$u(t)$	ação de controle no domínio tempo (sinal contínuo no tempo).	
$u[k]$	ação de controle discreto (sinal discretizado no tempo – controle digital).	
J	matriz Jacobiana do robô manipulador.	$n_x \times n_x$
J_A	matriz Jacobiana Analítica do robô.	$n_x \times n_x$
J_A^{-1}	matriz inversa da matriz Jacobiana Analítica.	
J_A^T	transposta da matriz Jacobiana Analítica.	
$\dot{J}_A$	derivada da matriz Jacobiana Inversa.	
$K(q)$	transformação relativa à cinemática direta.	
dx	vetor que corresponde ao *offset* provocado na posição final do robô no espaço operacional pela inclusão de uma ferramenta (*tool tip*) ao mesmo.	3
$K_{Offset}(q, dx)$	Cálculo da cinemática direta incluindo *offset* provocado pela inclusão da ferramenta no robô manipulador.	
τ_{FB}	vetor de torque gerado pelo controlador convencional de posição no espaço de juntas	n_x
τ_{FF}	vetor de torque relativo à compensações dinâmicas (*Feed-Forward*).	n_q
τ_{NN}	vetor de torque gerado pela rede neural para controle de posição	n_q
τ_{Force}	vetor de torque gerado pelo controlador convencional para controle de força	$\leq n_x$
$\tau_{NNForce}$	vetor que torque gerado pela rede neural para controle de força	$\leq n_x$
τ_{Final}	vetor dos torques finais enviados a cada uma das juntas do robô	n_q
S	Matriz de seleção (referente à controle híbrido de posição/força).	n_x
P	abreviatura adotada para controlador convencional do tipo Proporcional somente.	
PD	abreviatura adotada para controlador Proporcional-Derivativo.	
PID	abreviatura adotada para controlador Proporcional-Integral-Derivativo.	
K_u	vetor do ganho máximo proporcional que pode ser aplicado à cada junta do robô (limiar da estabilidade ou "*ultimate gain*").	n_q
T_u	vetor referente ao período de oscilação de uma junta (usado apenas para sintonizar controladores).	n_q
K_p	vetor de ganhos para ação proporcional.	n_q
K_{fp}	vetor do ganho proporcional para controlador convencional de força.	$\leq n_x$
K_i	vetor de ganhos para ação integral.	
K_{fi}	vetor dos ganhos para ação integral para controlador de força	$\leq n_x$
K_d	vetor de ganhos para ação derivativa.	

Continua na próxima página

Continuação da página anterior		
Símbolo	Descrição/Observação	Dimensão
RN	abreviatura adotada para Redes Neurais artificiais.	
MLP	refere-se a uma rede do tipo multicamadas de *perceptrons*.	
RBF	refere-se a uma rede de Funções de Base Radial.	

Capítulo 1

Introdução

As aplicações mais comumente encontradas para robôs industriais envolvem tarefas como tirar e colocar peças (*pick and place*) que requerem pouca ou nenhuma interação do robô com o meio (McClamroch, 1993; Bier and Guenther, 2000). Dados de 1999 publicados pelo Instituto de Robótica da Universidade de Carnegie Mellon[1] revelavam que aproximadamente 90% dos robôs industriais eram empregados em tarefas de solda ponto à ponto, movimentação de peças, pintura e solda à arco – operações que não exigem contato do efetuador (*end effector*) do robô com o meio e que menos de 7% das aplicações faziam uso de um sensor de força para controle de força de robô interagindo com o meio para operações como manipulação de ferramentas ou peças em tarefas de montagens, polimento de peças e remoção de material.

O problema começa a ficar mais complexo quando se exige que o manipulador siga uma determinada trajetória (de posição) desenvolvendo uma alta velocidade com baixo erro de seguimento da trajetória, como no caso de solda à plasma ou corte à laser ao invés de simplesmente atingir determinados pontos fixos, como no caso de movimentação de peças ou operações de soldagem à ponto (Asada and Slotine, 1986).

Mas as tarefas mais desafiantes para robôs manipuladores freqüentemente envolvem alguma especificação de interação com o meio (McClamroch and Wang, 1988).

As tarefas que requerem que um robô manipulador interaja com seu ambiente obrigatoriamente exigem que o robô, além de executar posições predefinidas, desenvolva forças

[1]Relatório disponível em 1999, publicado por D. A. Vicent, intitulado "*Robots Help Manufacturing Companies to Innovate, Automate, and Compete*", no site da *Robotics Institute* – http://www.ri.cmu.edu/home.html (parte da faculdade de ciências da computação da Univesidade de Carnegie Mellon).

necessárias para superar a rigidez do meio com o qual faz contato ou para acompanhar (contornar) uma peça (Zeng and Hemani, 1997; McClamroch, 1993; Asada and Slotine, 1986). Exemplos deste tipo de tarefa incluem operações de inserção e montagem de peças, movimentos de manivela, polimento, desbaste, batidas, amoladura de peças, seguimento de contornos (sobre este item específico ver (Bier and Guenther, 2000)) (McClamroch, 1993; Zeng and Hemani, 1997). Vicent (1999) relata que apenas 7% dos robôs industriais são utilizados em tarefas como remoção de materiais (tarefas de acabamento de peças) e montagens, tarefas nas quais o efetuador realiza contato com o meio.

Para estes tipos de tarefa, um controle de posição puro não é recomendado, uma vez que pode resultar em falta de contato com a peça ou o meio, dado o erro acumulado ao longo do tempo em trajetórias de posição, ou mesmo a peça pode se danificar. Normalmente se deseja uma espécie de controle de força na direção ortogonal à peça ou meio de forma a também garantir que o contato do efetuador com a peça seja mantido.

Na maioria das aplicações industriais, a tarefa a ser executada é descrita apenas em termos de metas de posicionamento (McClamroch, 1993). Os controladores industriais de robôs normalmente tratam as forças originadas do contato com o meio como perturbações que devem ser rejeitadas.

Já o controle de força envolve a integração de algumas tarefas: geração de trajetórias (de posição e de força); realimentação de posição e de força e modificação das trajetórias. Exige ainda um completo detalhamento da situação do contato (especificação da forma como se espera fazer o contato), para que estratégias e trajetórias possam ser efetivamente planejadas e para que se saiba que dados devem ser realimentados na malha do controlador para esta tarefa (Whitney, 1987).

O controle de força também exige respostas estáveis do robô. Isto é, usualmente requer a filtragem de sinais indesejados (ruídos provenientes do sensor de força) para que sejam obtidos os sinais necessários para os laços de realimentação de controle (Whitney, 1987).

No final dos anos 60 e 70, os primeiros controladores realimentados de força foram testados e várias estratégias de controle surgiram. Em 1966 Rothehild e Man lideraram o desenvolvimento de um cotovelo artificial para deficientes físicos, prevendo realimentação de força. E trabalhos com o controle contínuo de força, iniciaram em 1972, quando Groome desenvolveu um sistema de tele-operação prevendo a propagação do sinal de força

(Whitney, 1987).

Resumindo, o controle de força em robôs envolve a integração de tarefas-objetivos como modelagem do meio, posição, velocidade e realimentação de força e o ajuste dos torques aplicados nas juntas do robô. As realimentações dos vários sinais medidos como saídas do robô (posição, velocidade, força) e a escolha dos sinais de comando de entrada para o robô, resultam em diferentes estratégias para controle de força. Zeng and Hemani (1997) categorizaram estes métodos como:

- Algoritmos Convencionais (tradicionais) de controle de força;
- Estratégias de controle avançadas;

Podemos ainda acrescentar uma terceira linha de controladores de força, a que emprega ferramentas da área de inteligência computacional ou uma combinação destas com algoritmos convencionais e avançados (adaptativos) de controle de força/posição.

1.1 Controle Convencional de Força

Nesta classificação de controle de força de robôs estão incluídos algoritmos baseados na aplicação de relações entre posição e força aplicada ou entre velocidade e força aplicada, ou a aplicação de realimentações diretas de força ou combinações destas, como resumem Zeng and Hemani (1997):

- Métodos envolvendo a relação entre posição e força aplicada: a) controle de rigidez (*stiffness control*) somente com realimentação de posição (Sciavicco and Siciliano, 1996; Whitney, 1987; Spong and Vidyasagar, 1989); e b) controle de rigidez através da realimentação do erro de força (Sciavicco and Siciliano, 1996; Lewis et al., 1993; Whitney, 1987);
- Métodos aplicando a relação entre a velocidade e força aplicada: a) controle por impedância (Freund and Pesara, 1998; Sciavicco and Siciliano, 1996; Lewis et al., 1993; Spong and Vidyasagar, 1989; Whitney, 1987; Asada and Slotine, 1986); e b) Controle por Admitância (*Admittance Control* ou controle por acomodação);

- Métodos aplicando diretamente posição e força aplicada: a) controle híbrido de posição/força (Raibert and Craig, 1981; Sciavicco and Siciliano, 1996; Lewis et al., 1993; Spong and Vidyasagar, 1989); b) controle híbrido por impedância (Lewis et al., 1993); c) controle paralelo de força/posição (Chiaverini and Sciavicco, 1993)
- Métodos aplicando diretamente realimentação da força aplicada: controle explícito de força (Volpe and Khosla, 1993).

O Capítulo 2 descreve os controladores convencionais mais utilizados.

1.2 Controle Avançado de Força

Algoritmos avançados de controle de força são baseados em (Zeng and Hemani, 1997):

- Controle Adaptativo;
  Métodos de controle adaptativo incluem:
  - Controle de Rigidez (*Adaptative Compliant Control*);
  - Controle por Impedância (ou admitância) Adaptativa;
  - Controle Adaptativo de Posição/Força, e;
  - Controle Explícito de Força (Volpe and Khosla, 1993);

  Exemplos destes métodos podem ser encontrados em (Gasalino et al., 1987; Lewis et al., 1993; Villani et al., 1996).
- Controle Robusto;
  Métodos de controle robusto são:
  - Controle robusto de movimento por compliância (*robust compliant control*);
  - Controle Robusto por Impedância (ou admitância);
  - Controle Robusto de Posição/Força, e;
  - Controle Robusto Explícito de Força.

  Exemplos destes métodos podem ser encontrados em (Shin and Lee, 1997; Tang et al., 1997; Kwan, 1998; Lewis et al., 1993).

Parte destes métodos é descrita no Capítulo 2 do presente documento.

1.3 Controle Inteligente de Posição/Força

Do ponto de vista do controle dito "inteligente", muitos métodos para controle de robôs manipuladores podem ser utilizados, entre eles, pode-se destacar:

- Métodos baseado em Aprendizado Reforçado (*Reinforcement Learning*), especialmente indicado para os casos de operações repetitivas com o robô (Sutton and Barto, 1998);
- Estruturas empregando Redes Neurais Artificiais (RNAs) (Watanabe, 1996; Warwick, 1996; Morris and Khemaissia, 1996);
- Controladores usando Lógica *Fuzzy* (Kosko, 1992; Yager and Filev, 1994).

Em particular os algoritmos de controle baseados em redes neurais artificiais e lógica difusa (*fuzzy*) são avaliados como **abordagens de controle inteligente que normalmente não exigem nenhuma informação relativa à modelagem do robô ou do meio** – uma das vantagens advindas do uso destas técnicas (Watanabe, 1996).

Resta ainda mais uma classe de **controladores integrados** surgida em função do uso mesclado de técnicas convencionais ou avançadas com as técnicas inteligentes para controle de força/posição. Todas estas técnicas são exploradas no capítulo 3 do presente documento.

1.4 Objetivos e Contribuições deste trabalho

Este trabalho visa principalmente explorar a implementação prática de controladores integrados de posição e de força que mesclam o uso de uma rede neural trabalhando em paralelo com um controlador convencional nas malhas de controle de posição e de controle de força de um robô manipulador real do tipo Scara.

Inicia com o controle de posição integrado explorando o uso de duas redes neurais diferentes: a rede MLP (multicamadas de *perceptrons*) e a rede RBF (de função de base radial). É proposta uma maneira para incorporar redes RBF na estrutura de um neuro-controlador baseado em realimentação do erro (*feedback error learning control*), mesma estrutura usada com a rede MLP. Estas duas redes diferem na forma como mapeiam suas informações e na suas velocidades de treinamento. O modo como a rede RBF mapeia a informação de entrada se assemelha muito a um sistema *fuzzy*. As duas redes são treinadas em tempo real (*on-line*)

usando-se o mesmo algoritmo de aprendizado, o tradicional *back propagation* expandido com o termo *momentum*. A idéia é então identificar qual estrutura se mostra mais adequada para um caso prático de aplicação na área de controle de um sistema não-linear, de parâmetros incertos e variáveis no tempo. Estes controladores são comparados com outros convencionais como o PD e PID e um do tipo robusto simples baseado em modos deslizantes com o robô operando em diferentes regiões do seu espaço operacional e outro teste verificando como os controladores integrados reagem a uma perturbação (cinta elástica) presente no meio de uma trajetória retilínea. No caso do teste com a perturbação, o objetivo é avaliar-se a ocorrência de algum efeito residual de memória provocado por alguma das redes neurais utilizadas. São testadas redes MLP contendo 1 e 2 camadas ocultas na sua estrutura e redes RBF trabalhando com 5 à 9 funções gaussianas na sua camada intermediária. Os resultados obtidos são muito bons, principalmente no uso combinado de um controlador PID + uma rede RBF contendo 5 funções gaussianas por elemento de entrada de dados, apesar de ter sido percebido o acréscimo exigido na capacidade de processamento da CPU do robô. De modo geral, os controladores integrados de posição resolveram melhor o problema de regulação dada sua capacidade adaptativa em tempo-real. Em contrapartida, seu uso implica em maior poder de processamento disponível pela CPU do sistema onde este controlador visa ser aplicado.

Este trabalho também tenta propor a aplicação de diferentes índices de desempenho como forma de se buscar critérios mais objetivos para avaliação comparativa de controladores. Também foi testado na prática a aplicação de controladores de posição operando diretamente no espaço operacional: controlador por Jacobiana Inversa, controlador PD e PID operando diretamente sobre o espaço operacional do robô.

O trabalho finaliza com a proposta de diferentes estruturas de controladores integrados de força trabalhando dentro de um sistema de controle híbrido de posição/força usualmente adotado para robôs manipuladores. As diferentes estruturas vão evoluindo conforme diferentes resultados práticos vão sendo obtidos. Infelizmente não se logrou alcançar um bom controlador integrado de força. Diferentes combinações de entrada de dados foram avaliadas conduzindo a diferentes resultados. O uso da rede RBF foi impossibilitado dada sua acelerada capacidade adaptativa (o torque gerado pela rede rapidamente era levado à valores extremos) o que sugere o uso de períodos de amostragens inferiores a 2 milisegundos utili-

zados na prática. É sabido que controladores de posição/força exigem períodos de amostragem inferiores aos adotados para os controladores de posição em função das dinâmicas de elevada freqüência que surgem decorrentes da interação do robô manipulador em contato contra um meio. Infelizmente períodos de amostragem inferiores não puderam ser adotados na prática devido a limitações do poder de processamento da atual CPU do robô.

Duas novas estruturas de controladores integrados de força são propostas: uma baseada em modelo de referência (MRAC) e outra em que apenas uma rede neural trabalha tanto na malha para controle de força quanto para a malha de controle de posição.

1.5 Organização deste Documento

Este documento foi organizado em 7 capítulos acrescido de mais 3 apêndices, a saber:

Capítulo 1: o capítulo atual que introduz este trabalho.

Capítulo 2: Controle de Posição/Força de Robôs Manipuladores apresenta uma visão geral de controladores convencionais de força/posição além de uma rápida revisão de controladores de posição adotados neste trabalho. Introduz o conceito de Restrições Naturais e Artificiais além dos primeiros controladores desenvolvidos para esta área: controle de rigidez, controle por impedância, controle por admitância, controle híbrido de posição/força, controle híbrido por impedância, controle explícito de força e controle implícito de força. Continua apresentando métodos mais avançados para controle de força como os adaptativos, robustos e baseados em algoritmos de aprendizado. A principal contribuição deste capítulo é uma tabela em que métodos convencionais de controle de posição/força são comparados e outra tabela que compara alguns métodos avançados de controle de força.

Capítulo 3: Controle Inteligente: trata-se de uma revisão bibliográfica sobre controladores neurais na área de controle e mais especificamente quando aplicados no caso de robôs manipuladores. Inclui revisão de arquiteturas comumente usadas para controladores baseados em redes neurais. E termina com rápida revisão sobre controladores *fuzzy*.

Capítulo 4: Descrição do Robô e dos Controladores Propostos: introduz o robô manipulador usado na prática para implementar os trabalhos relacionados com esta tese e apre-

senta a metodologia adotada para a implementação dos controladores neurais integrados de força/posição. Descreve em detalhes algumas das implementações realizadas na prática. E termina introduzindo critérios de desempenho utilizados para inferir conclusões mais objetivas acerca dos resultados obtidos com os diferentes tipos de controladores testados.

Capítulo 5: Resultados de Controladores de Posição: compila uma série de resultados obtidos na prática com a implementação de controladores integrados de posição. Inclui testes realizados com 2 diferentes trajetórias além de um teste de rejeição a uma perturbação aplicada ao sistema de controle de posição. Termina com uma discussão final relacionando conclusões dos resultados obtidos com os controladores de posição implementados.

Capítulo 6: Resultados de Controladores de Força: agrupa os resultados obtidos com controladores convencionais e integrados de força. Inicia com testes realizados com controladores de posição operando diretamente sobre o espaço operacional, segue com teste de controladores Proporcionais e Proporcionais-Integrais de força (convencionais), finalizando esta seção com uma rápida discussão a respeito da importância da ação integral no controle de força e por fim apresenta os resultados obtidos pelas diferentes estruturas de controlares integrados de força propostos. Termina com uma discussão à respeito dos controlares integrados de força testados e com uma discussão mais geral à respeito de controle de força.

Capítulo 7: Conclusões e Perspectivas: resume os pontos abordados neste trabalho, ressalta as principais contribuições obtidas, resume as conclusões obtidas acerca dos controladores integrados de posição e dos controladores integrados de força, ressalta limitações e pontos não abordados por este trabalho e termina com sugestões para trabalhos futuros na área.

Apêndice A: Transformações de Coordenadas: contribui trazendo as deduções relativas às transformações de estados do robô (posição, velocidade e aceleração), do espaço de juntas para o espaço operacional (também dito cartesiano) e vice-versa. Completando trabalhos anteriores realizados com este robô.

Apêndice B: Controladores PID Digitais: revisa formas de implementação de controladores Proporcionais-Derivativos-Integrativos num sistema amostrado no tempo (controle digital). Inclui o método de ajuste adotado para os controladores convencionais PID usados neste trabalho.

Apêndice C: Algoritmo de aprendizado *backpropagation* expandido com termo *momentum* que revisa o algoritmo de aprendizado usado para as redes neurais MLP e RBF utilizadas neste trabalho.

Capítulo 2

Controle de Posição/Força de Robôs Manipuladores

2.1 Introdução

São apresentados alguns controladores desde os mais simples (clássicos), industrialmente adotados até algumas variações mais sofisticadas incluindo versões mais simplificadas de controle robusto. Note que industrialmente alguns robôs realizam tarefas em contato com o meio sem usar sensor de força ou levar em conta no algoritmo de controle alguma interação com o meio. Neste caso, está sendo realizado apenas controle de posição (no espaço de juntas ou no espaço operacional) e se o robô realmente entra em contato com o meio se faz necessário estabelecer a configuração que o robô deve alcançar dentro do meio, isto é, quanto está em contato com o meio.

Note que esta forma de definição é altamente imprecisa pois exige exatidão no estabelecimento dos pontos de contato do robô com a peça (tarefa que pode se tornar bastante complexa, quanto mais complexa for a superfície de contato de robô com o meio) e ainda, mais difícil de ser obtido na prática, o perfeito posicionamento entre o meio com o qual o robô entrará em contato e o robô, isto é, um perfeito ”casamento” de coordenadas entre o robô e o meio – o que na prática exige ”moldes” rígidos para posicionamento de peças dentro de uma linha de montagem operando com um robô sob estas condições. Obviamente, imprecisões de alinhamento da peça ou inexatidões presentes na peça e que não puderam ser modeladas podem incorrer em contados abruptos do robô com a peça (com forças de

contato superiores às que seriam desejadas), acarretando em casos mais extremos, danos à peça ou ao efetuador do robô.

Portanto, especialmente no caso de tarefas que exigem contato do robô com o meio (peça) é altamente desejável algoritmos de controle de levem em conta este contato, usando preferencialmente algum sensor de força e eventualmente sensores adicionais como câmeras de vídeo (ou mais simplesmente sensores CCD). Isto pode implicar no uso de 3 diferentes tipos de sensores: (1) sensores de posição nas juntas do robô, (2) sensor de força, normalmente montado entre a última junta do robô e seu efetuador e eventualmente: (3) sensor de visão. Obviamente que o último caso, trabalhando com sensor de visão exige algoritmos de controle mais complexos capazes de realizar integração sensorial entre os dados provenientes dos sensores do robô e o sensor de vídeo (Xiao et al., 2000), além de apenas a parte relacionada com o sensor de visão exigir complexas tarefas de reconhecimento de imagens.

Neste trabalho, não contamos com o terceiro sensor, mas dispomos do sensor de força, que por causa do seu custo elevado é descartado em aplicações que não exigem muita precisão de posicionamento de peças mesmo que o robô entre em contato com o meio.

Nos casos em que não está disponível um sensor de força, mas se exige que o robô execute em certos instantes de tempo, uma certa pressão contra um meio, industrialmente se costuma adotar controladores baseados em impedância. Um exemplo típico deste tipo de caso acontece na indústria montadora, em tarefas onde o robô realiza simples encaixes de peças baseados apenas em pressões. Neste último caso, não existe uma referência (perfil) de força especifica à ser realizado contra uma peça. Os valores absolutos que podem ser atingidos durante este tipo de atividade não são importantes, e uma variação entre certos limites que não impliquem em danos à peça ou ao efetuador do robô são perfeitamente aceitáveis. Para estes casos, não é necessário um **controle explícito de força**. Mas casos que exigem um controle mais preciso sobre a força que está sendo executada por sobre a peça exigem alguma forma de controle de força. Mesmo no caso de não estar disponível um controle de força, medidas indiretas da força realizada pelo robô em contato com o meio podem ser inferidas usando alguma técnica baseada na estimação da rigidez da peça em contato com o robô. Estas técnicas são conhecidas como **controladores de força implícitos** pois tentam levar em conta a força executada pelo robô em contato com um meio sem entretanto estar presente um sensor de força. Naturalmente, resultados melhores podem ser esperados

quando um sensor de força está presente e então não se trabalha sobre estimativas e sim sobre valores reais de forças advindas do contato entre o robô e o meio e assim algoritmos de controle explícitos podem ser empregados.

O problema básico envolvido com o controle de força de robôs é como determinar as forças de interação com o meio e realimentar eficientemente as informações disponibilizadas pelos sensores do robô, de forma a gerar os sinais de controle e obter as trajetórias de movimento e forças desejadas, respeitando as restrições do meio sem danificar o meio e nem submeter o robô manipulador a esforços (forças e momentos) que possam comprometer sua estrutura.

2.2 Controle de Posição

A grande maioria dos robôs industriais são utilizados apenas em tarefas que exigem no máximo controle de posição de seu efetuador. São operações nas quais, o manipulador não entra em contato com o meio e nos idos de 1999 correspondia a aproximadamente 90% dos casos de aplicações de robôs industriais (Vicent, 1999). Para controle de posição apenas, diversas soluções já foram desenvolvidas e técnicas de controle clássico resolvem perfeitamente bem diferentes casos à não ser nas situações onde ocorrem variações paramétricas no tempo (sistema variante no tempo): robô realizando tarefas de *pick-and-place*, manipulando diferentes peças (diferentes cargas, diferentes centros geométricos, diferentes momentos de inércia), e onde se exige alto desempenho do mesmo, entendido aqui como, desenvolvimento de altas velocidades (por exemplo, para transporte de pára-brisas dianteiros de automóveis, típica aplicação numa indústria montadora de carros).

O modelo dinâmico direto de um robô manipulador pode ser caracterizado pela equação (Sciavicco and Siciliano, 1996; Lewis et al., 1993; Spong and Vidyasagar, 1989):

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = \tau \tag{2.1}$$

onde: $B(q)$ se refere a matriz de inércia do robô manipulador (leva em consideração parâmetros físicos do robô: comprimento das juntas, distâncias até centros de massa de cada uma das juntas, massa de cada junta, massa dos motores, momentos de inércia em cada junta, etc); $C(q,\dot{q})$ são os termos de Corioli centrífugos do sistema (relaciona as interações

que ocorrem entre as diferentes juntas do robô notavelmente quanto maior são as velocidades desenvolvidas pelas juntas do mesmo); F está relacionado aos atritos nas juntas envolvidos neste sistema (varia com o tipo de acoplamento motor $\times$ elo em cada junta e o índice de redução envolvido entre os mesmos); $g(q)$ se refere ao vetor dos termos gravitacionais do sistema (nitidamente quando alguma junta sofre efeito gravitacional) e τ se refere ao vetor de torques aplicados em cada uma das juntas do robô para que este alcance as posições angulares desejadas para cada uma das juntas (vetor q), obedecendo uma certa velocidade (vetor $\dot{q}$) e aceleração angulares (vetor $\ddot{q}$).

2.2.1 Controle PD com Compensação de Gravidade

Este tipo de controlador leva em conta os efeitos gravitacionais que influenciam a posição do efetuador do robô. Sua lei de controle é dada por (Sciavicco and Siciliano, 1996):

$$u = g(q) + K_p\tilde{q} + K_d\dot{\tilde{q}} \tag{2.2}$$

que é aplicada sobre a equação do modelo dinâmico de um robô manipulador:

$$B(q)\,u + n(q,\dot{q}) = \tau \tag{2.3}$$

onde $n(\cdot)$ corresponde à compensação dinâmica, dado por:

$$n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F\dot{q} \tag{2.4}$$

onde:
- q corresponde à posição angular atual das juntas;
- q_d posição angular desejada para as juntas;
- $\tilde{q}$ corresponde ao erro de posicionamento: $\tilde{q} = q_d - q$;
- $\dot{q}$ corresponde às velocidades angulares instantâneas das juntas;
- $\dot{q_d}$ corresponde às velocidades angulares desejadas para as juntas;
- $\dot{\tilde{q}}$ corresponde ao erro de velocidade angular: $\dot{\tilde{q}} = \dot{q}_d - \dot{q}$;
- B corresponde a matriz de inércia estimada para o sistema;
- $n(\cdot)$ corresponde a outros termos de compensação dinâmica do sistema;
- $C(\cdot)$ corresponde aos termos de Coriolis;
- $F(\cdot)$ corresponde aos termos relativos ao atrito;
- u corresponde à lei de controle (ação determinada pelo PD);
- τ corresponde ao torque que é enviado às juntas;

Note que um Controlador Proporcional no espaço de juntas é obtido apenas zerando-se o ganho derivativo ($K_d = 0$) e então temos:

$$u = g(q) + K_p\tilde{q}$$

Um controlador PD é globalmente assintoticamente estável enquanto um controlador Proporcional simples é apenas globalmente estável (Sciavicco and Siciliano, 1996).

2.2.2 PD no Espaço Operacional

A lei de controle para um PD no espaço operacional pode ser obtida de forma quase intuitiva, uma vez que basta traduzir estados do robô do espaço de juntas para o espaço operacional ($x = K(q)$ e $\dot{x} = J_a(q) \cdot \dot{q}$), aplicar os devidos ganhos e novamente voltar para o espaço de juntas (o torque deve ser gerado no espaço de juntas – $\Delta q = J_A^T(q) \cdot \Delta x$). Sua lei de controle pode ser dada por (Sciavicco and Siciliano, 1996):

$$u = g(q) + J_A^T(q) \cdot K_P \cdot \tilde{x} - J_A^T \cdot K_D \cdot J_a(q) \cdot \dot{q} \tag{2.5}$$

A figura 2.1 ilustra na forma de um diagrama em blocos o controlador PD no espaço operacional.

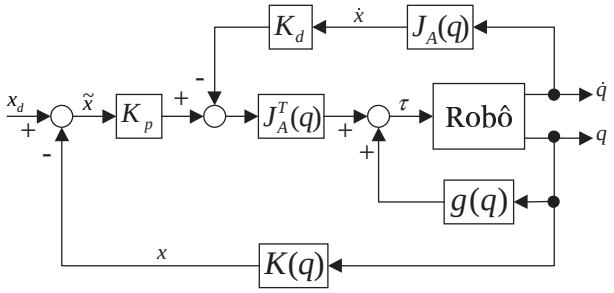


Figura 2.1: Controlador PD no espaço operacional

Bier (2000) provou que o PD com compensação de gravidade é globalmente assintoticamente estável enquanto a matriz Jacobiana Analítica for de posto completo (isto é, o robô não esteja próximo de uma região de singularidade).

Outra variação simples de controlador no espaço operacional é o controlador baseado na inversão da matriz Jacobiana.

2.2.3 Controlador por Jacobina Inversa, J_A^{-1}

A lei de controle para este controlador é dada por Sciavicco and Siciliano (1996):

$$u = K_p \, J_A^{-1}(q) \left(x_d - x \right) \tag{2.6}$$

2.2.4 Controladores de Posição no Espaço de Juntas $\times$ Espaço Operacional

Da mesma forma que normalmente se realiza controle de um robô no espaço de juntas, pode ser feito o controle do robô diretamente a partir do espaço operacional. Entretanto esta técnica deve ser aplicada com mais critério uma vez que exige que a ação de controle calculada à partir do espaço operacional deve ser "traduzida" para o espaço de juntas através da matriz Jacobiana. Ocorre que quando o robô se encontra em configurações singulares (fora ou no limite do seu espaço de trabalho), esta matriz reduz seu posto e pode fica impossível de se determinar o torque para a junta afetada ocasionando paralisia do robô manipulador. Uma condição simples de singularidade que já afeta esta matriz, para o caso de uma robô planar XY (caso das duas primeiras juntas de um robô Scara adotado na parte prática deste trabalho), ocorre quando estendemos as 2 primeiras juntas deste robô, formando um ângulo nulo entre as suas primeiras juntas. Este ângulo nulo implica em posições zeradas na matriz Jacobiana e portanto pode resultar na indeterminação de torques necessários para

o comando de certa junta (no caso da segunda junta).

2.3 Controle de Força

O modelo dinâmico referente a um robô manipulador em contato com o meio pode ser expresso pela equação (Sciavicco and Siciliano, 1996; Lewis et al., 1993; Asada and Slotine, 1986):

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = \tau - J^T(q)h \tag{2.7}$$

que adicionalmente à equação 2.1 (do robô enquanto não está em contato com o meio) possui os termos: $J^T(q)h$ que se referem à: J^T = matriz jacobiana geométrica transposta do robô, que realiza um mapeamento entre a posição e velocidade angular desenvolvida por cada uma das juntas do robô (no espaço de juntas) versus posição (p) e orientação (o) do efetuador (fornecidos no espaço operacional[1] e não mais no espaço de juntas): $v = [\dot{p} \quad (do/dt)]^T = J(q)\dot{q}$; e vetor h que se refere às forças de contato desenvolvidas pelo efetuador do robô contra a superfície em contato. Note que agora os torques a serem desenvolvidos por cada uma das juntas do robô devem levar em conta as forças de contato desenvolvidas pela interação o meio. Para maiores detalhes sobre modelagem de um robô rígido em contato com ambientes passivos sugere-se: Amaral (2000).

O controle de força pode ser realizado de 2 formas: 1) com um sensor de força presente (**controle explícito**) ou 2) sem sensor de força presente (**controle implícito**). No segundo caso, provavelmente por questões de custo ou simplicidade da tarefa, não existe um sensor de força para possibilitar uma realimentação direta deste sinal para o cálculo da ação de controle. Neste caso, a força de contato desenvolvida contra meio pelo robô é estimada com base na constante de rigidez estática do meio: $\hat{f} = k_s\,(x - x_e)$, onde k_s se refere à constante de rigidez estática do meio; x é a posição atual sendo capturada pelos *encoders* de posição do robô; e x_e é o ponto onde inicia o contato – ver figura 2.2. O problema é que muitas vezes é difícil se conhecer com precisão o valor de x_e ou variam os valores de k_s e portanto, a força de contato desenvolvida com o meio se torna mal estimada. Entretanto não deixa de

[1]Espaço Operacional: se refere as coordenadas do efetuador do robô informadas no espaço cartesiano cuja origem coincide com a base do robô. Alguns autores também se referem a este espaço como de coordenadas cartesianas. Notar que pode existir ainda o "espaço tarefa" usado quando uma tarefa é definida no espaço cartesiano usando coordenadas cuja origem não coincide com a base do robô; neste caso são necessárias transformações geométricas entre coordenadas do espaço operacional e do espaço tarefa.

ser uma solução econômicamente atraente para os casos em que não é necessário exatidão nos valores das forças de contato realizadas contra o meio; caso da indústria de auto-peças em tarefas que exigem encaixe de peças sob pressão (não é necessário precisão de força mas força o suficiente para encaixar painéis por exemplo).

Além disto o controlador de força pode estabelecer um laço de realimentação baseado apenas na informação captada pelo sensor de força (controle explícito puro de força) ou estabelecer uma malha de realimentação que necessita de informação da posição atual do robô. Neste último caso, este tipo de controle é dito ”por impedância”porque necessita da medição de posição do robô (Volpe, 1990).

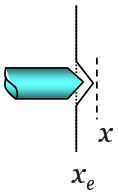


Figura 2.2: Detalhe do robô em contato com o meio, estimando força de contato.

2.3.1 Restrições Naturais e Artificiais

Geralmente uma tarefa que envolve contato com o meio pode ser descrita em função da geometria deste meio, que pode ser uma peça ou superfície. Assim, referências de força e posição são especificadas em função da geometria da tarefa e diferentes esquemas de interações de controle (de posição e/ou força) podem ser empregadas para a execução desta tarefa que possui movimentos restringidos em certas direções.

Podemos definir o ”espaço da tarefa” na qual a especificação da tarefa a ser realizada poder ser especificada de forma mais simples, em função da geometria do meio. Trata-se de um sistema de coordenadas ortogonais entre si e variante no tempo, de tal forma que a cada instante de tempo e ao longo dos eixos deste sistema de coordenadas, a tarefa pode ser expressada como um problema de controle ”puro” da trajetória de posição ou como um problema de controle de força. O ”ou” aqui é exclusivo: não se pode controlar num simples grau de liberdade tanto a posição quanto a força aplicada no efetuador. A título de analogia, não se pode especificar tanto a tensão como a corrente que se deseja passar por um resistor (Asada and Slotine, 1986). A dualidade existente entre controle de posição e de força pode ser expressada em termos de restrições naturais e restrições artificiais.

Uma tarefa real de manipulação é caracterizada por situações complexas, onde em algumas direções existem restrições para o posicionamento do efetuador e em outras se está sujeito a interações com forças limitadas. Durante a execução de uma tarefa, a natureza da restrição ainda pode mudar substancialmente.

A necessidade de lidar com situações de contato requer a capacidade (possibilidade) de especificar e desempenhar um controle tanto de posição do efetuador quanto de controle da força de contato. Um exemplo é o caso de uma tarefa de acabamento (desbaste) de uma peça onde o movimento da ferramenta (efetuador) é especificado como sendo tangente à peça, enquanto é desejado a aplicação de um força de contato obedecendo um certo valor ao longo da direção normal de contato com esta peça.

Um aspecto fundamental a ser considerado é que não é possível se impor simultaneamente valores arbitrários de posição e força ao longo de cada direção. Como conseqüência, se deve garantir que as referências de trajetórias para o sistema de controle deverão ser compatíveis com as restrições impostas pelo meio durante a execução de tarefas, de forma a se obter uma especificação correta para o problema. Isto é, não danificar nem a peça em contato com o efetuador e nem forçar a estrutura do robô manipulador.

Faz-se necessário uma análise estático-cinemática da situação de interação entre o manipulador e o meio, que leva em conta as seguintes considerações:

1. Ao longo de cada grau de liberdade no espaço de tarefas, o meio impõe tanto uma restrição de posição ou de força ao efetuador do manipulador. Estas restrições denominam-se restrições naturais uma vez que elas são determinadas pela própria geometria da tarefa (meio, peça);

2. Ao longo de cada grau de liberdade no espaço de tarefas, o manipulador pode controlador somente as variáveis que não estão sujeitas às restrições naturais. Os valores das referências para estas variáveis são denominadas de restrições artificiais, uma vez que elas são impostas com respeito à estratégia adotada para executar uma dada tarefa.

Estes dois conjuntos de restrições são complementares, uma vez que são especificadas diferentes variáveis para cada grau de liberdade. Também estes conjuntos devem permitir a completa especificação da tarefa. As variáveis estato-cinéticas estão associadas com restrições, isto é, velocidades, forças e momentos generalizados.

Num caso geral, vale a pena introduzir uma sistema de coordenadas Restritivo, também conhecido como *constraint frame* ou como doravante passaremos a adotar, coordenadas do **espaço tarefa**, O_{task}–$X_{task}Y_{task}Z_{task}$, não necessariamente alinhado com o sistema de coordenadas de base do robô, também conhecido como *base frame* ou coordenadas do **espaço operacional**, definição que de agora em diante passaremos a utilizar, de forma a facilitar a descrição da tarefa e permitir a especificação das restrições naturais e artificiais. A introdução do sistema de coordenadas do espaço tarefa simplifica notavelmente o planejamento da tarefa, mas a estratégia de controle terá que naturalmente levar em conta a matriz de transformação de coordenadas envolvida no processo e eventualmente sua variação no tempo durante a execução da tarefa. As transformações de coordenadas são necessárias para transformar quantidades expressadas no espaço operacional nos correspondentes valores no espaço da tarefa (quando se está lendo as informações vindas dos sensores do robô) e vice-versa (quando se está repassando comandos de torque para as juntas do robô).

Do ponto de vista geométrico de uma tarefa simples, os graus de liberdade são descritos com referência a um sistema de coordenadas de base adotado para expressar quantidades no espaço operacional.

Deve ser ressaltado que as estratégias de controle de posicionamento e força em tarefas do robô em contato com um meio estão implicitamente levando em conta a presença de restrições naturais e artificiais. Assim, diferentes esquemas de interações de controle podem se empregados para execução de movimentos restringidos estabelecendo-se apropriadamente as referências de força e posição em função da geometria do meio.

Exemplo de deslocamento sobre uma superfície

A tarefa consiste em deslizar o efetuador do robô sobre uma superfície aparentemente plana mas com irregulares conforme mostra a figura 2.3.

A geometria da tarefa sugere que o sistema de coordenadas da tarefa (o restringido) coincida com sistema de coordenadas de base do robô manipulador (ou espaço operacional como é referenciado no restante do trabalho). Percebe-se que o movimento não é restringido na direção X mas este movimento é controlado (controle de posição/velocidade), pois o robô deve assumir uma velocidade limitada: $\dot{x} \leq \dot{x}_d$. Mas o movimento é restringido na direção Z e neste caso é realizado controle de força: a força desenvolvida pelo robô em contato com

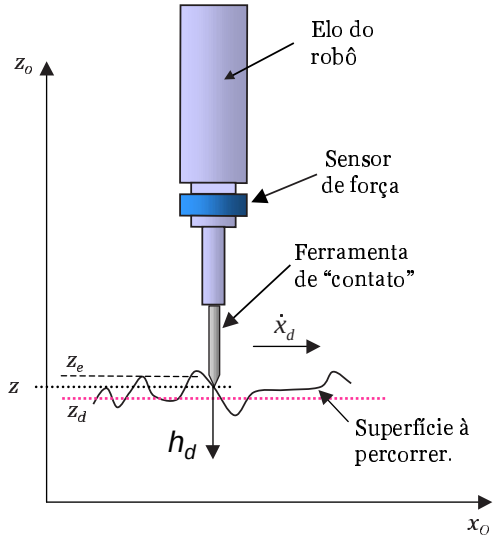


Figura 2.3: Exemplo de movimento restringido

o meio deve ser: $h \leq h_d$.

2.3.2 Controle de Força por Rigidez

O controle de rigidez (*Stiffness control* ou *Compliance Control*[2]) pode ser ativo ou passivo. No **controle passivo**, o efetuador do braço de um robô é equipado com um dispositivo mecânico passivo composto por molas, que amortecem e direcionam o contato do efetuador com o meio. Funciona em muitas tarefas específicas como por exemplo, inserção de pinos, trabalho com rebites (*pegs*) de certo comprimento e orientação predefinida em relação ao braço do robô (Zeng and Hemani, 1997; Asada and Slotine, 1986). **Vantagem**: dispositivos mecânicos passivos são capazes de respostas rápidas e são uma solução relativamente barata (Asada and Slotine, 1986). **Desvantagem**: sua aplicação está necessariamente restrita a tarefas muito específicas como por exemplo na manipulação de pinos de certo comprimento e com certa orientação predefinida em relação ao efetuador (Asada and Slotine, 1986).

Em contraste, o controle ativo de rigidez, pode ser comparado a uma mola programável (Zeng and Hemani, 1997; Sciavicco and Siciliano, 1996; Spong and Vidyasagar, 1989; Whitney, 1987) uma vez que através da realimentação de força, a rigidez (*stiffness*) do sistema em malha fechada é alterado. A Fig 2.4 mostra o princípio básico de um controle ativo por rigidez.

[2]Rigidez é definido como: $= K_p$ e Compliância como: $= K_p^{-1}$

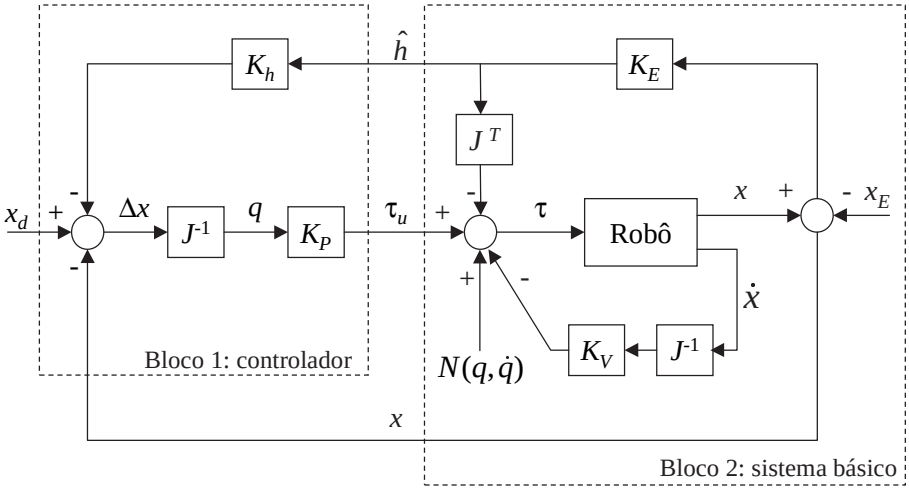


Figura 2.4: Controle ativo de rigidez, primeira versão (controle explícito).

onde:

J	é a matriz Jacobiana do Robô;
x_d	corresponde ao vetor das posições desejadas;
x e $\dot{x}$	são respectivamente os vetores de posição e velocidade no espaço cartesiano.
q	é o vetor das posições angulares das juntas;
$\dot{q}$	é o vetor das velocidades angulares das juntas;
τ_u	é o vetor de torques da ação de controle para cada junta, associado com o controle por rigidez;
$N(q,\dot{q})$	é um vetor não linear de compensação dinâmica (*feedforward*) relacionado com os termos: matriz de inércia, forças centrífugas e de Coriolis e de gravidade;
τ	é o vetor final de torques à serem enviados às juntas;
x_E	é o vetor de posição do meio com o qual está sendo realizado o contato (no espaço cartesiano);
K_E	é a matriz de rigidez dos sensores e do ambiente; e,
$\hat{h}$	é o resultante vetor de força de contato (estimado) no espaço de trabalho.
K_P e K_V	são os ganhos do controlador, normalmente matrizes diagonais;
K_h	é a matriz de compliância para os comandos de mudança de posição.

Note que é considerado um robô não redundante no espaço 3-D, assim, a dimensão dos

vetores são de 6×1, e das matrizes de 6×6 para um robô com 6 graus de liberdade (ou 6 D.O.F.).

Para melhor entendimento, a Fig. 2.4 é dividida em duas partes: a do controlador – bloco 1 e a do sistema básico no bloco 2. O sistema básico inclui o robô e o meio, realimentação de velocidade ($\dot{x}$) e uma compensação não linear para o sistema dinâmico do robô (matriz $N(q, \dot{q})$). O laço de controle por rigidez compreende uma realimentação do tipo proporcional para a força e posição (K_p). Os torques nas juntas, τ_P, são definidos em função da força de contato que se deseja manter contra o meio:

$$\tau_u = K_P \, q \tag{2.8}$$

A unidade de K_P na equação 2.8 é força/deslocamento, que define a rigidez. Assim, o controle ativo de rigidez permite que o usuário especifique arbitrariamente a rigidez mecânica do manipulador selecionando diferentes valores para a matriz K_h. Se Δx não é convertido para deslocamentos angulares nas juntas através do inverso do Jacobiano (J^{-1}), então o esquema de controle se modifica para o que é mostrado na figura 2.5. O bloco 2 da figura 2.5 é o mesmo bloco 2 da figura 2.4. A malha de controle (bloco 3) calcula o torque nas juntas como sendo:

$$\tau_P = J^T \, K_x \, \Delta x \tag{2.9}$$

onde: K_x representa a matriz de rigidez.

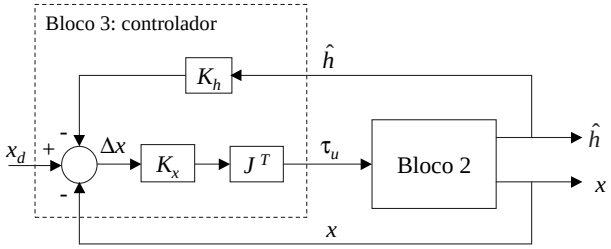


Figura 2.5: Controle ativo de rigidez, segunda versão (controle implícito).

Uma vez que $\Delta x = J \, q$ então $\tau_u = J^T \, K_x \, q$, o que significa que:

$$K_P = J^T \, K_x \, J \tag{2.10}$$

Vantagens: não exige um sensor de força (caro) e trata-se de uma estrutura de controle de

força simplificada, baseada na força de reação ao contato desenvolvida pelo meio (o meio é modelado simplesmente como sendo um mola com certa constante de rigidez): $\hat{h}_E = K_E(x - x_E)$.

Desvantagens: Notar que este é um método indireto de controle de força pois não é usado um sensor de força, isto é, através do erro de posição, $\tilde{x}$, é gerado uma força equivalente de contato, F_E que é posteriormente traduzida para uma referência de posição (via matriz K_h) que é comparada com a posição desejada, x_d e com a posição lida dos *encoders* de posição do robô, x. Esta abordagem exige *a priori* que se conheça a constante de rigidez do meio, K_E, que pode ainda mudar conforme a direção do movimento e composição do meio. E como realiza uma compensação dinâmica não linear do sistema, $N(q, \dot{q})$, fica sujeito às incertezas presentes na modelagem do sistema. Ainda, pode ser difícil sintonizar a matriz K_h responsável por gerar uma referência de erro de posição conforme a força de reação desenvolvida no contato. Note ainda que esta estrutura de controle não prevê variação de parâmetros no sistema robô-meio, o que quer dizer que não possui nenhuma característica adaptativa.

2.3.3 Controle de Força por Impedância

A idéia básica por detrás do controle por impedância, não é especificar as posições e força desejadas, mas ao invés disto, se especificar uma relação dinâmica entre força e posição, definindo-se uma rigidez mecânica ou "impedância" (Zeng and Hemani, 1997; Sciavicco and Siciliano, 1996; Lewis et al., 1993; Spong and Vidyasagar, 1989; Whitney, 1987; Asada and Slotine, 1986). O sistema de controle do manipulador deve ser projetado não apenas para seguir um segmento de trajetória, mas também para dosar a impedância mecânica do manipulador (Zeng and Hemani, 1997).

A relação entre a velocidade $\dot{x}$ e a força aplicada, h, é referenciada como a impedância mecânica, Z_m, sendo representado por:

$$\frac{H(s)}{\dot{X}(s)} = Z_m(s) \tag{2.11}$$

Em termos da posição $X(s)$ pode ser escrito (no domínio da freqüência):

$$\frac{H(s)}{X(s)} = s\, Z_m(s) \tag{2.12}$$

Num caso linear, a impedância desejada poderia ser especificada como:

$$s\,Z_m(s) = Ms^2 + Ds + K \tag{2.13}$$

As matrizes constantes, M, D e K representam respectivamente valores desejados para inércia, amortecimento e rigidez do contato. É tarefa do algoritmo de controle, garantir uma resposta do sistema como o especificado pela equação 2.12.

A figura 2.6 mostra uma estrutura típica de uma malha de controle por impedância, que determina o valor apropriado para $Z_m(s)$.

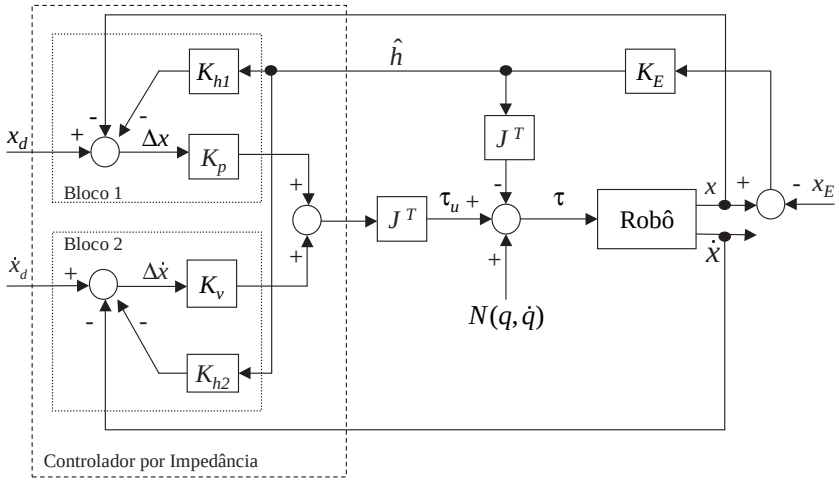


Figura 2.6: Controle típico por impedância.

A figura 2.6 é similar à figura 2.4 com a adição de uma malha de realimentação para a incorporação da velocidade e do efeito da força de contato sobre a velocidade. Desta forma, o controle por impedância tranforma-se num controlador Proporcional-Derivativo. Note que a força de contato é mais uma vez estimada fazendo-se: $\hat{h} = K_h(x - x_E)$ (o meio é modelado como uma mola simples) de maneira similar ao que foi realizado no controle por rigidez. A força de contato estimada é transformada numa referência extra de velocidade por meio da matriz K_{h2}. O comando de torque nas juntas, é então definido como:

$$\tau_u = J^T(q)\,(K_p\,\Delta x + K_v\,\Delta \dot{x}) \tag{2.14}$$

Na figura 2.6, a malha de controle do bloco 2 têm o efeito introduzir um fator de amortecimento no manipulador quando ele está em contato com o meio.

O controle por impedância é normalmente empregado quando um robô necessita se adaptar às características de amortecimento do meio (Zeng and Hemani, 1997). O controle por impedância pode ser realizado de outra forma ainda, como mostra a figura 2.7, no qual, x_F representa a equivalente realimentação de força com trajetória, x_I é a transformação da trajetória desejada definida como a solução para a equação diferencial:

$$M\,\ddot{x}_I + D\,\dot{x}_I + K\,x_I = -\hat{h} + M\,\ddot{x}_d + D\,\dot{x}_d + K\,x_d \tag{2.15}$$

onde $x_I(0) = x_d(0)$, $\dot{x}_I(0) = \dot{x}_d(0)$. M, D e K são as mesmas matrizes já definidas anteriormente. A equação (2.15) pode ser obtida por inspeção à partir da figura 2.7. Isto implica na mesma definição de impedância que as equações (2.12) e (2.13). Pode ser percebido que no projeto do controle por impedância, x_I é função tanto da entrada x_d quanto da força de contato estimada, $\hat{h}$. A impedância desejada para o manipulador é obtida uma vez que o subsistema controlado por posição no bloco 5 assegura que a posição x do efetuador segue a trajetória x_I, definida na equação (2.15).

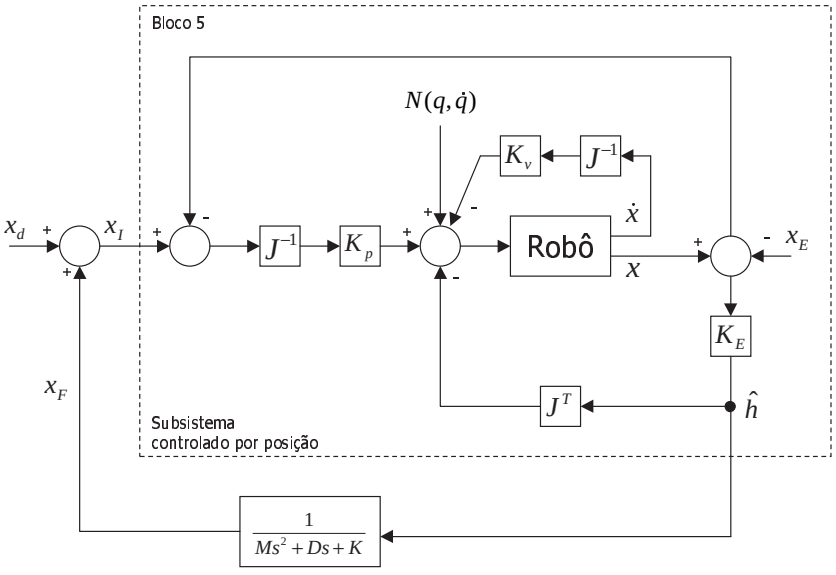


Figura 2.7: Controle por impedância baseado em posição.

Vantagens: é um tipo de controlador econômico (continua não exigindo sensor de força) e simples de ser implementado e ainda robusto em relação a incerteza de parâmetros (excetuando-se os efeitos gravitacionais) (Asada and Slotine, 1986). É uma abordagem que tenta monitorar a relação dinâmica existente entre a força e a posição, ao invés de controlar

diretamente a posição ou a força (Sciavicco and Siciliano, 1996; Asada and Slotine, 1986).
Desvantagens: apresenta problema de baixo desempenho porque funciona bem quando o movimento é limitado à velocidades muito baixas (Asada and Slotine, 1986). Além disto apresenta problemas quando o manipulador está próximo de configurações de singularidade – este problema surge pela presença da matriz Jacobiana, $J(q)$ que aparece na lei de controle dada pela expressão (2.14) (Asada and Slotine, 1986).

2.3.4 Controle Híbrido de Posição/Força

No controle híbrido de posição/força, o controle da posição e controle da força podem ser considerados separadamente. A eficiência desta abordagem híbrida de controle foi primeiro estudada por Raibert and Craig (1981).

Este controle consiste em reescrever a dinâmica do manipulador diretamente em termos de coordenadas do efetuador (ou coordenadas do espaço operacional) ao invés de deslocamentos nas juntas ($q, \dot{q}, \ddot{q}$), e então se controlar a posição, força ou impedância mecânica ao longo de cada um dos eixos do espaço da tarefa (Asada and Slotine, 1986).

O controle híbrido de posição/força combina a informação de torque e força com dados de posição, baseado no conceito de (Mason, 1981), que define dois espaços de trabalho ortogonais e complementares entre si, um para o deslocamento e outro para força.

A figura 2.8 representa um robô manipulador, seu meio e o elemento de compensação da gravidade. Doravante esta figura será utilizada para representar o robô, seu meio e compensação gravitacional. A figura 2.9 mostra o esquema de controle híbrido de posição/força.

Na figura 2.9, $S = diag(s_j)$ onde $j = 1, \ldots, n$ é chamada de matriz de seleção de compliância, onde n corresponde ao número de graus de liberdade do robô. A matriz S determina os subespaços nos quais são controlados a força e posição. s_j é ajustado com 1 ou 0. Quando $s_j = 0$, o j-ésimo grau de liberdade (*DOF*) deve ser controlado por força, senão, deve ser controlado por posição.

O comando de torque é dado por:

$$\tau = \tau_p + \tau_f \tag{2.16}$$

onde τ_p e τ_f são os comandos de torque atuando respectivamente nos subespaços de posição

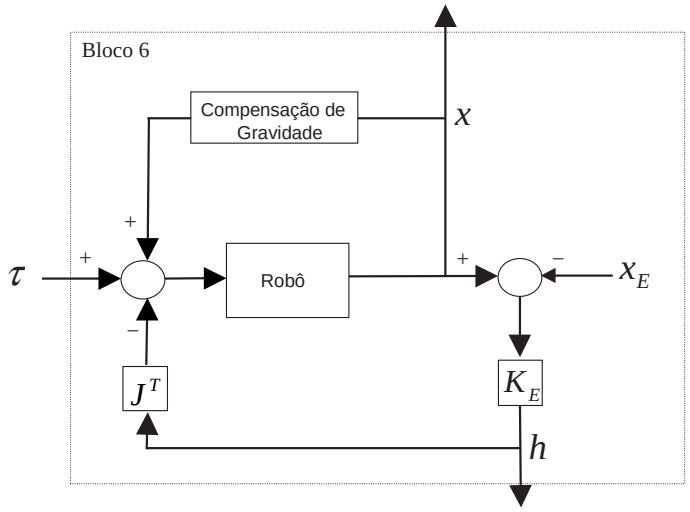
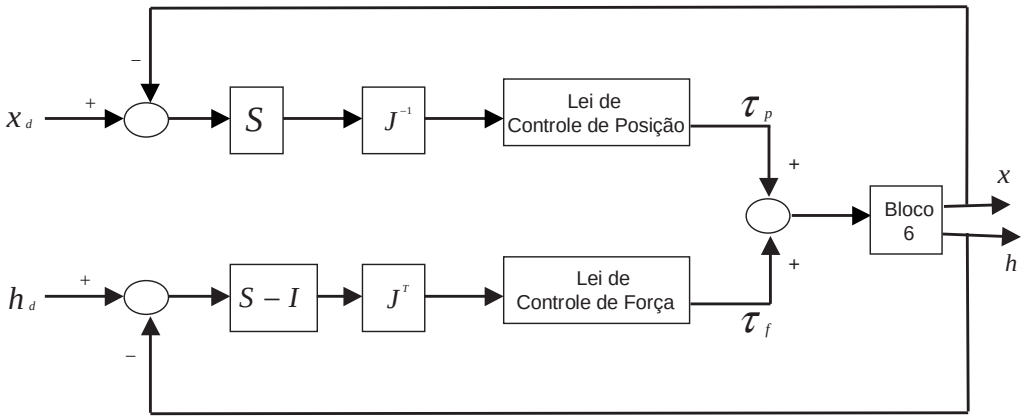


Figura 2.8: Representação de um robô manipulador, contato com o meio e compensação gravitacional.



Obs: O bloco "6" é o que aparece detalhado na figura 2.8.

Figura 2.9: Estrutura básica do controle híbrido de posição/força.

e força. Desta maneira, o controle de posição é desacoplado do controle de força. As leis de controle para cada uma dos subespaços podem ser determinados de forma independente, assim, as diferentes exigências de controle para os seguimentos desejados para posição e força são cumpridas simultaneamente. Normalmente, a lei de controle para posição da figura 2.9 consiste de uma ação PD e a lei de controle de força é do tipo de ação PI. É assim porque é mais desejado uma resposta mais rápida para o controlador de posição e para o controlador de força, um erro menor é preferível (Zeng and Hemani, 1997).

Vantagens: simplicidade da idéia, intuitiva: separar o controle de posição e força dentre os graus de liberdade para movimentos não restringidos e movimentos restringidos relativos ao espaço especificado para a tarefa.

Desvantagens: apresenta problemas de robustez relacionado à compensação dos termos gravitacionais e ainda que a dinâmica da interação com o meio está significativamente mais relacionada com o espaço da tarefa que com o espaço das juntas (Asada and Slotine, 1986). Este último problema torna-se mais evidenciado em baixas velocidades de movimentação quando a dinâmica do robô é dominada pelos termos gravitacionais e de atrito nas juntas (Asada and Slotine, 1986). Mais genericamente o controle de força/posição está mais relacionado com os fortes efeitos de adesão (*stiction*), flexibilidade das juntas e elos e folga nas transmissões (*backslach*) (Asada and Slotine, 1986).

2.3.5 Controle Híbrido por Impedância

O controle híbrido por impedância (Anderson and Spong, 1988), que combinaram numa única estratégia, o controle por impedância com o controle híbrido de posição/força. Isto permitiu ao projetista obter maior flexibilidade na escolha da impedância desejada para o sistema robótico sendo controlado. Uma distinção de impedâncias nos subespaços controlados por força e controlado por posição pôde então ser obtido. Assim, além de se manter os requisitos de velocidade (ou posição), a trajetória desejada para força pôde ser seguida. A figura 2.10 mostra o diagrama de blocos da estrutura de controle híbrida por impedância.

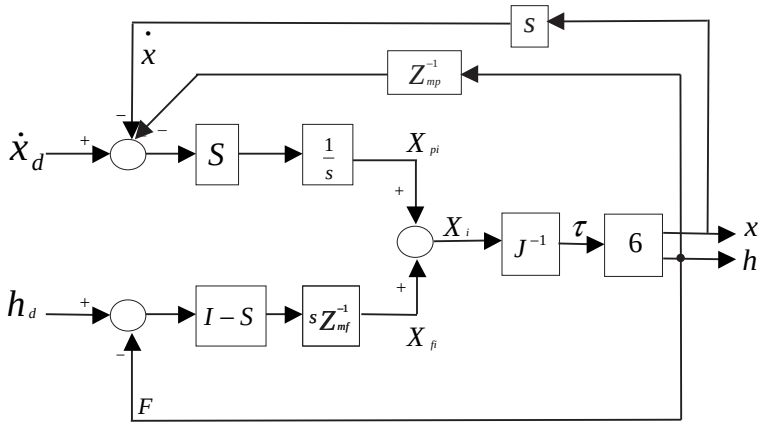


Figura 2.10: Controle Híbrido por impedância.

Na figura 2.10, Z_{mp} e Z_{mf} são os termos desejados de impedância selecionados pelo usuário. Se eles forem selecionados como matrizes diagonais, seus elementos representam a impedância para cada grau de liberdade do robô. S é a matriz compliância de seleção, a mesma que a do controle híbrido de posição/força. A trajetória desejada e modificada, X_i é

dada por:

$$X_i = X_{pi} + X_{fi} \tag{2.17}$$

onde X_{pi} e X_{fi} são as trajetórias transformadas para Posição e força expressadas nos subespaços de posição e força, respectivamente.

Estabilidade:. A estabilidade do controlador híbrido de posição/força têm sido discutidas em (An and Hollerbach, 1989; Dégoulange and Dauchez, 1994; Ficher and Mujtaba, 1992; Murphy and Wen, 1991; Yabuta et al., 1988). Uma **arquitetura generalizada** para o controle híbrido de posição/força foi indicada por (Seraji, 1994). A **estabilidade** do controlador híbrido de posição/força usando o **método direto de Lyapunnov**, e as **condições fundamentais de estabilidade** para o algoritmo híbrido também foram mostradas por (Yabuta, 1992).

Vantagens: tentar controlar as interações dinâmicas entre o manipulador e o meio diretamente no espaço tarefa (em oposição ao controle por Impedância) (Asada and Slotine, 1986). De modo similar ao controle por impedância, é simples e robusto em relação à incertezas paramétricas (Asada and Slotine, 1986).

Desvantagens: primeiramente, de modo similar ao caso do controle por impedância, apresenta baixo desempenho (restrito à baixas velocidades de movimentação) (Asada and Slotine, 1986). E adicionalmente este esquema de controle padece de problemas relacionados com a configurações do robô próximas à pontos de singularidade, como pode ser esperado à partir da presença da matriz Jacobiana, $J(q)$ (Asada and Slotine, 1986).

2.3.6 Comparação dos métodos convencionais para controle de força

Um resumo comparativo segue na tabela 2.1 que mostra as semelhanças e diferenças nos vários métodos de controle de força.

2.4 Controladores "Avançados"

A utilização do robô em tarefas complexas levam a tratar de vários problemas desafiantes, como incertezas paramétricas, meios desestruturados e perturbações externas. As dificuldades encontradas como a presença de dinâmicas não modeladas, ruído em sensores e perturbações externas estimula as necessidades para mais pesquisas em algoritmos

Classificação do Algoritmo		Espaço de Trabalho	Variáveis Medidas	Variáveis Modificadas	Baseado em
Controle Ativo de Rigidez	Versão 1.	Espaço de de Juntas	Posição Força	Deslocamento das Juntas Força de Contato	Matriz de Rigidez das Juntas
	Versão 2.	Espaço de Tarefa[a]		Erro de Posição Força de Contato	Matriz de Rigidez
Controle por Impedância	Básico	Espaço de Tarefa	Posição, Velocidade, Força	Erro de Posição e de Velocidade, Força de Contato	Impedância
	Baseado em Posição			Transformação da Trajetória Desejada e Força de Contato	
Controle por Admitância			Força	Erro de Força	Admitância
Controle Híbrido	Híbrido Posição/Força	$\mathbb{P}$[b]	Posição	Erro de Posição	Posição
		$\mathbb{F}$[c]	Força	Erro de Força	Força
	Híbrido por Impedância	$\mathbb{P}$	Força	Erro de Velocidade	Z_{mp}[d]
		$\mathbb{F}$		Erro de Força	Z_{mf}[e]
Controle Explícito de Força	PI, PD, PID, etc	Espaço de Tarefa	Força	Erro de Força	Força Desejada F_D
Controle de Força Explícito		Espaço de Tarefa	Posição	Erro de Posição	Rigidez Prédefinida

[a] Espaço de Tarefas = $\mathbb{P} \oplus \mathbb{F}$
[b] $\mathbb{P}$ é o subespaço de Posição.
[c] $\mathbb{F}$ é o subespaço de Força.
[d] Z_{mp} é a impedância expressa no subespaço de Posição.
[e] Z_{mf} é a impedância expressa no subespaço de Força.

Tabela 2.1: Comparativo de vários métodos de controle de força convencionais.

de controle de força avançados de forma a ser melhorar o desempenho do controlador. Um método avançado deve estar apto a propiciar um seguimento de força preciso ou a realização perfeita de uma tarefa, mesmo na presença de parâmetros desconhecidos e incertezas com relação ao robô e ao meio. Baseado nos métodos de controle tradicionais, existem muitas técnicas avançadas de controle de força, as quais são classificadas em:

- Controle Adaptativo (Zeng and Hemani, 1997);
- Controle Robusto (Zeng and Hemani, 1997); e,
- Controle utilizando técnicas da área de Inteligência Computacional (Watanabe, 1996):
  - Controle por Aprendizado (Zeng and Hemani, 1997; Song and Chu, 1998);
  - Controle utilizando Redes Neurais Artificiais;
    (Battistela, 1999; Katič and Vukobratovič, 1996; Er and Liew, 1997; Ge et al., 1997; Watanabe, 1996; Kim and Lewis, 1999; Lewis et al., 1995; Vukobratović and Katić, 1996; Carelli et al., 1995; Ozaki et al., 1991; Gutiérrez et al., 1998).

  - Controle utilizando Lógica Nebulosa (*Fuzzy*);
    (Kiguchi and Fukuda, 1997; Lin and Huang, 1998; Corbet et al., 1996; Liu, 1995).
  - Controle combinando as últimas duas técnicas anteriores;
    (Kiguchi and Fukuda, 1996; Kuo, 1997; Lightbody and Irwin, 1997).

- Controle combinando técnicas convencionais com agentes "inteligentes"
  – controles de concepção "híbrida", melhor dito talvez: integrada. (Battistela, 1999; Yilditrim and Sukkar, 1996; Sundareshan and Askew, 1997; Kwan, 1998; Song and Chu, 1998).

2.5 Controle Adaptativo de Força

Algumas técnicas convencionais para controle de força, estão baseados na modelagem do robô, do meio ou da interação robô-meio. A **desvantagem** em requerer **modelos exatos da dinâmica do sistema** limitam a utilização destas técnicas uma vez que durante a operação do robô, a carga no efetuador pode variar em função da tarefa prevista para o manipulador em questão e pode não ser conhecida com exatidão. Industrialmente é comum um robô manipular peças com diferentes características (peso, tamanho, rigidez, etc.. – o que modifica momentos de inércia e centros de massa) e por isso, para fazer frente a esta realidade se fazem necessários controles avançados. Para tentar lidar com este problema, estratégias de controle adaptativo têm sido desenvolvidas porque à *priori* não requerem informação de parâmetros desconhecidos, como a massa de carregamento do efetuador (Ge et al., 1997).

O objetivo básico do controle de força adaptativo é tentar manter um desempenho consistente do sistema de controle mesmo na presença de parâmetros desconhecidos do robô ou do meio. O controlador adaptativo possui a capacidade de monitorar seu próprio desempenho e modificar sua ação de controle para melhorar seu próprio desempenho (Narendra, 1997). Um controlador deste tipo é indicado para o uso em sistemas onde um certo nível de incertezas quanto ao processo à ser controlado está presente e se deseja manter um desempenho satisfatório.

Baseado nas definições dos métodos de controle de força convencionais, um controle de força adaptativo incorpora certas estratégias adaptativas de forma a manter uma desejada rigidez, impedância, admitância quando estão presentes parâmetros desconhecidos do robô

e há contato com o meio. Desta forma um desejado seguimento de posição e de força pode ser garantido.

Normalmente, existem duas categorias de estratégias adaptativas para controle de força em robôs: os métodos (1) indireto e (2) direto. No método indireto se atualiza uma estimativa para os parâmetros incertos do modelo dinâmico do sistema (robô) – cálculo da matriz regressora: $Y(q, \dot{q}, \ddot{q})$. Um exemplo de lei do controlador adaptativo indireto de posição no espaço de juntas pode ser dado por (Sciavicco and Siciliano, 1996):

$$\begin{aligned} u &= \hat{B}(q)\ddot{q} + \hat{C}(q, \dot{q}) + \hat{F}\dot{q} + \hat{g}(q) + K_d\,\sigma \\ &= Y(q, \dot{q}, \ddot{q})\hat{\pi} + K_D\,\sigma \end{aligned} \tag{2.18}$$

onde $Y(\cdot)$ se refere à matriz regressora, composta de termos que variam apenas com a posição, velocidade e aceleração robô; $\hat{\pi}$ representa o vetor de estimativa de parâmetros do robô de acordo com os termos $\hat{B}$, $\hat{C}$, $\hat{F}$ e $\hat{g}$ da equação dinâmica direta do manipulador (termos estimados); u se refere ao vetor de torque à ser enviado às juntas do robô; o termo K_d se refere a uma ação PD sobre o erro definido como: $\sigma = \dot{\tilde{q}} + \Lambda\,\tilde{q}$, que inclui além do erro de velocidade ($\dot{\tilde{q}}$) uma ponderação (Λ) sobre o erro de seguimento de trajetória ($\tilde{q}$). Segundo Sciavicco and Siciliano (1996) o controlador de posição retratado na equação 2.18 é globalmente assintoticamente estável.

Com base na estimativa de parâmetros realizada, um controlador baseado em dinâmica inversa tenta reduzir os erros de seguimento do sistema. Já no caso de uma estratégia adaptativa direta, os ganhos de um controlador convencional são diretamente atualizados com base em alguma função relacionada com parâmetros de erro. O objetivo de ambas estratégias é forçar que um parâmetro relacionado com erros (função "custo") convirja para zero. Como o projeto de um controlador adaptativo indireto exige o conhecimento preciso da estrutura de todo o robô (controle de posição apenas) e robô-meio (caso de controle de força), na prática isto é freqüentemente difícil de ser obtido (Zeng and Hemani, 1997) (principalmente os termos relacionados com matriz de inércia, B e forças de atrito envolvidas, F), a estratégia de controle adaptativo direta acaba sendo a mais utilizada no controle de força de robôs. A figura 2.11 mostra uma estrutura genérica para controle adaptativo de força.

Na figura 2.11, o bloco "mecanismo adaptativo" corresponde ao esquema de adaptação ou ao estimador de parâmetros. Exemplos de técnicas de controle adaptativo indireto

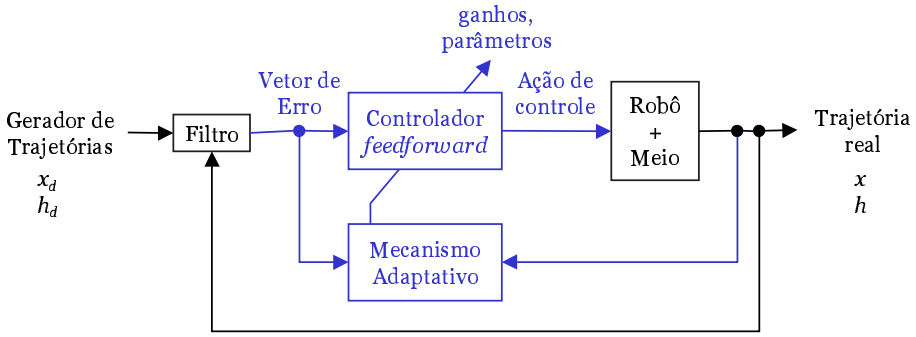


Figura 2.11: Controle Adaptativo genérico de Força.

(também conhecidas pela sigla "IA") podem ser vistas em (Yabuta et al., 1988; Craig, 1984), Controle de Força Explícito IA (Carelli et al., 1990a), Controle por Impedância IA (Carelli et al., 1990a; Sinh and Popa, 1995); e técnicas de controle adaptativo direto ("DA") podem ser vistas em: Controle Adaptativo Direto (DA) por Admitância (Seraji, 1994), Controle Adaptativo Direto por Impedância (Colbaugh et al., 1993), Controle Adaptativo Direto de Posição e Força (Carelli et al., 1990b; Seraji, 1987).

Para finalizar, o objetivo do controle adaptativo de força é projetar uma ação de controle para o robô de forma a que certos **objetivos de controle** sejam alcançados mesmo na presença de parâmetros incertos do robô ou do meio, tais como:

Controle por Rigidez :

$$\lim_{t\to\infty}(x_d - x) = -K_A^{-1}\, h \tag{2.19}$$

Ou seja, busca encontrar um ponto de equilíbrio ($\tilde{x}$) no qual o robô realiza um certa ação de controle de força baseado na estimativa da constante de rigidez do meio (K_A).

Controle por Impedância :

$$\lim_{t\to\infty}(x_d - x) = -[Ms^2 + Ds + K]^{-1}\, h \tag{2.20}$$

Neste caso, a idéia é tentar se definir o perfil dinâmico da resposta do contato do robô com o meio na forma de um sistema sub-amortecido de $2^{\underline{a}}$, como resultado de uma modelagem mecânica deste contato como um sistema massa (termo M) + mola (termo K - constante de rigidez desta mola) + fator amortecimento (termo D). Como os termos M, D e K são definidos em nível de projeto deste controlador, pode-se dizer que está sendo realizado um controle ativo de impedância do robô contra o meio. Cabe

ressaltar que esta estratégia de controle pode exigir a introdução de um efetuador com alguma capacidade elástica para compensar a rigidez do manipulador, principalmente no caso de contato contra meio muito rígido (como uma pedra de mármore por exemplo). A figura 2.12 mostra um caso genérico de controlador por de força implícito por impedância.

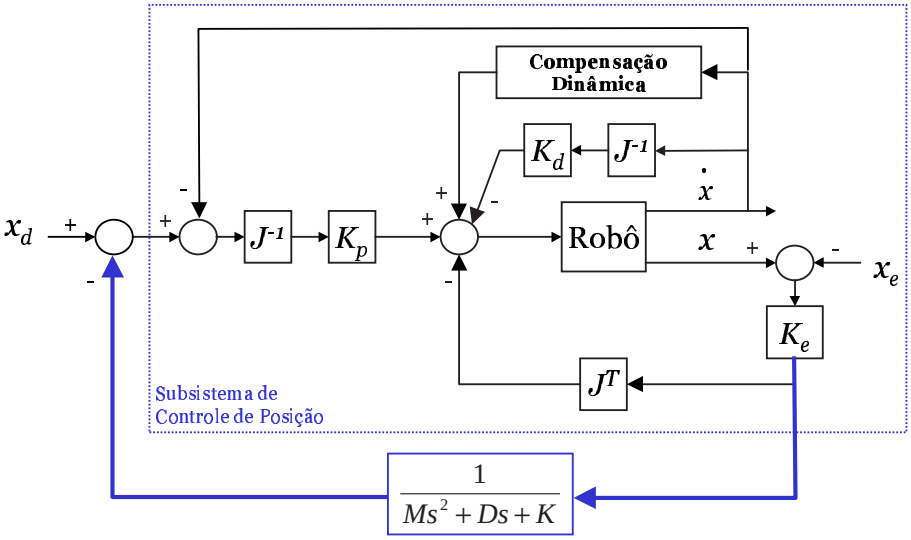


Figura 2.12: Exemplo de controlador de força implícito por impedância ativa.

Controle Híbrido :

$$\lim_{t\to\infty}(x_d - x) = 0 \qquad \text{e} \tag{2.21}$$

$$\lim_{t\to\infty}(I - S)(h_d - h) = 0 \tag{2.22}$$

Controle Híbrido por Impedância :

$$\lim_{t\to\infty}(\dot{x}_d - \dot{x}) = -Z^{-1}\,h \qquad \text{e} \tag{2.23}$$

$$\lim_{t\to\infty}(I - S)(h_d - h) = 0 \tag{2.24}$$

Controle Implícito de Força :

$$\lim_{t\to\infty}(x_d - x) = 0 \tag{2.25}$$

Controle Explícito de Força : inclui o controle por admitância com uma realimentação

explícita de força:

$$\lim_{t\to\infty}(h_d - h) = 0 \tag{2.26}$$

2.6 Controle Robusto de Força

O objetivo do controle robusto é se aproximar o máximo possível da dinâmica de resposta desejada para o sistema (por exemplo: impedância desejada) e tentar garantir estabilidade mesmo na presença de limitadas incertezas do modelo (dinâmicas não modeladas) do robô e do meio. A figura 2.13 mostra a estrutura genérica de um controle robusto de força.

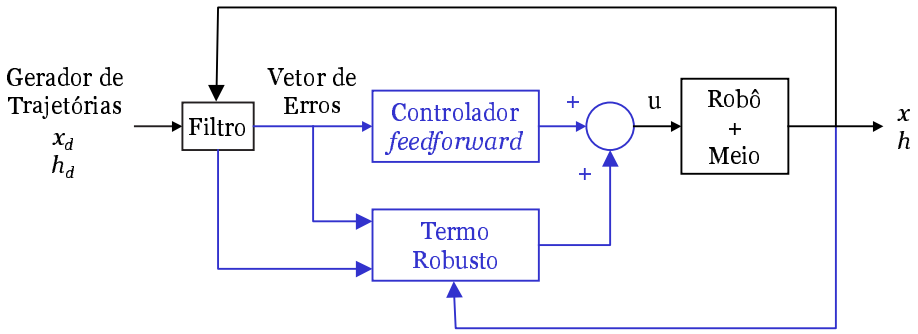


Figura 2.13: Controle Robusto genérico de Força.

Na figura 2.13 a entrada de comando inclui duas partes: 1) o termo robusto e 2) uma lei de controle convencional. Como lei de controle convencional é utilizado normalmente um PI, PD ou PID com esta parte da malha de controle complementada por compensação dinâmica (baseado em termos estimados). A dificuldade está em projetar uma boa lei de controle relativo ao termo robusto que tenta compensar as incertezas de modelagem. Por esta razão, conceitos de projeto de controlador por "modos deslizantes" (*sliding mode*) (Slotine, 1984; Sciavicco and Siciliano, 1996) são largamente empregados. Em termos de erro de informação e saída da realimentação, a lei de controle robusto é normalmente determinada utilizando o método direto de Lyapunov (usado para análise de estabilidade). Juntas, a lei de controle do controlador convencional mais o termo robusto visam garantir a dinâmica desejada definida à *priori*, enquanto se mantém a estabilidade em presença de erros de modelagem (incertezas paramétricas).

Os sistemas baseados em modos deslizantes são aqueles nos quais uma estrutura variável (o termo robusto) faz o sistema controlado deslizar sobre determinada superfície dentro do espaço de estado do sistema. Para tanto, a ação de controle deve ser projetada de

forma a fazer com que a trajetória de estados retorne a uma superfície de atração de erros sempre que dela se desviar. Uma vez que a trajetória de estados do sistema alcança esta superfície, mantém-se nela para os tempos subseqüentes deslizando até o ponto de equilíbrio (Ramírez et al., 2000). A figura 2.14 dá uma idéia da trajetória de erro (ϵ) para um caso de projeto de superfície de deslizamento bidimensional. Percebe-se que desde o instante inicial ($\epsilon(0)$), pode ser visualizado que a trajetória do erro é atraído para um hiperplano de deslizamento (uma linha, com $z = 0$) que tende para a origem do estado de espaços de erro conforme o tempo avança (Sciavicco and Siciliano, 1996). Depois que o sistema atinge o modo deslizante, seu movimento passa a ser definido pelas características da superfície na qual está deslizando tornando-se insensível a variações paramétricas e não linearidades. Entretanto, na prática, limitações físicas dos elementos que fazem parte da malha completa de controle (como velocidade de resposta dos atuadores, atrasos de resposta e zonas mortas em juntas) acabam fazendo com que o sinal de controle comute rapidamente, numa freqüência limitada, devido justamente ao termo robusto, o que faz com o sinal de controle oscile de forma a manter a trajetória de erro dentro do subespaço de deslizamento, ocasionando o que é conhecido na prática como efeito de *chattering*(Sciavicco and Siciliano, 1996). Este efeito pode ser atenuando modificando ligeiramente a lei de controle robusto de forma a garantir um chaveamento mais suave da ação de controle relativa ao termo robusto, que mesmo que não garanta mais a convergência do erro de seguimento de trajetórias para zero, ainda assim seja capaz de garantir o erro dentro de uma faixa limitada (camada limite). Maiores detalhes sobre o projeto de controladores à estrutura variável de robôs interagindo com ambientes passivos pode ser visto em (Amaral, 2000).

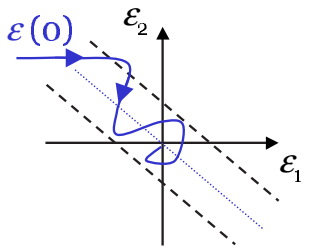


Figura 2.14: Trajetória do erro para controle robusto.

2.7 Controle de Força por Algoritmos de Aprendizado

Controle por Aprendizado têm sido introduzido independentemente de outras técnicas (Arimoto et al., 1984; Song and Chu, 1998). Recentemente foi implementado para controle híbrido de posição/força (Jeon and Tomizuka, 1993; Lucibello, 1993; Pandian and Kawamura, 1996; Battistela, 1999). A figura 2.15 mostra o princípio do algoritmo de aprendizado para controle de força de robô.

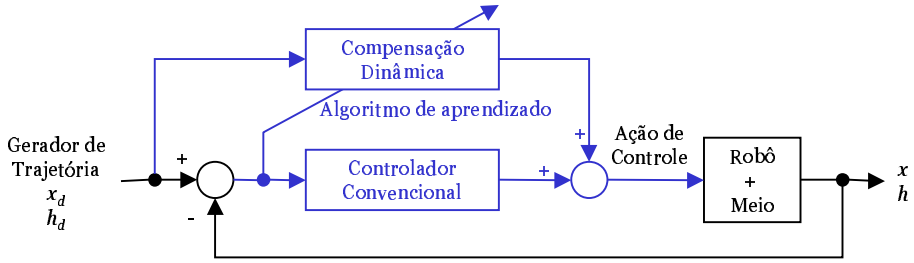


Figura 2.15: Princípio do algoritmo de aprendizado para controle de força de robô

O algoritmo de aprendizado é introduzido no controle híbrido de posição/força quando o robô executa a mesma tarefa repetidamente. Esta estratégia pode melhorar substancialmente o desempenho do sistema robótico sendo controlado. O algoritmo emprega erros de posição, velocidade e aceleração ou erro de força para aprender o comando de entrada requerido para executar a tarefa. Isto garante a convergência das trajetórias de erro, tanto de posição quanto de força, bem como robustez, para incertezas nos parâmetros e perturbações suficientemente pequenas (Zeng and Hemani, 1997; Song and Chu, 1998).

2.8 Resumo de Controle Avançado de Força

A tabela 2.2 faz uma comparação mostrando semelhanças e diferenças nos métodos de controle avançado de força descritos.

2.9 Conclusão

As seções anteriores trataram sobre técnicas de controle tradicionais e avançadas, com resumo nas tabelas 2.1 e 2.2. Com base nos resultados de técnicas ainda sendo pesquisadas, percebe-se que mais trabalhos são necessários envolvendo os seguintes aspectos (Zeng and Hemani, 1997):

Algoritmo		Objetivo do Controle	Técnica de Controle	Teoria Aplicada	Aplicado no caso de
Controle Adaptativo	Indireto	Convergência do Parâmetro de erro	Estimativa do parâmetro	Principio Adaptativo	Existência de Parâmetros incertos
	Direto	Convergência da Trajetória do Erro	Esquema de Adaptação		
Controle Robusto		Convergência da trajetória do erro	Lei de controle robusto	Modo de "controle deslizante", Estabilidade com Entradas-Saídas Limitadas	Existência de incertezas na modelagem
Controle por Aprendizado		Compensação completa de termos desconhecidos	Compensação direta (*feedforward*)	Princípio de aprendizado	Execução de tarefa repetitiva

Tabela 2.2: Comparação de métodos avançados de controle de força.

- Filtros e estimadores (de estado) mais eficientes para permitir algoritmos mais sofisticados;
- Pesquisa de melhores teorias de estabilidade para permitir decisões quanto a estratégia de realimentação que deve ser empregada para cada tarefa do robô;
- Capacidade de aprendizado mais acelerada capaz de lidar com mudanças não previsíveis nos parâmetros do robô (efeitos de carregamento) e do ambiente (desgaste de ferramenta, quebra de ferramenta);
- Melhor robustez para lidar com restrições desconhecidas e distúrbios impostos pelo ambiente.

Capítulo 3

Controle Inteligente

3.1 Introdução

As equações dinâmicas dos robôs manipuladores podem ser obtidas através do método de Lagrange baseado na conservação de energias (Burton, 1986, Cap. 6) ou do método de Newton-Euler (Burton, 1986, Cap. 2) que se focaliza no balanceamento entre as forças e momentos (Watanabe, 1996; Sciavicco and Siciliano, 1996; Asada and Slotine, 1986). Estas equações são de natureza não linear quando levam em consideração os ângulos nas juntas.

Abordagens convencionais para controle de sistemas não lineares têm se apoiado na teoria de controle linear utilizando basicamente um modelo linearizado em torno de um ponto de equilíbrio ou operação (Sciavicco and Siciliano, 1996; Watanabe, 1996). As soluções envolvem manipulações das equações dinâmicas do manipulador ou manipulador-meio, e especificações das leis de controle baseado em critérios de estabilidade como as funções de Lyapunov (Sciavicco and Siciliano, 1996; Lewis et al., 1993).

O problema é que estas técnicas exigem a estimação de parâmetros, isto ainda assumindo-se que estamos trabalhando com um modelo bem estruturado (conhecido). É difícil se estimar com exatidão o termo relativo ao momento de inércia e ao atrito. Normalmente, para controle de posição apenas não se leva em consideração os efeitos de atrito para um mecanismo muito complexo. No caso do controle do robô interagindo com o meio, a modelagem se torna ainda mais complexa e inexata, seja pela falta de conhecimentos sobre a tarefa à ser realizada, seja pela adoção de certas hipóteses relativas à tarefa em questão e em relação ao meio.

Os **controladores adaptativos** visam suprimir os problemas advindos de uma modelagem com parâmetros inexatos. Os controladores adaptativos e outros baseados em modelagem constituem-se nas estratégias de controle normalmente adotadas, capazes de lidar com parâmetros inexatos, incertezas relacionados às características do manipulador e do meio e variações de carga (variações paramétricas). Porém um controlador adaptativo normalmente exige que os parâmetros desconhecidos do sistema sejam lineares e que ainda se realize o cálculo (tedioso) da matriz regressora, $Y(q, \dot{q}, \ddot{q})$ [1], referente ao manipulador em questão (Kim and Lewis, 1999). Não está claro que estratégias de controle adaptativo e baseados em modelagem são capazes de lidar com incertezas advindas da presença de modos de alta freqüência presentes no modelo (dinâmicas que normalmente não são modeladas), atrasos de tempo que não foram considerados e atritos que normalmente são não lineares (Er and Liew, 1997).

É necessário então se desenvolver um sistema de **controle não linear robusto** para combater estas incertezas (Watanabe, 1996).

Estruturas de controle capazes de aprendizado foram desenvolvidas, de forma a melhorar o desempenho do sistema, mesmo frente a variações paramétricas. Por exemplo, a técnica de **aprendizado reforçado** (*reinforcement learning*), pode ser útil no caso de uma tarefa repetitiva com um robô, quando o aprendizado pode ser interessante para melhorar o desempenho deste. Neste último caso, a única desvantagem desta técnica é estar limitada à operações que são repetitivas.

Além disso, é interessante se deixar o robô preparado para lidar com mudanças no ambiente, e assim o sistema de controle deve ainda embutir alguma inteligência em si mesmo, relativa às ações que são produzidas simultaneamente, produzidas de acordo com os diversos estados do ambiente, decisões que eventualmente sejam necessárias num ambiente robótico distribuído competitivo ou cooperativo (Watanabe, 1996). Para estes casos, uma forma de solucionar estes problemas é utilizar estruturas de controle inteligentes (Passino, 1995) como: controle cooperativo baseados em redes neurais (Cui and Shin, 1996) e controle nebuloso.

As abordagens da área de controle não linear e controle inteligente têm criado um impacto significativo para controle de robôs e sistemas mecatrônicos, em contraste com

[1] A equação dinâmica de um robô manipulador pode ser descrita na forma: $B(q)\ddot{q} + C(q, \dot{q})q + F\dot{q} + g(q) = Y(q, \dot{q}, \ddot{q})\pi$, onde: $Y(q, \dot{q}, \ddot{q})$ se refere à matriz regressora, composta de termos que variam com a configuração do robô, e π se refere aos parâmetros que independem da configuração do robô, mas que sofrem a influência de variações paramétricas deste. O controlador adaptativo tenta estimar os parâmetros de π.

técnicas de controle convencionais aplicadas em sistemas lineares.

Watanabe (1996) define **controle inteligente** como uma abordagem que especifica um algoritmo de controle através da emulação de algumas propriedades de um sistema biológico inteligente. Os controladores não lineares como o do torque computado (Sciavicco and Siciliano, 1996; Lewis et al., 1993), controlador baseado no método de Lyapunov, controlador por "modo deslizante" (*sliding mode control*) (Sciavicco and Siciliano, 1996), e controlador por linearização normalmente utilizam informação sobre o modelo do objeto à ser controlado, enquanto um controle inteligente usualmente não utiliza esta informação.

Atualmente as **redes neurais** têm sido empregadas em aplicações de controle de sistemas que são tanto complexos quanto não-lineares ou nos casos em que se tenha de lidar com incertezas (Narendra, 1997). Particularmente estão sendo aplicadas nos casos em que:

1. Modelos matemáticos do processo a ser controlados são pobres. Em alguns casos, não existem modelos. Entretanto, dados do processo estão disponíveis enquanto o processo está em operação. E modelos lineares do processo são inadequados para representar o sistema com suficiente exatidão.

2. O sistema está em operação e possui desempenho satisfatório com o controlador existente (normalmente um controlador linear) e quando o domínio de operação é pequeno. Entretanto, por questões econômicas o domínio de operação aumentou tornando o sistema reconhecidamente não linear. Como conseqüência, o desempenho do sistema se degrada consideravelmente, requerendo técnicas de controle não linear.

Mas levando em conta, tanto considerações teóricas quanto práticas, parece ser vantajoso manter componentes lineares e não-lineares em paralelo, tanto para identificação quanto controle, com os componentes não- lineares sendo representados pelas redes neurais (Narendra, 1997). Há de ser notado os curtos períodos de treinamento e aumento na precisão obtidos com abordagens integradas (Narendra, 1997).

Alternativamente outra opção "inteligente" seria usar técnicas de **lógica *fuzzy*** para desenvolver outros tipos de controladores que não requerem nenhuma modelagem mas que podem tirar proveito do conhecimento acumulado por especialistas humanos em controle, já que esta técnica permite lidar com conhecimento expresso na forma de regras lingüisticas um tanto vagas. Por exemplo, para especificar a força desejada em tarefas como de

polimento de uma peça, um especialista humano normalmente controla a força do contato de acordo com as propriedades da peça (rigidez, atrito) e da área envolvida no contato entre a ferramenta de polimento e a peça, através de seu conhecimento *a priori* (Kiguchi and Fukuda, 1997).

Entretanto, o projeto sistemático de controladores *fuzzy* é difícil (Kiguchi and Fukuda, 1997). Mesmo depois que as regras e funções de pertinência do controlador tenham sido implementadas, ainda se fazem necessários pequenos ajustes para tornar o controlador mais perfeito, além dos usuais ajustes de escalas necessários entre as variáveis de entrada e de saída de controladores *fuzzy*. Lembramos que as redes neurais possuem capacidade de aprendizagem e adaptação a novos ambientes, mas podem exigir uma etapa de pré-aprendizado (aprendizado *off-line*, depende da estrutura do controlador). E vão ainda exigir um período para adaptação, que num pior caso pode comprometer a integridade física da peça, superfície de contato ou mesmo componentes do robô (também aqui, esta desvantagem depende da estrutura do controlador neural, no caso de uma rede neural integrada com um controlador *feedforward*, se espera que este último assuma o controle durante a fase de aprendizado/adaptação da rede neural). Assim uma outra opção interessante pode ser combinar-se um controlador neural com um controlador *fuzzy*, resultando num **controlador neuro-*fuzzy*** (Kiguchi and Fukuda, 1997).

Assim, **neste capítulo** serão revistas algumas abordagens de controladores baseadas em redes neurais e lógica *fuzzy*, incluindo comentários acerca as propriedades inteligentes de cada uma destas abordagens que normalmente não utilizam ou não se baseiam em modelos matemáticos do sistema à ser controlado. E o capítulo finaliza ainda descrevendo abordagens neuro-*fuzzy* e aprendizado reforçado para o caso de controle de força em robôs manipuladores.

3.2 Redes Neurais em Controle

3.2.1 Introdução

Desde 1980, vários autores mostraram que redes neurais do tipo *feedforward* multicamadas são capazes de aproximar uma função contínua qualquer num conjunto compacto e assim, desde lá, muitas redes deste tipo encontraram variadas aplicações em muitas áreas

tanto para aproximação de funções (Girosi and Poggio, 1993; Hornik et al., 1993; Rao and Gupta, 1993) quanto de reconhecimento de padrões. Em 1990, (Narendra and Parthasarathy, 1993) propuseram o uso de redes neurais como componentes em sistemas dinâmicos para controle prático de sistemas lineares, envolvendo a combinação de modelagem matemática, computação e experimentação. E sugeriram que tais técnicas também deveriam ser empregadas para o controle de sistemas não-lineares. As redes neurais começaram a ser muito empregadas para modelagem, identificação e controle (Warwick, 1996; Narendra and Mukhopadhyay, 1993; Antsaklis, 1992; Narendra and Parthasarathy, 1993). Desde meados da década de 90, os estudos se concentraram no controle de sistemas não-lineares usando redes neurais. E desde 1997 na solução de problemas industriais práticos que necessitavam de redes neurais para controle e identificação em tempo-real (Narendra, 1997).

Possíveis aplicações de redes neurais em robótica incluem diferentes propósitos que podem ser classificados de acordo com o tipo hierárquico de controle do sistema robótico, isto é, as redes neurais podem ser aplicada em nível de controle estratégico (planejamento de tarefas), ao nível de controle tático (planejamento de trajeto - *path planning*) e ao nível executivo de controle (*path control*) (Katič and Vukobratovič, 1996). Todos estes problemas de controle em diferentes níveis hierárquicos podem ser resumidamente formulados em termos de um problema de otimização ou de associação de padrões ou como um problema de aproximação de funções (Katič and Vukobratovič, 1996). Um problema de otimização pode ser o de planejamento do trajeto para robôs autônomos, visão estéreo e planejamento da tarefa (*task planning*). Como problema de associação de padrões podemos destacar o controle de um sensor ou de um motor, o controle do movimento voluntário e mais recentemente o controle de uma articulação através de um modelo cerebelar (*CMAC = Cerebellar Model Articulatio Controller* - ver (Albus, 1975; Commuri and Lewis, 1997)) (Katič and Vukobratovič, 1996). Como aproximador de funções, uma rede neural poderia modelar as relações entre entradas/saídas envolvidas com a cinemática ou dinâmica de um robô.

Inspiração Biológica

Redes neurais artificiais, ou mais simplesmente RNs, como passaremos a utilizar doravante, são biologicamente inspiradas. O princípio básico das redes neurais artificiais parte da idéia de se modelar individualmente células nervosas, os neurônios e seu comporta-

mento. Um neurônio biológico passa informações através de um axônio (normalmente único) e recebem informações (a nível eletro-químico) de vários dendritos. As junções entre o axônio de uma célula e o dendrito de outra constituem as sinapses. Um neurônio artificial modela os axônios e os dendritos através de conexões e as sinapses através de ponderações ou pesos de ajuste, de uma forma adequada e então conecta estes modelos de uma maneira altamente paralela, de forma similar ao que ocorre realmente nas redes nervosas biológicas, o que possibilita um mecanismo de processamento complexo, com características não lineares que ainda possui capacidade de aprendizado (Warwick, 1996).

Numa rede neural biológica as sinapses inicialmente são determinadas pelo genótipo daquele sistema/organismo. Novas sinapses são criadas ao longo do tempo e as outras já existente são melhor ajustadas no decorrer da vida de um organismo vivo, caracterizando o que se conhece como aprendizado. Note porém que diferente do que se consegue implementar com uma rede neural artificial, numa rede biológica real podem existir mais de 100 bilhões de células nervosas interconectadas trabalhando todas ao mesmo tempo (em paralelo), o que possibilita o elevado poder de processamento do cérebro humano.

Assim uma rede neural artificial é caracterizada primeiro isoladamente pelos seus neurônios, depois pelas conexões entre eles (o que define a arquitetura ou topologia desta rede) e por fim pelo seu esquema de aprendizado (Maren, 1990).

As redes se distinguem basicamente através do seu método de aprendizado (maneira de atualizar seus pesos sinápticos). Existem duas formas de aprendizado para as RNs:

1) **Aprendizado Supervisionado**: existe um "professor"que na fase de aprendizado indica para a rede o quão bem está sendo seu desempenho (aprendizado por reforço ou *reinforcement learning* - ver (Sutton and Barto, 1998; Barto, 1989)) ou este professor indica para a rede qual saída correta era esperada ("*fully supervised learning*") (Sarle, 2002, Part 1).

2) **Aprendizado não supervisionado**: a rede é autônoma, ela simplesmente examina os dados apresentados a ela, descobre algumas propriedades presentes no conjunto de dados e aprende a refletir estas propriedades nas suas saídas (faz uma associação de padrões). O que exatamente são estas propriedades, que a rede aprende a reconhecer, depende do modelo particular de rede neural e método de aprendizado sendo utilizado. Normalmente a rede aprende uma forma compacta de representar os dados de entrada (Sarle, 2002, Part 1). Existem diferentes tipos de arquiteturas de rede para aprendizado não supervisionado

em robótica (Katič and Vukobratovič, 1996).

É importante perceber que não existe nenhum método para treinar as redes neurais que possa magicamente extrair informação que não esteja contida no conjunto de dados para seu treinamento (Sarle, 2002, Part 1).

Conforme a aplicação, diferentes tipos de redes neurais existem. Um resumo dos tipos mais comuns de redes neurais artificiais pode ser encontrado em (Sarle, 2002; Lippman, 1987; Widrow and Lehr, 1990). As características de uma RN dependem da aplicação desejada.

3.2.2 RNs em Controle (Geral)

Na prática, as RNs são úteis em problemas de classificação e aproximação ou mapeamento de funções que envolvem alguma imprecisão mas para os quais é difícil ou demorado a obtenção de regras. Normalmente muitos dados para treinamento estão disponíveis. Quase toda função vetorial finita dimensionalmente e contida num conjunto compacto, pode ser aproximada por uma rede *feedforward* com uma certa precisão tendo-se suficientes dados disponíveis e recursos operacionais (Sarle, 2002).

As redes neurais podem ser empregadas para:

- Mapear funções não-lineares arbitrárias. Sendo assim são indicadas para lidar com sistemas não-lineares (Narendra, 1997; Warwick, 1996; Gupta and Rao, 1993a; Gupta and Rao, 1993b; Hunt et al., 1993; Hunt and Sbarbaro, 1993);

- Realizar operações com múltiplas variáveis, dada sua habilidade de mapear correlações entre muitas variáveis de entrada e de saída (Warwick, 1996; Cui and Shin, 1996);

- As RNs podem ser treinadas tanto *off-line* e subseqüentemente utilizadas e/ou treinadas e empregadas *on-line*, ou então podem ser treinadas *on-line* como parte de um esquema de controle adaptativo (Zhihong et al., 1998; Noriega and Wang, 1998; Jagannathan, 1996) ou utilizadas para identificar sistemas em tempo-real (*real-time*) (Narendra and Parthasarathy, 1993);

- As RNs são inerentemente dispositivos de processamento paralelo com características tolerantes a faltas. O que possibilita um processamento rápido dos dados, aliado a

uma estrutura que possui degradação suave em caso de falhas (Warwick, 1996; Soucek, 1989).

A maior parte dos problemas complexos em controle lida com sistemas complexos, não-lineares e com incertezas. As redes neurais podem lidar com estes tipos de problemas. Sua natureza de inerente paralelismo a faz ser atrativa para lidar com casos complexos. Sua habilidade para mapear funções não lineares e a disponibilidade de métodos para ajustar seus próprios parâmetros com base nos dados de entrada/saída, as fazem ser particularmente atrativas quando não-linearidades desconhecidas estão presentes num sistema (Narendra, 1997; Katič and Vukobratovič, 1996).

Desde os anos 90 mais projetos significativos e ambiciosos estão aplicando redes neurais para lidar no controle em tempo-real de sistemas não lineares (Narendra, 1997).

Como uma rede neural é capaz de realizar um mapeamento não linear entre as suas entradas e saídas, elas são utilizadas para realizar aproximações, em problemas da área de controle, excetuando a identificação e controle de sistemas (Narendra, 1997).

Está disponível alguma literatura acerca do desempenho (Demath and Beale, 1999) e muita a respeito dos diferentes tipos de RNs (Demath and Beale, 1999; Sarle, 2002; Lippman, 1987; Knight, 1990). Entretanto, as redes mais populares e que parecem ser as mais promissoras para área de controle parecem ser as redes:

a) **Redes *perceptron* Multicamadas (MLP)** (Tveter, 1998; Sarle, 2002; Jondarr, 1996; de Leon Ferreira de Carvalho et al., 1998; Widrow and Lehr, 1990);

b) **Redes de Função de Base Radial** (RBF = *Radial Basis Networks* (Orr, 1999; Gabrijel and Dobnikar, 1997; Orr, 1996).

c) **Redes CMAC** (*Cerebellar Model Articulation Controller*): que vêm despertando um interesse mais recente (Commuri and Lewis, 1997; Albus, 1975);

3.2.3 Redes Multicamadas Perceptron

São redes neurais formadas por múltiplas camadas de *perceptrons* não-lineares (ver figura 3.1). Elas possuem um certa estrutura e forma de implementação, permitindo que se estabeleçam relações não lineares entre sua camada de entrada e de saída.

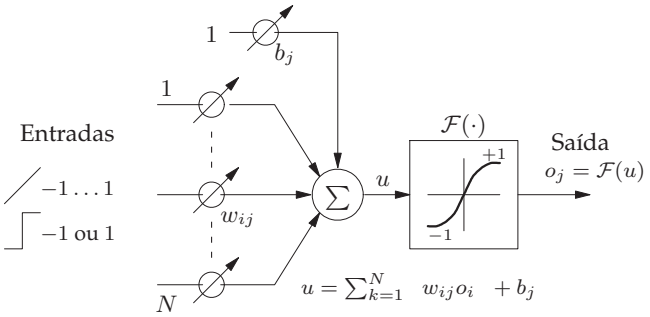


Figura 3.1: Neurônio artificial simplificado do tipo *perceptron*.

A figura 3.2 mostra uma rede MLP com duas camadas invisíveis (*hidden layers*). Note que a primeira camada, a de entrada é passiva pois apenas recebe os sinais de entrada os redistribui para cada neurônio da camada seguinte. As camadas intermediárias, são ditas invisíveis porque não são diretamente acessíveis dos nós de entrada e de saída da rede. A última camada da Fig. 3.2 seria a camada de saída da rede.

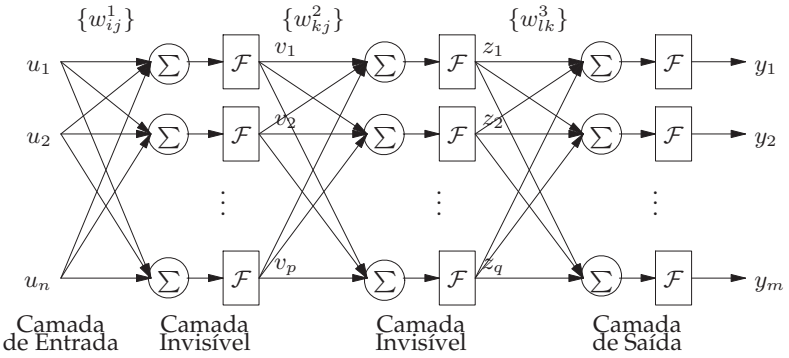


Figura 3.2: Rede MLP de 2 camadas invisíveis.

Uma rede MLP é do tipo *feedforward* porque é constituída por um certo número de camadas de neurônios (ou elementos processadores) que se intercomunicam, no sentido da camada de entrada para a de saída (os dados fluem da camada de entrada para a de saída).

Cada neurônio de uma certa camada possui uma ligação ponderada (sinapse) com todos neurônios da camada anterior à este. A informação que chega de cada um dos neurônios é ponderada via matriz de pesos associada, w_{ij}, somada e passada por um operador não linear, ou função de ativação, $\mathcal{F}(\cdot)$, ver figura 3.2.

Note que opcionalmente associado à cada neurônio, pode existir um *offset*, ou "*bias*", representado pelo vetor b_i (ver figura 3.1). O vetor de *bias* é necessário porque sem ele os hiperplanos de separação relacionados ao mapeamento entre os dados da camada de entrada

e da camada de saída obrigatoriamente passarão pela origem dos espaços definidos pelas entradas (Sarle, 2002). Uma rede MLP com função de ativação sigmoidal com N entradas, definirá um hiperplano de dimensão N. Os pesos sinápticos entre a entrada e saída de uma camada determinam onde este hiperplano repousa. Note ainda que Hornik et al. (1993) afirmaram que a propriedade de "aproximador universal"de uma rede MLP deixa de ser válida se as unidades de *bias* forem omitidas (Sarle, 2002, Part 2). Mas Hornik et al. (1993) mostraram que uma condição suficiente para manter a propriedade do aproximador universal sem *bias* é que as derivadas das funções de ativação existam quando próximas da origem, o que quer dizer que, com funções de ativação normalmente sendo sigmoidais, unidades de *bias* com valores fixos não nulos podem ser utilizados sem serem treinadas. Outro problema com redes MLP sem *bias* é que normalmente elas apresentam um mau comportamento durante seu treinamento e normalmente são mais difíceis de treinar que as redes que utilizam *bias* (Sarle, 2002, Part 2).

Resumidamente, a informação que entra numa rede MLP é propagada da camada de entrada para a de saída de forma direta (*feedforward*) da seguinte forma:

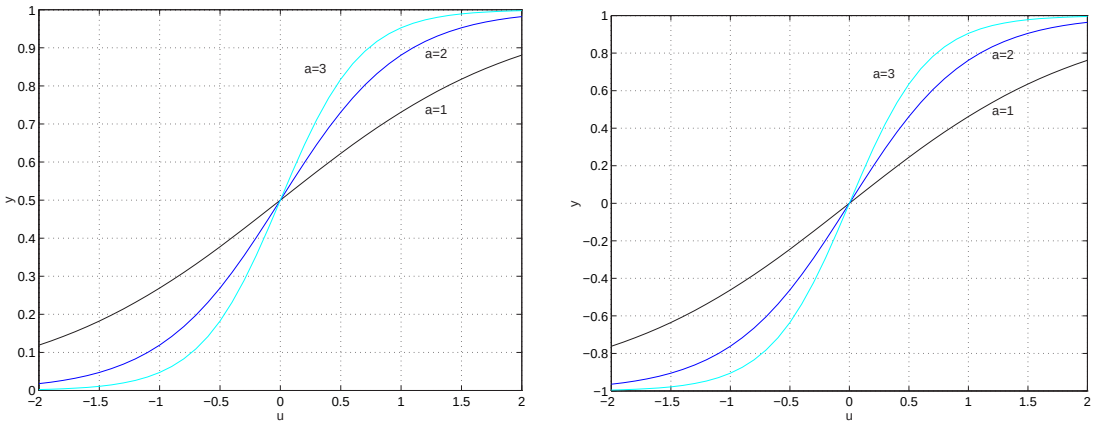
$$v_j = \sum_{k=1}^{N} \Big(w_{ij} y_i \Big) + b_j \tag{3.1}$$

onde v_j representa a soma ponderada dos N sinais de entrada do neurônio j; w_{ij} representa o valor sináptico associado à conexão do neurônio i para o neurônio j; y_i representa a saída do i-ésimo neurônio; e b_j representa o *bias* associado ao neurônio j. O sinal de saída do neurônio j é então determinado aplicando-se o resultado da soma ponderada das suas entradas, u_j, numa função ativação:

$$y_j = \varphi_j(v_j)$$

Para cada somatório ponderado de um neurônio, existe uma função de ativação, $\varphi_j(\cdot)$, que normalmente é um operador não linear ou no mínimo é não linear nos neurônios da camada intermediária de uma rede MLP para que não linearidades possam ser introduzidas no mapeamento entre as entradas e saídas criadas por uma rede MLP. Sem esta não linearidade, estas redes não seriam tão poderosas quanto as redes com *perceptrons* "planos", isto é, cuja função de transferência pode ser a função $Sgn(\cdot)$ ou uma equação da reta (Sarle, 2002).

A figura 3.3 mostra algumas funções de ativação tipicamente utilizadas em redes MLP.



(a) $\mathcal{F}(u) = \frac{1}{1+exp(-a \cdot u)}$ (b) $\mathcal{F}(u) = \frac{1-exp(-a \cdot u)}{1+exp(-a \cdot u)}$

Obs.: a representa o ganho do neurônio

Figura 3.3: Funções de ativação típicas de redes MLP.

Outras funções não lineares também podem ser empregadas, como as tangentes hiperbólicas $tanh$ e as funções gaussianas (mais próprias das redes RBF).

Note que apesar da função de ativação ser não linear é desejável que ela seja diferenciável, pois normalmente o algoritmo de aprendizado utilizado se baseia na derivada do erro entre o sinal de saída desejado e o gerado pela rede. Também é desejável que esta função seja limitada, ou seja, que sua saída esteja restringida numa faixa de valores, como é o caso da função sigmoidal ou tangente hiperbólica (diferenciáveis).

Redes neurais MLP normalmente empregam o algoritmo de aprendizado baseado na "Regra Delta Generalizada", algoritmo mais comumente conhecido como "*back propagation*"(Freeman and Skapura, 199?, Cap. 3) (Jondarr, 1996; Knight, 1990; Maren, 1990).

O algoritmo *back propagation* baseado no gradiente do erro entre a saída desejada e a gerada pela rede, é um método para minimizar erros. A qualidade da aproximação realizada por uma rede pode ser avaliada pela soma dos erros quadráticos de saída da rede:

$$SE = \frac{1}{2} \sum_j \left(t_j^p - y_j^p \right)$$

onde t_j^p corresponde ao valor desejado para a saída j, padrão de treino p, e y_j^p é a saída j computada para o padrão p. É desejável que a medida que a rede aprenda, $d_j^p \cong y_j^p$. A idéia por trás do algoritmo *back propagation* é diminuir o erro quadrático da saída da rede

ajustando seus pesos sinápticos usando o método do gradiente descendente.

Para o caso de uma rede MLP com apenas uma camada oculta, a cada passo de treinamento, cada uma das sinapses da camada de saída desta rede sofre o acréscimo:

$$w_{ij,novo} = w_{ij,anterior} + \eta \Delta w_{ij}$$

onde η corresponde a taxa de aprendizado sendo adotada e Δw_{ij} varia com o gradiente do erro de saída da rede, ou seja:

$$\Delta w_{ij} = -\frac{\partial SE}{\partial w_{ij}} \cdot y_i = -\delta_j \cdot y_i$$

$\frac{\partial SE}{\partial w_{ij}}$, correspondente ao gradiente de erro da rede e é calculado para cada neurônio (δ_j) da seguinte forma:

$$\delta_j = -e_j \cdot \varphi'_j(v_j)$$

onde: $e_j = t_j - y_j$ e $\varphi'_j(v_j)$ corresponde a derivada da função transferência de saída do j-ésimo neurônio da camada de saída da rede. Note que o sinal $\delta_j = -\frac{\partial SE}{\partial w_{ij}}$ (regra delta) usado para atualizar cada um dos pesos da rede é retropropagado da camada de saída em direção a camada oculta da rede MLP. O apêndice C traz mais informações sobre o algoritmo *back propagation*.

O algoritmo *back propagation* é suscetível de mau comportamento por causa dos mínimos locais que podem existir na superfície da gradiente do erro (também referenciado como "energia"da rede) (Maren, 1990; Knight, 1990). Neste caso a rede pode entrar num estado de paralisia durante seu treinamento, causado pelo próprio processo de otimização do erros pelo gradiente. Pelo próprio formato não monotônico da derivada da função transferência, se w_{ij} é deslocado numa direção de gradiente pequeno, o treinamento praticamente para num ponto conhecido como mínimo local, que não corresponde ao $w_{ij(min)}$ ou mínimo local (Lawrence, 1992; Maren, 1990; Knight, 1990). Normalmente é introduzido um termo denominado momento, α, com o objetivo de atenuar o problema do mínimo local, fazendo a rede escapar de alguns mínimos locais:

$$w_{ij,novo} = w_{ij,anterior} + \eta \Delta w_{ij} + \alpha \Delta w_{ij,anterior}$$

Mas a introdução de ruído randômico à rede numa técnica conhecida como *simmulated annealing* é mais eficiente pois provoca pequenos "saltos"nos valores das sinapses, possibilitando o deslocamento dos pesos para "vales"mais profundos na superfície da gradiente do erro (Lawrence, 1992; Knight, 1990; Aleksander and Morton, 1990).

Assim o algoritmo de aprendizado *back propagation* é conhecido por demandar muito tempo com o aprendizado (*time comsuming algorithm*) (Demath and Beale, 1999; Soucek, 1989; Maren, 1990; Knight, 1990; Sarle, 2002). Atualmente se buscam algoritmos de aprendizado mais velozes (Demath and Beale, 1999; Tveter, 1998; Verma, 1997; Jondarr, 1996; Riedmiller, 1994a) capazes de possibilitar aprendizado *on-line* em aplicações na área de controle. Entre elas podemos destacar os algoritmos: RPROP (Riedmiller and Braun, 1993; Riedmiller, 1994b) e QuickProp (Fahlman, 1988).

No nosso caso, de controle de força/posição em robôs manipuladores é desejável um tipo de rede neural com capacidade de treinamento *on-line*, e ainda que seja capaz de aprender da maneira mais rápida possível.

Entretando estudos na área de controle aplicado, indicam que redes neurais artificiais raramente podem ser utilizadas como "caixas-pretas" (*black-box*) (Warwick, 1996). Assim, uma aplicação *on-line*, exige cuidados na forma de empregar uma RN de forma à garantir estabilidade para o sistema.

3.2.4 Redes RBF

A estrutura de uma rede de função de base radial (RBF) é mostrada na figura 3.4 e ela consiste de apenas três camadas.

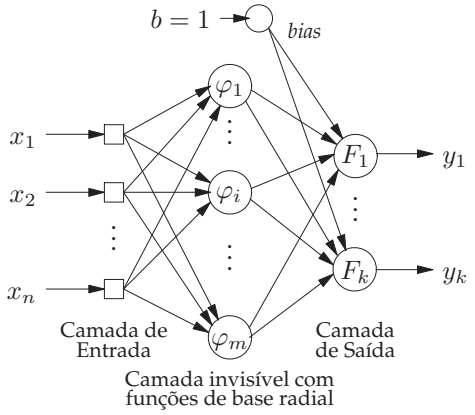
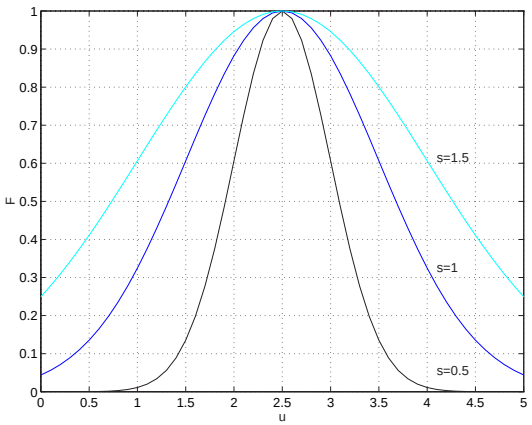


Figura 3.4: Estrutura básica de uma rede RBF.

Diferente das redes MLP, as camadas desta rede possuem diferentes funções. A primeira camada serve apenas para conectar o mundo real a este modelo de rede. A Segunda camada é a camada oculta que realiza a transformação não linear entre o espaço do vetor de entrada e o espaço de vetores interno que normalmente é de elevada dimensão. A última camada é a de saída e transforma o espaço de vetores interno num espaço de saída de forma linear. Os neurônios da camada oculta são funções de base radial. Estas funções são do tipo Gaussianas:

$$\varphi_\sigma(x) = e^{\frac{1}{2}\cdot\frac{\|x-c\|^2}{\sigma^2}} \qquad \sigma > 0$$

onde φ se refere à função gaussiana; c se refere ao centro da função gaussiana, que é treinável; σ está relacionado com a abertura da curva gaussiana (valores grandes tornam a curva mais aberta e valores menores estreitam a curva); x se refere ao vetor de entrada daquela função. Note que $(\|x - c\|)$ mede a distância entre os vetores de entrada e o centro desta função gaussiana, por exemplo, a distância Euclidiana entre os vetores $\vec{p}_1 = [x_1, y_1]$ e $\vec{p}_2 = [x_2, y_2]$ é dada por $d(\vec{p}_1, \vec{p}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ (Orr, 1996). A figura 3.5 dá uma idéia do aspecto da função gaussiana para diferentes valores do parâmetro σ.



Onde $c = 2.5$ e s da figura corresponde à σ.

Figura 3.5: Função Gaussiana.

A rede RBF é composta de m centros, de valor c_i associando cada ponto de entrada com seu centro mais próximo. Se for aplicada uma técnica de agrupamento dos pontos de entrada (*clustering*) para obter o valores destes centros, cada centro terá um diferente número de pontos associado consigo. O parâmetro que acaba determinando uma escala para as distâncias entre os pontos e os centros, σ_i, pode ser obtido heuristicamente, ou determinado pela escolha do valor do parâmetro p que indica o número de centros vizinhos à cada centro

individualmente, ou seja:

$$\sigma_i = \frac{1}{p}\sum_{j=1}^{m}\sqrt{\|c_j - c_i\|^2}$$

onde c_j se trata de um dos p vizinhos mais próximo ao centro c_i.

Os neurônios da camada de saída processam a função linear:

$$y_i = w_{i0}b + \sum_{j=1}^{m} w_{ij}\varphi_j(x) \qquad i = 1, \ldots, k$$

onde o índice i se refere ao neurônio i de saída; o índice j cobre todos os neurônios da camada invisíve (são m neurônios)l; b é a entrada de *bias* e w_{ij} é a conexão ponderada do neurônio j da camada oculta para o neurônio de saída i e w_{i0} representa o termo de *bias* (polarização).

O processo de aprendizado é divido em duas etapas por causa das diferenças entre a camada oculta e a de entrada. Primeiramente, a camada não linear oculta é determinada através de algum algoritmo de agrupamento (*clustering*) e então a camada de saída é otimizada ou pelo método da pseudo inversa (Gabrijel and Dobnikar, 1997; Keller, 1999) ou mais comumente, através do algoritmo de menor erro quadrático médio (LMS = *Least Mean Squared algorithm*) (Warwick, 1996):

$$w_i(k+1) = w_i(k) + \mu_i e(k)\varphi_i\big(x(k)\big)$$

onde μ_i é uma função positiva definida; $e(k) = y(k) - \hat{y}(k)$, onde $y(k)$ representa o dado de saída real correspondente à entrada $u(k)$.

As redes RBF são biologicamente inspiradas em receptores locais, por exemplo, as células de *Cochlear Stereocilia* são sensíveis à certas frequencias (Keller, 1999). Porque as funções gaussianas são locais, isto é, fornecem resposta significativa apenas na vizinhança próxima ao seu centro (Orr, 1996), as redes RBF são mais sensíveis aos dados de entrada próximos aos centros das suas funções gaussianas (Keller, 1999). Esta "sensibilidade"pode ser ajustada através do parâmetro σ. Quanto maior o σ (maior espalhamento da função gaussiana), menor a sensibilidade individual dos neurônios (Keller, 1999).

Pode-se mostrar que as redes RBF também podem ser consideradas aproximadores universais de função como no caso de uma rede MLP com funções de ativação sigmoidal

(Keller, 1999).

As redes RBF exibem boas propriedades de interpolação mas baixas propriedades de extrapolação, se comparadas com as redes MLP com funções de ativação sigmoidal. As funções sigmoidais são melhores para extrapolação porque elas seguem a função (que se deseja aproximar) e suas derivadas fora do espaço de treinamento (Keller, 1999). Porém (Warwick, 1996) cita como uma grande vantagem das redes RBF, o fato de seus tempos de aprendizado/treinamento serem aproximadamente 3 vezes menores do que o de redes MLP com algoritmo *back propagation*.

3.2.5 Tabela Resumo de Redes Neurais

A tabela 3.1 apresentada na página à seguir, resume as principais características de cada tipo de rede anteriormente analisada.

3.2.6 Detalhes de Implementação com Redes Neurais

Número de camadas e neurônios em cada camada

O número de camadas e de neurônios a serem utilizados em cada uma das camadas da rede, são geralmente determinadas por tentativa e erro. Mas Narendra (1997) perceberam que até duas camadas invisiveis são adequadas para a maior parte das aplicações em controle.

Número de redes neurais em sistemas multivariáveis

No caso de sistemas multivariáveis, uma escolha deve ser feita para decidir se é utilizado uma única rede neural com múltiplas saídas ou várias redes neurais com uma única saída. As duas abordagens apresentam vantagens uma em relação à outra. Narendra (1997) preferem usar várias redes neurais, uma para cada saída.

Problema de memória e processamento...

Uma questão genérica surge freqüentemente em situações que decisões devam ser tomadas levando em conta memória e processamento (Narendra, 1997). Quanto do passado deve ser guardado na memória e que parte da solução deve ser calculada *on-line* ?

Rede MLP *Multilayer Perceptrons Network*	Rede RBF *Radial Basic Functions*
Mais adequada para controle (Katič and Vukobratovič, 1996)	Mais adequada para controle (Katič and Vukobratovič, 1996)
Rede *feedforward*	Rede *feedforward*
Função de ativação não linear do tipo Sigmoidal	Função de ativação não linear Gaussina (não monotônica)
Leis de aprendizado: *backpropagation*, *Quick Prop* e RPROP.	2 Etapas: 1ª) Algoritmo de *clusterring* para a camada intermediária (a das funções Gaussianas); 2ª) algoritmo LMS (*Least Mean Squared error*) ou método da pseudo-inversa (Gabrijel and Dobnikar, 1997; Keller, 1999)
Capacidade de aprendizado generalizado global. As funções sigmoidais são melhores para extrapolação porque elas seguem a função (que se deseja aproximar) e suas derivadas, fora do espaço de treinamento (Keller, 1999)	Meio termo entre generalização local e global (Katič and Vukobratovič, 1996). Alguns atores afirmam que está rede possui capacidade de generalização local porque as funções de ativação Gaussianas utilizadas fornecem resposta significativa apenas na vizinhança prõxima à seu centro (Orr, 1996).
O algoritmo de aprendizado *back propagation* é do tipo *time consuming*. Acelerações no algoritmo de aprendizado foram sugeridos por Verma (1997), como o algoritmo QuickProp proposto por Fahlman (1988), RPROP (Riedmiller, 1994b) proposto por Riedmiller and Braun (1993). Ver também (Demath and Beale, 1999; Tveter, 1998; Sarle, 2002)	Treinamento mais acelerado.
Outros problemas: determinar-se o número de neurônios da camada intermediária da rede. Normalmente uma camada oculta é suficiente para a maioria dos problemas de controle (Narendra, 1997)	Problemas para determinar a largura das funções Gaussinas utilizadas – parâmetro σ, já que este parâmetro influencia a sensibilidade individual destes neurônios (Keller, 1999). Orr (1998) propõe alternativas para otimizar a largura destas funções. Enquanto Gabrijel and Dobnikar (1997) propõem uma rede RBF adapativa como forma de solucionar parte destes problemas.

Tabela 3.1: Características de redes neurais MLP e RBF.

Em problemas de controle onde a possibilidade de instabilidade existe, este problema é de grande importância. De fato, em muitas situações (como no caso de aeronaves mais ágeis), soluções ótimas devem ser computadas *off-line* e armazenadas *on-board* na aeronave (Narendra, 1997). Métodos eficientes para se obter isto através de redes neurais estão sendo estudados. Problemas de controle ótimo são resolvidos para uma grande variedade de condições iniciais, de forma *off-line* e então utilizadas para treinar controladores por realimentação (Narendra, 1997).

3.2.7 Estabilidade de neurocontroladores

Se o processo a ser controlado é estável, a identificação não costuma ser um problema sério. Mas, se um controlador fixo foi treinado *off-line*, sua implementação no controle em tempo-real de um processo não é direta, uma vez que questões relacionadas à estabilidade devem ser levadas em consideração.

Uma forma de lidar com isto é comparar a saída de controle gerada pelo neurocontrolador com o controlador que existia antes. Comparações entre as saídas dos dois controladores são monitoradas e se o desempenho do neurocontrolador se eleva, o controlador que existia antes pode ser substituído (Narendra, 1997).

Se um bom modelo do sistema a ser controlado foi encontrado, estudos de simulação deveriam ser realizados para garantir-se sua estabilidade assintótica antes de se implementar controladores neurais (Narendra, 1997). Vale a pena notar que este estudo é exigido apenas para o caso de problemas de seguimento de trajetória (*path tracking*) e não para os casos de problema de regulação (erros numa trajetória ponto à ponto) (Narendra, 1997).

3.2.8 Reconhecimento de Padrões em Controle

Em muitos casos, o controle é baseado em reconhecimento de padrões ao invés de estimativas instantâneas de estado. Assumindo-se que um padrão corresponda a um conjunto compacto tanto no espaço de estados quanto no espaço de parâmetros, isto implica em que a mesma ação de controle é tomada para todos os membros deste conjunto (Narendra, 1997). Note que a exatidão do controle é reduzida neste caso. Entretanto, neste mesmo caso, um controle mais grosseiro permite que ações mais rápidas sejam tomadas. São os casos de controle contra colisões (caso típico de aplicações em robótica móvel), resposta rápida para situações de alto risco e planejamento de contingências (Narendra, 1997). A vantagem seria o desenvolvimento de um controlador mais simples. O problema é saber quando um controlador de ajuste mais grosseiro deve ser usado e quando deveria ser trocado por uma estimativa e controladores mais precisos.

3.3 Estruturas Típicas de Controladores Neurais

Os controladores não lineares como o do torque computado, os com a lei de controle baseado no método de Lyapunov, o controle por modos deslizantes (*sliding-mode control*), normalmente usam informação do modelo do robô à ser controlado. A idéia de aplicar um controle inteligente aqui é não ser imprescindível um modelo do sistema à ser controlado.

Muitos dos controladores neurais usam redes do tipo MLP, constituídas por uma camada de entrada, uma invisível e uma de saída.

Arquiteturas típicas de redes neurais para controle (Watanabe, 1996; Katič and Vukobratovič, 1996) são do tipo:

a) de **aprendizado especializado**: Aqui a rede é sintonizada através do erro entre a resposta desejada e a instantânea do sistema

b) de **aprendizado generalizado**;

Normalmente a rede é primeiro treinada *off-line* baseada no erro de controle até que boas propriedades de convergência tenham sido atingidas, e então a rede é colocada para trabalhar *on-line* com um controlador do tipo *feedforward*, onde a rede então continua sua adaptação frente às mudanças no sistema de acordo com procedimentos de aprendizado especializado (Katič and Vukobratovič, 1996). A próxima seção esclarece melhor as diferenças entre uma arquitetura para aprendizado generalizado e outra com aprendizado especializado.

Katič and Vukobratovič (1996) e Morris and Khemaissia (1996) afirmam que as **arquiteturas** de aprendizado **mais apropriadas, promissoras** para o controle de robôs parecem ser:

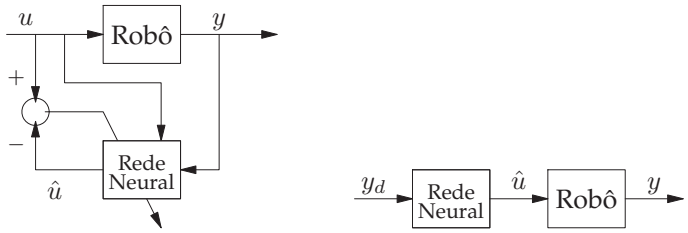
a) **arquitetura de aprendizado baseado na realimentação do erro (*feedback-error learning architecture*)** e;

b) **arquitetura de aprendizado adaptativo (*adaptive learning architecture*)**

Estas arquiteturas também são analisadas a seguir.

3.3.1 Arquitetura de Aprendizado Generalizado

Como mostra a figura 3.6(a), uma vez fornecidas as entradas e saídas de um sistema – os padrões de entrada da rede $\{u(k), y(k)\}$, a rede deve ser treinada *off-line* para que a saída da rede, $\hat{u}(k)$, se aproxime de $u(k)$, usando o algoritmo de *back propagation*.



(a) RN sendo treinada. (b) RN treinada.

Figura 3.6: Arquitetura de aprendizado generalizado para uma RN.

Se a saída da planta, $y(k)$, pode ser aproximada para a referência $y_d(k)$, então se espera que ao final do treinamento da rede, se possa obter da rede treinada, a entrada de controle, $u(k)$, que deve ser aplicada ao sistema para que o sistema gere na sua saída a resposta $y(k)$ - ver 3.6(b). Isto implica em que, uma vez a rede neural esteja bem treinada ($y(k) \cong y_d(k)$), ela se comporte como um **modelo dinâmico inverso** do sistema à ser controlado em malha aberta.

Depois de treinada esta rede pode ser utilizada como um controlador *feedforward*, como mostrado na figura 3.6(b). Ou pode ser útil para projetar a lei de controle de um controlador adaptativo (Er and Liew, 1997; Narendra and Parthasarathy, 1993; Ge et al., 1997). O problema é que esta estratégia implica primeiro numa etapa *off-line* para que a rede possa aprender a aproximar-se do modelo dinâmico do sistema. E ainda, dependendo do esquema de controle projetado depois, este pode não possuir capacidade própria para se adaptar à mudanças no sistema.

3.3.2 Arquitetura de Aprendizado Especializado

Na estrutura mostrada na figura 3.7, a rede usa o sinal de referência, $y_D(k)$ como uma das entradas. A rede é treinada *on-line* para encontrar a entrada $u(k)$ que leve a saída do sistema, $y(k)$, para o valor desejado, $y_d(k)$. Os pesos da rede são ajustados com base no erro entre a saída desejada e o valor da resposta atual do sistema. O algoritmo de treinamento utilizado é normalmente baseado no gradiente descendente (*steepest descent*).

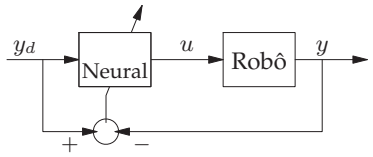


Figura 3.7: Arquitetura de aprendizado especializado para uma RN.

3.3.3 Controlador Neural *Feedforward*

A figura 3.8 mostra um controlador neural de malha direta (*feedforward*), que utiliza a rede neural treinada *off-line* mostrada na figura 3.6(a).

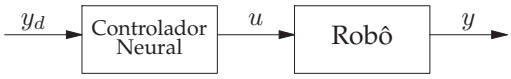


Figura 3.8: Controlador neural *feedforward*.

3.3.4 Controlador baseado no aprendizado do erro de realimentação

A figura 3.9 mostra um controlador com arquitetura baseado no aprendizado *on-line* do erro de realimentação entre o valor de saída de desejado e o valor de saída atual do sistema, estrutura mais conhecida como ***Feedback-error-learning control system*** (Watanabe, 1996).

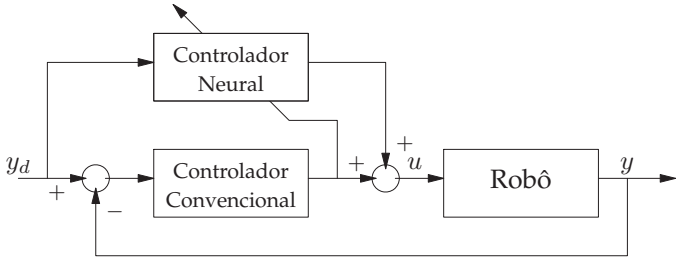


Figura 3.9: Controlador baseado no aprendizado do erro de realimentação.

Nesta abordagem, durante o treinamento *on-line* a saída do controlador de malha direta predomina durante a primeira metade do treinamento da rede neural. Gradualmente a rede neural assume sobre o controlador de malha direta quanto o treinamento completa (Watanabe, 1996).

Esta arquitetura possui uma característica especial: ela é rapidamente aplicável à um sistema instável, porque se supõe que o sistema já foi estabilizado pelo controlador baseado na realimentação (Watanabe, 1996).

3.3.5 Controlador Auto-Ajustável baseado numa Rede Neural

Uma rede neural ainda pode ser utilizada como um mecanismo adaptativo para um controlador baseado em realimentação, isto é, como um controlador auto-ajustável da figura 3.10 (Watanabe, 1996). O aprendizado da rede nesta abordagem continua sendo especializado.

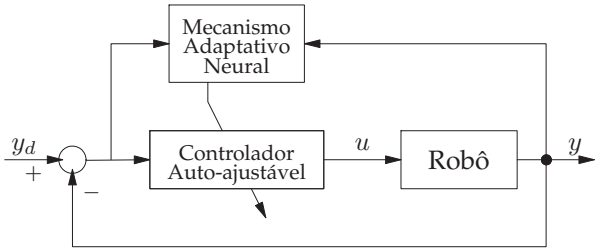


Figura 3.10: Controlador auto-ajustável baseado em rede neural.

3.3.6 Rede Neural para Identificar Sistemas

A estrutura mostrada na figura 3.11 pode ser utilizada para identificar o modelo dinâmico direto de um sistema (Warwick, 1996; Watanabe, 1996) e pode, depois de treinada, ser utilizada para testar/treinar outros tipos de controladores para este sistema. (Watanabe, 1996). O objetivo desta rede é se comportar da mesma forma que os sistema. Neste esquema, a rede age como um aproximador de funções não lineares (Warwick, 1996).

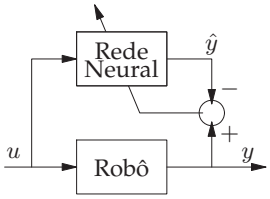


Figura 3.11: Rede neural para identificação do modelo dinâmico direto de um sistema.

Redes RBF exibem melhores propriedades de aproximação em comparação com redes MLP. As redes RBF poder sempre ser construídas com uma única camada invisível e seus pesos ajustados usando uma técnica linear que garante uma solução global (Warwick, 1996). No entanto, o problema em usar redes RBF consiste em se determinar adequadamente os centros das suas funções gaussianas. O que pode ser problemático e na prática, inviabilizar uma solução à menos que o número de funções e seus centros tenham sido afortunadamente determinados (Warwick, 1996).

Um **problema** a ser considerado na modelagem de um sistema: a maioria dos sistemas possui uma certa dinâmica; então é desejável que a ferramenta que será utilizada para modelar este sistema também incorpore dentro si uma dinâmica (Warwick, 1996). Uma solução seria desenvolver rede recorrentes que envolvem realimentações internas. Outra possibilidade seria introduzir um comportamento dinâmico dentro dos próprios neurônios da rede. Outra solução mais usual é alimentar as entradas de uma rede neural com sinais de entrada e saída atrasados (que foram previamente amostrados) (Warwick, 1996).

Modelagem ARMA Não Linear

Sob certas condições uma função não linear qualquer pode ser modelada por séries de Wiener ou Volterra. Considere primeiramente a representação da média móvel (**MA** = *Moving Average*):

$$y(k+1) = \sum_{i=0}^{\infty} a_i u(k-i) \tag{3.2}$$

onde a_i corresponde à valores escalares associado às caraterísticas de resposta de um sistema frente à uma entrada degrau aplicada no tempo $k = 0$ (sistemas discretos no tempo). Tanto as séries de Wiener quanto de Volterra generalizam uma forma de representação linear da equação (3.2) num formato não linear. O resultado é um sistema dinâmico invariante no tempo caracterizado por uma transformação não linear. As séries de Volterrra, mostradas em (3.3) incluem uma combinação linear de valores de entrada de amostragens anteriores ponderados com respeito ao tempo:

$$y(k+1) = a_0 + \sum_{i=0}^{\infty} a_i u(k-i) + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} a_{ij} u(k-i)u(k-j) + \dots \tag{3.3}$$

onde a_0, a_i, a_{ij} são os coeficientes desconhecidos.

Uma rede neural inclui ao menos uma camada invisível que pode arbitrariamente aproximar qualquer mapeamento não linear, assim, a rede neural mostrada na figura 3.12 pode representar qualquer função não linear que também possa ser modelada pela série de Volterra, isto, supondo-se que todos os valores de entradas do passado estão disponíveis como dados de entrada para a rede neural (Warwick, 1996).

Note que uma rede não pode possuir um número infinito de entradas passadas, assim, um número finito de entradas é utilizado conforme o nível de exatidão que se queira. Por

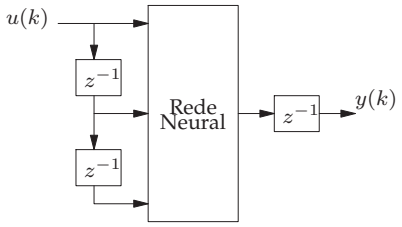


Figura 3.12: Rede Neural para modelagem por MA.

outro lado, se n sinais atrasados são aplicados numa rede, então se está assumindo que sinais de entrada aplicados antes dos n períodos de amostragem causam um efeito negligenciável no sinal de saída (Warwick, 1996).

Modelos Auto Regressivos (**AR**) também são em alguns casos, uma forma útil de representar a dinâmica de um sistema. Neste caso, a saída do sistema está linearmente relacionada com saídas atrasadas do sistema. Se restringirmos o modelo aos n primeiros sinais, a saída é dada por:

$$y(k+1) = \sum_{i=0}^{n-1} a_i y(k-1)$$

Esta forma é interessante para uma descrição por séries temporais envolvendo um número limitado de parâmetros, na qual, nenhum sinal de entrada é diretamente utilizado (Warwick, 1996).

A rede neural mostrada na figura 3.13 pode ser utilizada para obter uma função de saída. A rede neural é empregada para aproximar qualquer efeito não-linear. Os valores a_i da modelagem corresponderiam aos pesos sinápticos da rede aplicados à cada uma das entradas atrasadas da rede. Um sistema pode ser representado na forma AR se: (a) $\hat{y}(k+1)$ representa $y(k+1)$ através de uma função não-linear, (b) o sistema pode ser adequadamente modelado por um modelo AR de n-ésima ordem e (c) não existe ruído corrompendo o sistema ou modelo (Warwick, 1996).

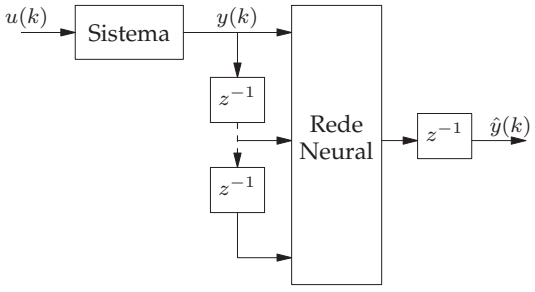


Figura 3.13: Rede Neural para modelagem AR.

Finalmente, pela combinação dos modelos **AR** e **MA**, qualquer sistema dinâmico linear de ordem n, discretizado no tempo, pode ser representado pela equação de diferenças:

$$y(k+1) = \sum_{i=0}^{n-1} a_i y(k-i) + \sum_{i=0}^{n-1} b_i u(k-i) \tag{3.4}$$

que é usualmente conhecido como modelo de média móvel auto-regressiva (**ARMA**=*AutoRegressive Moving Average*). Barto (1989) também faz seus comentários à respeito do número de termos necessários para realizar uma melhor aproximação de um sistema relacionado com a ordem estimada para o sistema.

Passando o resultado da equação 3.4 através de uma função não-linear e assumindo que a saída do sistema (real) é corrompida por ruído, têm-se o que se define por modelo não-linear de média móvel auto-regressiva de um sistema, ou modelo **NARMAX** e esta forma de modelagem está ao alcance de uma rede neural.

Com relação à identificação de sistemas, as redes neurais ainda trazem como vantagem o fato de possuírem boas propriedades de aproximação, são simples e fáceis de serem implementadas, ainda com processamento (interno) paralelo e características de tolerancia à falhas. Entretanto o uso de uma rede neural para identificação de um sistema qualquer deve ser melhor considerado. Por exemplo, se um sistema de aquecimento de primeira ordem for identificado através de uma rede neural MLP, a simplicidade do modelo original (pelas técnicas tradicionais), se perde nas camadas e conexões internas da rede.

Controle a Modelo Interno (IMC)

A identificação de um sistema pode primeiramente ser feito como mostrado na figura 3.11, onde um sinal de erro ($\tilde{y} = y - \hat{y}$) pode ser gerado para detectar os erros de modelagem. Nesta abordagem os pesos da rede são ajustados de forma a minimizar os erros de modelagem.

As redes neurais também podem ser organizadas de várias formas dentro de um esquema de controle com realimentação. Uma opção mais direcionada para análises de estabilidade e robustez trata-se do Controlador por Modelo Interno (**IMC**=*Internal Model Control*). Nesta abordagem, um modelo de sistema é operado em paralelo com o sistema sendo controlado, como mostra a figura 3.14.

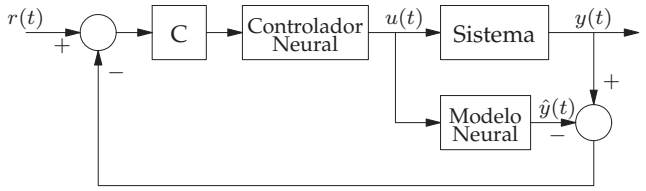


Figura 3.14: Controlador a modelo interno (IMC).

O erro entre as saídas do modelo e do sistema é utilizado para a realimentação. Neste caso, o modelo baseado numa rede neural pode ser obtido *off-line* através de alguma abordagem de identificação neural de um sistema ou através de um modelo ideal de especificação de desempenho para o sistema (este modelo poderia ser por exemplo, a forma esperada de resposta para um certo sistema).

O sinal de realimentação obtido numa abordagem IMC é passado através de um esquema de controle, que está diretamente relacionado com o modelo inverso do sistema, junto com um outro subsistema "C" (como mostrado na figura 3.14) que é incluído para garantir a robustez e propriedades de seguimento da referência.

Se este esquema de controle for realizado através de redes neurais, duas redes seriam utilizadas: uma para modelar as características da malha direta do sistema (malha aberta) e a outra para modelar as características inversas do sistema. Entretanto, principalmente por causa de um modelo inverso estar localizado dentro da malha de realimentação, uma implementação direta só é possível para sistemas estáveis em malha aberta (Warwick, 1996).

Outros métodos de controle como controle ótimo ou preditivo (**GPC** (Clarke et al., 1987a; Clarke et al., 1987b)) podem ser implementados através de uma rede neural. O IMC neural é apenas um exemplo de implementação por redes neurais (Warwick, 1996). Freund and Pesara (1998) aborda (rapidamente) a aplicação de preditor de Smith mais adequado para sistemas com longo tempo morto no controle de força de robôs, numa busca por um controlador com propriedades especiais com boa rejeição à perturbações relacionadas à fricção e movimento suave.

3.3.7 Controladores Adaptativos e Redes Neurais

Os controladores adaptativos baseados em redes neurais se mostram interessantes por não requererem conhecimento *a priori* do sistema à ser controlado. É uma abordagem diferente de outros controladores neurais que ainda exigem o levantamento do modelo

dinâmico inverso do sistema e que acabam consumindo muito tempo de processamento numa etapa de treinamento *off-line*. E é ainda uma abordagem *on-line*. A rede pode ser simplesmente inicializada com zeros, assumindo-se nenhum conhecimento a respeito do sistema que se deseja controlar.

Um controlador adaptativo é capaz de monitorar e modificar seu próprio desempenho, modificando sua ação de controle de forma a garantir um desempenho satisfatório. É indicado para os casos em que um certo nível de incerteza existe em relação ao processo à ser controlado mas mesmo assim se deseja manter um desempenho satisfatório (Narendra, 1997). Isto implica em que, tanto a identificação da planta quanto seu controle, tenham que ser atualizados *on-line*.

A dinâmica de um sistema discreto no tempo pode ser representada através de um modelo normalizado, determinístico de média móvel auto-regressiva (modelo **DARMA**) através da equação:

$$y(k+1) + a_0 y(k) + \cdots + a_{n-1} y(k-n+1) = b_0 u(k) + b_1 u(k-1) + \cdots + b_l u(k-l)$$

Esta equação pode ser expressa numa forma paramétrica conhecida como **modelo regressivo**:

$$y(k+1) = \Phi^T(k)\Theta_0$$

onde:

$$\Phi^T(k) = [y(k), y(k-1), \ldots, y(k-n+1), u(k), \ldots, u(k-l)]^T$$

corresponde ao modelo regressor e:

$$\Theta_0 = [-a_0, -a1, \ldots, -a_{n-1}, b_0, \ldots, b_l]^T$$

trata-se do vetor de parâmetros. A dinâmica do sistema descrito desta forma pode ser referenciado na forma de um **preditor** (Morris and Khemaissia, 1996).

Note que o vetor de parâmetros Θ_0 é estimado em tempo real usando dados de entrada/saída do sistema. Este vetor pode ser determinado de modo determinista usando o algoritmo preditivo e recursivo de erros (RPE = *Recursive Predition Error*) ou algoritmo recursivo dos menores erros quadráticos (RLS = *Recursive Least Squares* ou pelo filtro estendido de

Kalman (EKF = *Extended Kalman Filter*) (Morris and Khemaissia, 1996). Cada um destes algoritmos possui as suas deficiências dependendo da aplicação em questão e assim um novo e melhor método pode ser levado em consideração, o algoritmo baseado na ponderação paralela dos menores erros quadráticos (PWRLS=*Paraller Weighted Recursive Least Squares*). Este algoritmo pode ser implementado de diversas formas. Uma delas através de um rede neural conforme mostrado por Morris and Khemaissia (1996). Morris and Khemaissia (1996) chega a mostrar o caso de um neuro controlador baseado no torque computado. Nesta estrutura, a rede neural é utilizada para modelar a dinâmica inversa de cada uma das juntas de forma a realizar uma compensação não-linear para o manipulador – interessante notar que a rede neural é aplicada numa estrutura de aprendizado baseado na realimentação do erro, como já demonstrado no item 3.3.4.

Baseado nisto, alguns controladores adaptativos usando redes neurais têm sido propostos (Sun et al., 1998; Zhihong et al., 1998; Noriega and Wang, 1998; Ge et al., 1997; Song, 1997; Jagannathan, 1996; Carelli et al., 1995). Várias estratégias adaptativas foram incorporadas para ajustar *on-line* os parâmetros tanto do identificador quanto do controlador. **A precisão é garantida pela parcela não linear e variante no tempo do controlador enquanto a robustez é garantida pela parcela fixa deste controlador** (Narendra, 1997).

Narendra (1997) considera que uma combinação entre um controlador ótimo de estrutura fixa, treinado *off-line* e um controlador adaptativo treinado *on-line* parece oferecer o melhor compromisso entre robustez e desempenho em muitas aplicações industriais. Mas esclarece que há ainda problemas teóricos e práticos a serem resolvidos antes que procedimentos sistemáticos destes controladores sejam desenvolvidos. (Narendra, 1997) também chama a atenção para a investigação de controladores baseados em múltiplos modelos, chaveamento (*switching*) entre controladores (diferentes estratégias) e sintonização. Mas no nosso caso de controle de um robô manipulador, restrições de poder e tempo de processamento podem limitar a busca pela solução de um controlador deste tipo.

3.3.8 Resumo de controladores baseados em redes neurais

O uso de uma arquitetura de rede neural capaz de aprender o modelo dinâmico inverso de um sistema pode ser útil para o projeto da lei de controle para um controlador adaptativo direto (Er and Liew, 1997). Essencialmente redes neurais podem ser úteis para estimar

o modelo inverso do sistema (Er and Liew, 1997). Isto seria realizado em duas etapas. Primeiramente uma rede neural seria utilizada para estimar o modelo dinâmico e então outra rede é introduzida para estimar a dinâmica inversa do sistema (Er and Liew, 1997).

Também uma rede neural pode ser utilizada num laço de malha direta (*feedforward*) em paralelo com um controlador convencional por realimentação (Er and Liew, 1997; Watanabe, 1996). Nesta arquitetura (ver figura 3.9) a medida que a rede aprende, o compensador do sistema tende a ser *feedforward* com pouca compensação relativa à parte da realimentação (Er and Liew, 1997).

Como as abordagens anteriormente revisadas empregam funções não lineares como as sigmoidais nos seus neurônios e empregam ao menos uma camada intermediária (oculta), facilmente se desenvolve uma função de mapeamento não linear dentro da rede neural empregada (Watanabe, 1996).

Pode-se afirmar ainda que estas abordagens possuem uma capacidade de generalização, respondendo corretamente mesmo para entradas para as quais não foram especificadas, quando suas redes neurais empregarem algum algoritmo de aprendizado com capacidade de generalização (caso das redes MLP com algoritmo de aprendizado *back propagation*). Mas o mesmo não pode ser dito para redes sem capacidade de generalização garantida como parece ser o caso das redes RBF.

Perceba-se entretanto que é fácil de acontecer de uma destas redes neurais empregadas numa das abordagens anteriormente revisadas, atinja um mínimo local (e não um mínimo global que garantiria o menor erro de treinamento), principalmente no caso de se utilizar o algoritmo de aprendizado *back propagation* ou outro baseado no gradiente do erro.

Por outro lado, o tamanho da rede pode aumentar bastante no caso de um sistema de múltiplas entradas e múltiplas saídas (MIMO), como no caso do controle de robôs manipuladores. Em particular o número de unidades na camada intermediária pode aumentar bastante, chegando a inviabilizar um processo de treinamento *on-line* (Watanabe, 1996).

Watanabe mostra em artigos seus de 1994 e 1995 que pode ser empregado com sucesso um método para ajustar o número de neurônios funcionais das camadas intermediárias e talvez ainda ajustar o tipo de função de ativação não linear adotada nos neurônios da camada intermediária (utilizou funções sigmoidais bipolares e unipolares) além de simplesmente ajustar os pesos sinápticos dos neurônios da rede como os métodos de aprendizado con-

vencionalmente fazem, com o objetivo final de diminuir o número de neurônios da camada intermediária e propiciar um treinamento mais rápido para a rede (Watanabe, 1996).

Redes neurais podem ainda ser empregadas em decisões de controle ou em controle supervisório, num nível hierárquico mais elevado, ou ainda num sistema de inferência ou detecção de falhas – nestes últimos casos, a operabilidade, robustez ou convergência da rede é menos importante que dentro de uma estrutura em malha fechada de controle. (Warwick, 1996).

3.4 Aplicações de RNs em Robótica de Manipulador

Na área de robótica, as redes neurais têm sido empregadas como elementos de aprendizado e compensação em laços de controle (Katič and Vukobratovič, 1996). As redes neurais fazem uso de sua não linearidade, capacidade de aprendizado e capacidade de processamento paralelo e generalização para aplicações em robótica avançada. O aprendizado aqui está relacionado com adaptação, a adaptação de parâmetros embutida na interconexões entre os neurônios. Normalmente, o aprendizado e controle são realizados simultaneamente e o aprendizado continua enquanto perturbações estiverem presentes no robô sob controle (Katič and Vukobratovič, 1996).

Quatro modelos de redes neurais estão sendo mais empregadas na área de controle de robôs (Katič and Vukobratovič, 1996). Estes modelos de redes neurais podem ser distinguidas de acordo com o tipo de conexões entre a entrada e saída da rede, podendo ser uma rede do tipo de conexões diretas da entrada para a saída (*feedforward*) ou uma rede com realimentação (*feedback*). Em robótica, as redes "diretas"(*feedforward*) normalmente empregadas são as MLP (*Multilayer Perceptrons*) e RBF (*Radial Basis Functions networks*) e CMAC (*Cerenellar Model Arithmetic Computer network*). As redes com realimentação (*feedback*), Hopfield e versões recorrentes das redes *perceptron* multicamadas vêm sendo aplicadas no controle conexionista de robôs (Katič and Vukobratovič, 1996). Entre elas as redes mais adequadas para propósitos de controle são as redes MLP (*Multilayer Perceptrons*) e RBF (*Radial Basis Functions networks*).

3.4.1 Modelagem da Cinemática Inversa

Se refere ao modelo neural da cinemática inversa de um robô, ou seja, dadas as posições, velocidades e acelerações que se deseja que o robô desenvolva, no espaço cartesiano, se faz necessário determinar estas mesmas informações, mas no espaço de juntas, isto é, determinar as posições, velocidades e acelerações de cada uma das juntas deste robô. Note que a solução para um caso pode não ser única (Katič and Vukobratovič, 1996) – e neste ponto iniciam os problemas.

O cálculo da cinemática inversa para posição é dado por:

$$q = K^{-1}(x)$$

onde q se refere as posições angulares das juntas [rad]; x se refere a posição e orientação do efetuador final do robô, no espaço cartesiano – respectivamente [m] e [rad] – normalmente um vetor 6×1 do tipo: $x = [\ x \quad y \quad z \quad \varphi \quad \delta \quad \psi\ ]^T$; K simboliza a função que relaciona estes dois espaços, normalmente é não-linear por estar composta por termos de $senos$ e $cosenos$.

Para a determinação da velocidade angular se usa a relação:

$$\dot{q} = J_A^{-1}(q)\dot{x} \tag{3.5}$$

onde $\dot{q}$ se refere às velocidades angulares de cada junta do robô [rad/s]; $\dot{x}$ se refere a velocidade linear e angular do efetuador final do robô – respectivamente [m/s] e [rad/s]; e J_A se refere à transformação que define de que modo a velocidade angular de cada junta contribui na velocidade linear e angular do efetuador final.

E finalmente a aceleração que deveria ser desenvolvida por cada uma das juntas, $\ddot{q}$, é dado por:

$$\ddot{q} = J_A^{-1}(q)\Big[\ddot{x} - \dot{J}_A(q)\dot{q}\Big]$$

O Jacobiano (analítico) é definido como:

$$J_A(q) = \frac{\partial K(q)}{\partial q} \tag{3.6}$$

Note que aqui nos deparamos com um problema "comum" na área de robôs manipuladores – o problema da determinação da inversa deste Jacobiano. Note que a solução da

equação 3.5 só pode ser calculada se o Jacobiano for *full rank*, isto é, de posto cheio. Este problema ocorre em configurações de singularidade, isto é, situações em que: (a) se comanda o deslocamento do robô para um posição fora de seu espaço operacional (impossível que ele alcance esta posição – denuncia problemas no planejamento da tarefa); (b) quando a matriz da inversa do Jacobiano na equação (3.5) é singular, ou seja, algum de seus termos é nulo, o que significa que infinitas soluções para o problema da cinemática inversa são possíveis; e (c) quando o robô está próximo de uma região de singularidade, quando então, pequenas velocidades no espaço cartesiano podem desencadear grandes velocidades no espaço de juntas (Sciavicco and Siciliano, 1996). Singularidades também podem acontecer dentro do espaço operacional do robô causado pelo alinhamento de dois ou mais sistemas de coordenadas relacionadas às juntas do robô – este caso de singularidade é o mais sério pois pode ocorrer em qualquer configuração praticável pelo manipulador dentro da sua área de operação, e normalmente ocorre no planejamento da trajetória no espaço cartesiano, caso do controle de força, em que a tarefa é geralmente especificada no espaço cartesiano (Sciavicco and Siciliano, 1996).

Zalzala (1996) propõe uma rede neural para resolver este problema, no caso específico dele, para um robô manipulador do tipo PUMA 560, de 6 graus de liberdade. A rede proposta por Zalzala possui 6 neurônios na sua entrada (correspondendo a variação do vetor x) e 6 neurônios na sua camada de saída (correspondendo à variação de cada uma das 6 juntas deste robô). O detalhe é que esta sua rede neural é composta de 6 camadas ocultas, onde cada uma desempenha a seguinte função:

- camada oculta 1: calcula os elementos exigidos para a terceira coluna de J_{11} e a primeira coluna de J_{22} – possui 7 neurônios.
- camada oculta 2: calcula os elementos para as primeira e segunda colunas de J_{11} e as primeira e terceira colunas de J_{21} – contem 13 neurônios.
- camada oculta 3: calcula a matriz completa J e os elementos de J_{22}^{-1} – contêm 22 neurônios.
- camada oculta 4: calcula J_{11}^{-1} – possui 9 neurônios.
- camada oculta 5: calcula $-J_{22}^{-1}$, J_{21} e J_{11}^{-1} – composta de 9 neurônios também.

• camada oculta 6: calcula as taxas de variação desejadas, multiplicando o resultado da inversa do Jacobiano com as taxas de variação fornecidas no plano cartesiano – composta de 6 neurônios.

A arquitetura desta rede é baseada na formulação matemática para se determinar:

$$J^{-1} = \begin{bmatrix} & J_{11}^{-1} & & 0 \\ -J_{22}^{-1} & J_{21} & J_{22}^{-1} & J_{22}^{-1} \end{bmatrix}$$

Zalzala trabalhou com rede neural do tipo MLP e algoritmo de treinamento *back propagation* com termo momento, otimizado para lidar com conexões esparsas já que nem todos os neurônios de uma camada se ligam diretamente aos neurônios da camada imediatamente posterior. Usou função de ativação sigmoidal variando entre ± 0.5 e zerou todos os valores de *bias*. Foram executados 900 ciclos de controle durante a simulação e ainda para testar a habilidade adaptativa desta abordagem, alterou em 25% o parâmetro de comprimento da terceira junta deste robô, antes que o robô completasse todo o movimento especificado. Verificou erros de seguimento na trajetória variando de 0.052 à 0.511 graus. Todos os erros nas juntas se reduziram significativamento, exceto para as juntas 4 e 6, onde foi percebido um problema de mínimo local.

3.4.2 Controlador neural de posição baseado em realimentação do erro para um robô SCARA

Er and Liew (1997) apresentaram uma estratégia de controle de posição para um robô de n degraus de liberdade, baseado numa evolução para o controle usando rede neural com arquitetura baseada na realimentação do erro (EFELC=*Enhanced Feedback Error Learning Control*). Testaram sua proposta sobre um robô industrial do tipo *Adept One*. Este robô utiliza sistemas *direct-drive* para atuar sobre os motores das suas juntas. Não utiliza engrenagens ou outro mecanismo de conversão de torque. Ao invés disto utiliza motores DC sem escovas de alto torque e baixas velocidades. Este esquema limita as fricções ocasionadas por engrenagens e folgas nas juntas, permitindo operação limpa, precisa e em alta velocidade.

O robô *Adept One* foi simulado como um manipulador de 3 graus de liberdade onde a dinâmica do quarto elo foi negligenciada mas sua massa e a massa correspondente à carga no efetuador final foi incorporada à massa do terceiro elo. E por questões ainda

de simplicidade, realizaram simulações apenas para as duas primeiras juntas. Compararam sua proposta com a estratégia existente de controle baseado em realimentação do erro (FELC=*Feedback Error Learning Control*) frente à seguimento da trajetória, variações na trajetória, diferentes valores iniciais de carregamento.

A estrutura convencional FELC é mostrada na figura 3.15.

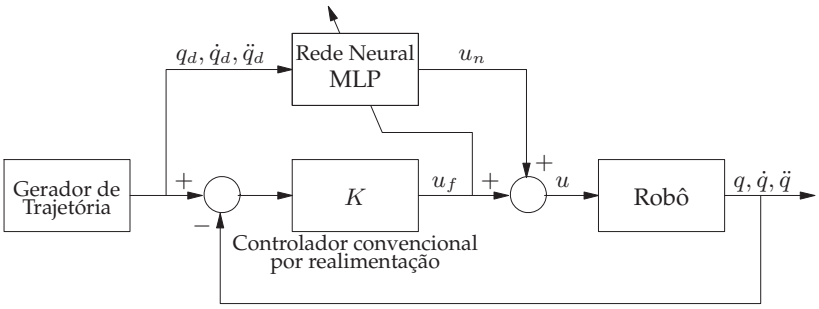


Figura 3.15: Arquitetura convencional baseada na realimentação do erro (FELC).

Trata-se de uma rede multicamadas de *perceptrons* não lineares (MLP). Note que a estrutura original do controlador convencional que trabalha em paralelo com a rede, não é alterada. Normalmente a rede utiliza como sinais de entrada, as posições, velocidades e acelerações angulares desejadas para cada uma das juntas (respectivamente: os vetores $q_d, \dot{q}_d, \ddot{q}_d$). O modelo da dinâmica inversa recebe a trajetória desejada, q_d e monitora o torque da realimentação, u_f, como seu sinal de erro. Se espera que o sinal gerado pela realimentação (u_f) tenda para zero à medida que a rede aprende. A *priori* conhecimentos sobre a estrutura dinâmica do manipulador não são necessários. A rede percebe a trajetória desejada para o robô através do torque gerado pelo sinal da realimentação, u_f. A rede utiliza este último sinal para calcular seu sinal de erro:

$$\tau \frac{d\left(w_{km}^{HO}\right)}{d(t)} = y_k^H \dot{\mathcal{F}}(x_m^O) u_{fm}$$

onde τ representa a constante de tempo relativa às modificações sinápticas; w_{km}^{HO} representa a matriz de pesos sináptica entre o k-ésimo neurônio da camada oculta (*Hidden*) e o m-ésimo neurônio da camada de saída (*Output*); y_k^H representa a saída do k-ésimo neurônio da camada oculta; $\mathcal{F}(\cdot)$ representa a função sigmoidal; x_m^O representa o somatório ponderado de entrada de dados ao m-ésimo neurônio da camada de saída; e u_{fm} representa o sinal gerado pela realimentação relativa ao comando de torque à junta m.

O sinal de erro entre o k-ésimo neurônio da camada oculta oculta é calculado como o somatório ponderado dos sinais de torque enviados pela realimentação ao robô, u_f, com base nos pesos sinápticos:

$$\delta_k^H = \sum_m w_{km}^{HO} \dot{\mathcal{F}}(x_m^O) u_{fm}$$

A taxa de variação dos pesos sinápticos entre a camada de entrada e a de saída é dada pelo produto entre as entradas dos neurônios da camada oculta e o sinal de erro:

$$\tau \frac{d\left(w_{jk}^{IH}\right)}{d(t)} = y_j^I \dot{\mathcal{F}}(x_k^H) \delta_k^H$$

onde y_j^I corresponde à saída do j-ésimo neurônio da camada de entrada (*Input*); x_k^H corresponde ao resultado do somatório ponderado chegando ao k-ésimo neurônio da camada oculta.

Os neurônios da camada de saída da rede, representam os comandos de torque *feedforward* para os diferentes atuadores em cada uma das juntas.

O torque enviado à cada uma das juntas do robô é dada pela soma entre o comando de torque gerado pela realimentação (depois de passar pelo controlador convencional) e o torque *feedforward* gerado pela rede neural, ou seja: $u = u_f + u_n$.

Neste esquema de controle, o sinal de controle gerado pelo controlador convencional (relacionado à realimentação) é utilizado como o sinal de erro para a rede neural, como também um dos sinais de comando de torque à ser enviado para o motor. Durante a fase de aprendizado, este sinal de controle é adicionado à saída do sinal gerado pela rede neural. A saída resultante parte de valores nulos e vai crescendo à medida que os pesos da rede são ajustados de forma a se aproximar da planta. Como o aprendizado continua, a parte do sinal de controle vinda da rede neural, aumenta e se sobressai, acabando por controlar o robô. Er and Liew (1997) se arriscam a afirmar que o controlador convencional poderia eventualmente ser eliminado como parte da ação de controle.

Er and Liew (1997) **ressaltam** três **problemas na estrutura tradicional do controlador baseado no aprendizado da realimentação do erro (FELC)**:

1) Para trajetórias com **variações lentas de posição**, o **seguimento da trajetória** usando a estrutura FELC é **quase impossível**;

2) Se for utilizada uma **trajetória diferente da utilizada no treinamento da rede neural** (ou seja, ocorre uma mudança na trajetória), o **erro** entre a posição desejada e a instantânea **aumenta** e assim, **mais aprendizado** se faz necessário. Isto significa que a **rede não aprendeu o modelo da dinâmica inversa** do sistema;

3) Perceberam que o **treinamento da rede neural é muito sensível de acordo com a seleção dos pesos sinápticos iniciais**. Notaram que o treinamento da rede neural só foi possível com uma escolha muito limitada dos pesos sinápticos iniciais.

Para solucionar estes problemas tiveram a **idéia** de fornecer diferentes sinais de entrada para a rede neural, **utilizando também os erros de posição ($\tilde{q} = q_d - q$), posição e velocidades angulares instantâneas das juntas (respectivamente: $q, \dot{q}$) como entradas da rede neural** – conforme mostra a figura 3.16.

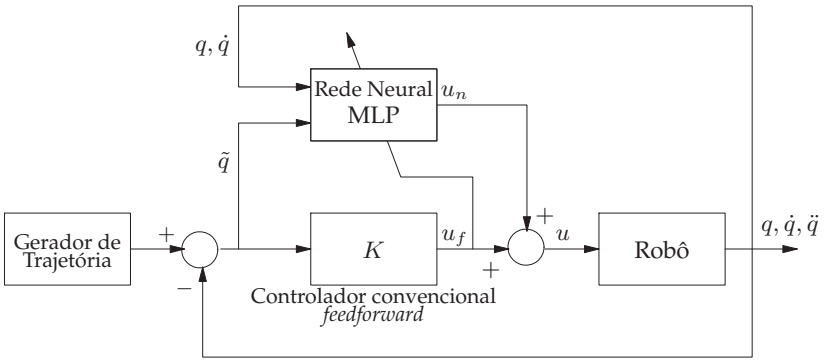


Figura 3.16: Arquitetura avançada sendo proposta – EFELC.

A rede MLP de Er and Liew (1997) trabalha com 7 neurônios na camada de entrada ($q_1, \dot{q}_1, q_2, \dot{q}_2, \tilde{q}_1, \tilde{q}_2$) e 7 na camada oculta e apenas 2 neurônios na camada de saída, correspondendo aos comandos de torque para as duas juntas do robô simulado (u_{n_1}, u_{n_2}). O comando de torque enviados aos motores deste robô é dado por:

$$\begin{aligned} u_j &= u_{f_j} + u_{n_j} \qquad\qquad j = 1, 2, \ldots, n \\ u_{f_j} &= K_{p_j}(q_{d_j} - q_j) \end{aligned}$$

onde K_{p_j} é o ganho proporcional de realimentação para a j-ésima junta do manipulador.

Simulações comprovaram a superioridade da proposta do EFELC sobre a estrutura tradicional FELC. Especificaram erro máximo de posição como sendo menor que 0.05 [rad] (2.87^o). A estrutura convencional FELC não conseguiu atingir estes requisitos enquanto a

estrutura melhorada EFELC atingiu a posição final com erro máximo de 0.016 [rad] (0.92^o). **Diferente do esquema convencional FELC, a estrutura melhorada EFELC não se mostrou tão sensível aos valores iniciais dos pesos sinápticos**. E em simulações para verificar o erro de seguimento da trajetória, um erro máximo de 0.04 [rad] (2.3^o) foi encontrado. Nos casos de seguimento de uma trajetória a estrutura convencional FELC, os erros inicialmente convergiam mas parece que um **mínimo local** era encontrado, fazendo o sistema oscilar nos próximos pontos. Este é um **problema típico inerente ao algoritmo de treinamento baseado no método do gradiente descendente**. Justificam desta forma as falhas da estrutura tradicional FELC. A estrutura falha em trajetórias com lentas variações de posição (variações mais rápidas, acaba excitando a rede, eventualmente fazendo a mesma escapar de mínimos locais). Na estrutura EFELC o problema do mínimo local, foi superado, adicionando-se pequenos valores aleatórios (ruído) à rede, numa técnica similar à de *simulating annealing* que pode ser facilmente incorporada ao tradicional algoritmo *back propagation*.

3.4.3 Experiência de um Controlador (de Posição) Neural Adaptativo, no Espaço Cartesiano

Ge et al. (1997) consideram uso de controlador neural adaptativo para um robô manipulador, no espaço da tarefa. O controlador é baseado numa técnica de modelagem de redes neurais que não exige a determinação do modelo da dinâmica inversa do processo e nem processos que consumem muito tempo com o treinamento da rede. Usam redes RBF, com mecanismo adaptativo uniformemente estável obtendo assim um seguimento assintótico. O controlado obtido não exige a inversão da matriz Jacobiana. Afirmam que um controle robusto pode ser incorporado de forma a suprimir os erros de modelagem de rede neural e as perturbações confinadas dentro de certos limites. Realizaram apenas simulações numéricas.

Diferente de outras abordagens neurais adaptativas, esta foi estendida diretamente para operação no espaço da tarefa (às vezes também chamado de espaço Cartesiano), permitindo-se assim que este robô seja aplicado à certos tipos de tarefa onde além do controle da posição do efetuador final, se deseja controlar a força exercida sobre um objeto ou sobre o meio. E isto é mais fácil de ser obtido se a lei de controle for projetada já no espaço da tarefa. Ocorre que algumas destes técnicas de controle exigem a **inversa da matriz Jacobiana**, que pode ser um **procedimento consumidor de tempo de processamento** e difícil de obter ou problemática

para configurações do robô próximas de seus pontos de singularidade cinemática. A **idéia** aqui seria se **parametrizar a lei de controle já diretamente no espaço da tarefa, eliminando-se a necessidade da inversa da matriz Jacobiana** - seria uma **vantagem** deste método de controle.

Asseguram que se redes neurais com funções Gaussianas de base radial forem utilizadas, um adaptação uniformemente estável e um seguimento assintótico é obtido.

Ge et al. (1997) utilizaram redes RBF numa arquitetura que utiliza l funções gaussianas do tipo:

$$a_i(y) = exp\Big(- \frac{(y-\mu_i)^T(y-\mu_i)}{\sigma^2}\Big)$$

onde $\mu_i \in \Re^n$ é o centro do vetor; $\sigma^2 \in \Re$ se refere à variância da função Gaussiana (sua abertura); $i = 1, \ldots, l$ corresponde a i-ésima função Gaussiana.

A rede neural é empregada para modelar o manipulador. A posição e orientação do efetuador final no espaço de tarefa é dado por $x \in \Re^n$. A dinâmica no espaço da tarefa pode ser descrita como:

$$B_x(q)\ddot{x} + C_x(q,\dot{q})\dot{x} + G_x(q) = H_x$$

$$\begin{aligned} B_x(q) &= J^{-T}(q)B(q)J^{-1}(q) \\ C_x(q,\dot{(}q) &= J^{-T}(q)\big(C(q,\dot{q}) - B(q)J^{-1}(q)\dot{J}(q)\big)J^{-1}(q) \\ G_x(q) &= J^{-T}(q)G(q) \\ H_x &= J^{-T}(q)\tau \end{aligned}$$

onde $J(q) \in \Re^{n\times n}$ é a matriz Jacobiana (analítica)[2], que depende da configuração do manipulador, que se assume não singular no espaço de trabalho limitado, Ω. As equações dinâmicas acima seguem as seguintes propriedades estruturais que serão exploradas no projeto no controlador descrito mais a frente.

Propriedade 1: A matriz de inércia, $B_x(q)$ é simétrica e positiva definida;

Propriedade 2: A matriz $N = \dot{B}_x(q) - 2C_x(q,\dot{q})$ é anti-simétrica [3].

Pode ser observado que tanto $B_x(q)$ quanto $g_x(q)$ são funções que dependem apenas de q

[2]**Matriz Jacobiana (Analítica)**: realiza a transformação entre o espaço de juntas (velocidades em [rad/s] e o espaço cartesiano (velocidades em [m/s]): $\dot{x} = J_A(q)\dot{q}$

[3]**Propriedade anti-simétrica para matrizes**: significa que podemos aproveitar esta propriedade ($a_{ij} = -a_{ji}$, onde a_{ij} correspondem aos elementos da matriz a) para fazer: $w^T\ \dot{B}(q) - 2C(q,\dot{q})\ w = 0$. Em robótica, se $w = \dot{q}$ então não se está restrito a escolher a matriz C como sendo símbolos de Christoftel, ou seja, C pode ser uma matriz qualquer. No caso, em robótica, se refere aos termos de Corioli e Centrífugos envolvidos na equação da dinâmica que determina os movimentos de um robô manipulador.

(posição angular das juntas), consequentemente, redes neurais estáticas são suficientes para modelar estas funções. Assuma então que $d_{xkj}(q)$ e $g_{xk}(q)$ possam ser modelados como:

$$\begin{aligned} d_{xkj}(q) &= \sum_l \theta_{kjl}\xi_{kjl}(q) + \varepsilon_{dkj}(q) = \theta_{kj}^T xi_kj(q) + \varepsilon_{dkj}(q) \\ g_{xk}(q) &= \sum_l \beta_{kl}\eta_{kl}(q) + \varepsilon_{gk}(q) = \beta_k^T \eta_k(q) + \varepsilon_{gk}(q) \end{aligned}$$

onde $\theta_{kjl}, \beta_{kl} \in \Re$ correspondem aos pesos sinápticos das redes neurais; $\xi_{kjl}(q), \eta_{kl}(q) \in \Re$ são as correspondentes funções Gaussianas com vetor de entrada q; e $\varepsilon_{dkj}(q), \varepsilon_{gk}(q) \in \Re$ correspondem aos erros de modelagem de $d_{xkj}(q)$ e $g_{xk}(q)$ respectivamente e se assumem que são limitados.

Enquanto isto, para a modelagem de $C_x(q, \dot{q})$ uma rede neural dinâmica dependente de q e $\dot{q}$ é necessária para modela-la. Assuma que $c_{xkj}(q, \dot{q})$ possa ser modelada como:

$$c_{xkj}(q, \dot{q}) = \sum_l \alpha_{kjl}\zeta_{kjl}(z) + \varepsilon_{ckj} = \alpha_{kj}^T \zeta_{kj}(z) + \varepsilon_{ckj}(z)$$

onde $z = [q^T \quad \dot{q}^T]^T \in \Re^{2n}$; $\alpha_{kjl} \in \Re$ correspondem aos pesos sinápticos; $\zeta_{kjl}(z) \in \Re$ corresponde às funções Gaussianas com vetor de entrada z; e, $\varepsilon_{ckj}(z)$ corresponde aos erros de modelagem de $c_{xkj}(q, \dot{q})$ que se assumem limitados.

Assim, a dinâmica no espaço tarefa do manipulador pode ser descrita como:

$$D_x(q)\ddot{x} + C_x(q, \dot{(}q)\dot{x} + g_x(q) = H_x$$

O controlador PD utilizado também foi projetado diretamente sobre o Espaço da Tarefa.

Simularam a proposta sobre um manipulador 2DOF especificando uma trajetória em (x, y) como sendo um círculo de raio $0.2[m]$ com centro localizado em $(x_1, x_2) = (1.0, 1.0)[m]$, com posição inicial do robô sendo $x(0) = (1.0, 1.0)[m]$. Para cada elemento de $B_x(q)$ e $g_x(q)$, uma rede neural RBF com 100 neurônios na camada oculta foi utilizada, enquanto que para cada elemento de $C_x(q, \dot{q})$, uma rede neural com 200 elementos na camada oculta foi utilizada. Os valores de μ e σ^2 foram fixados em 0.0 e 10.0 respectivamente. Os ganhos para o controlador foram fixados em: $K = diag[10.0]$ e $\Lambda = diag[5.0]$. O termo deslizante do controlador foi excluído fazendo $k_s = 0$ para mostrar a eficiência deste controlador neural. E para testar a capacidade do controlador em rejeitar perturbações, no tempo $t = 4.0[s]$ foi adicionada uma carga de $p_t = 0.5[Kg]$.

Enquanto as redes não estão sendo treinadas, ou seja, a lei de adaptação das redes não está ativa, o controlador se comporta simplesmente como um PD, o que resulta num significativo erro de seguimento e o controlador não consegue lidar com erros de aproximação nem perturbações de carga.

Quando o algoritmo de adaptação é ativado, fazendo $\Gamma_k = diag[0.02]$, $Q_k = diag[0.1]$ e $N_k = diag[5.0]$, o mecanismo de aprendizado das redes faz com que o erro de seguimento do sistema seja menor. A rede iniciou sem nenhum conhecimento à respeito do sistema, o que significa que as redes conseguiram controlar a dinâmica não linear desconhecida deste sistema. Diferentes desempenhos foram encontrados conforme se variava o ganho adaptativo e tamanho das redes neurais.

Perceberam que o controlador **não conseguiu aproximar: a matriz de Inércia nem a matriz relacionada aos termos de Coriolis**; não conseguiu fazer $\hat{B}_x(q) \rightarrow B_x(q)$, e $\hat{C}_x(q, \dot{q}) \rightarrow C_x(q, \dot{q})$. O **único termo que convergiu** para seu valor real foi o **termo gravitacional**, $\hat{g}_x(q) \cong g_x(q)$. Seus autores afirmam que isto se deve ao fato de que a **trajetória especificada não excitava continuamente as redes neurais** (trajetória com variações lentas, o que ocorre com freqüência na prática).

3.4.4 Controle Não Linear Baseado em Modelagem usando Redes Neurais

Lightbody and Irwin (1997) investigam em detalhes a possibilidade de se aplicar redes neurais na modelagem e controle adaptativo de sistemas não lineares. Tratam primeiramente da modelagem de sistemas não lineares baseado em redes neurais citando a capacidade de aproximação de redes multicamadas *perceptron*.

Propõem uma estrutura de rede (*feedforward*) embutida numa estratégia de controle adaptativa baseada num modelo direto Mencionam a dificuldade envolvida no treinamento *off-line* da rede que está embutida dentro da malha fechada de controle.

Apresentam uma nova metodologia para a modelagem da rede neural, baseada numa análise de sensibilidade do sistema em malha fechada, para permitir a retropropagação (*backpropagation*) dos erros através da planta para o controlador neural.

Finalmente sugerem uma nova estratégia de modelo interno de controle não linear (IMC = *Internal Model Control*) que utiliza um modelo neural não linear da planta para estimar parâmetros gerais sobre a região de operação não linear para um modelo adaptativo in-

terno linear, sem os problemas associados com algoritmos recursivos de identificação de parâmetros. Nova proposta para MRAC ⇒ *direct Model Reference Adaptative Control* – ver figura 3.17.

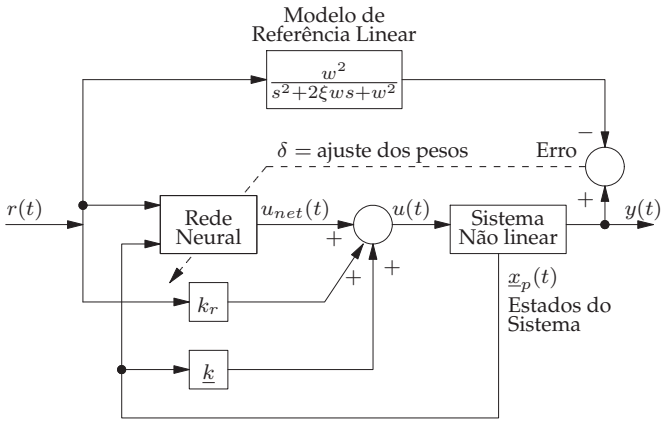


Figura 3.17: MRAC neural – proposta de Lightbody e Irwin (1997).

A proposta foi aplicada para controle de um tanque de reação por agitação contínua e comparada com um PI convencional.

3.4.5 Rede Neural numa estrutura NARMA

Noriega and Wang (1998) apresentam uma estratégia de controle adaptativo direto baseado em redes neurais para sistemas não lineares não conhecidos. O sistema é descrito por um modelo **NARMA** (ver (Zalzala, 1996; Warwick, 1996)) desconhecido. Uma rede neural *feedforward* é utilizada para aprendizado do sistema e empregada num esquema de otimização. A rede neural é tratada como um modelo neural do sistema.

Os sinais de controle são obtidos diretamente pela minimização tanto das diferenças instantânea ou acumulativa entre a referência e a saída do modelo neural. Uma vez que o algoritmo de treinamento garante que a saída do modelo neural se aproxime da saída atual do processo, Noriega and Wang (1998) mostram que o sinal de controle obtido pode também manter a saída real do sistema próxima da referência desejada. Aplicaram sobre um sistema de controle de fluxo de temperatura num processo químico.

Trabalharam com redes MLP, mas mencionaram o possível uso de redes RBF e baseadas em *splines*-B. O algoritmo de treinamento por retropropagação de erros (*backpropagation*) é utilizado *on-line* para estimar o sistema. Tomam o resultado da estimação realizada como um modelo dinâmico não linear do sistema e assim obtêm diretamente o sinal de controle

baseado na regra (bem conhecida) do gradiente descendente (*back propagation* tradicional). Apresentam a estrutura (figura) típica de controlador neural (atuam como um otimizador). A estrutura é mostrada na figura 3.18.

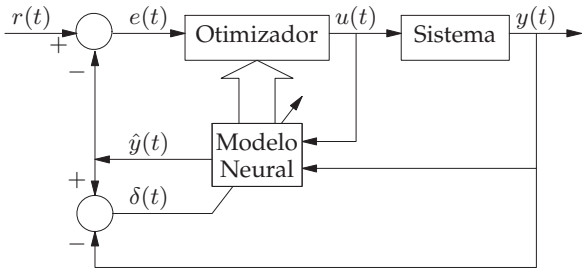


Figura 3.18: Controlador adaptativo neural genérico – proposta de Noriega e Wang (1998).

Abordaram questões de estabilidade usando métodos de Lyapunov e citam outros autores propondo tais controladores (ver (Lewis et al., 1995)). Discutem a possibilidade de se analisar a estabilidade de um controlador neural que adotaram, utilizaram função de Lyapunov, $V_1(\theta, \delta)$:

$$\Delta V_1(\theta, \delta) = V_1[\theta(t-1), \delta(t-1)] < 0$$

sobre a saída da modelador neural, $\hat{y}(t)$:

$$\Delta\hat{y}(t) = f_1(\phi) + g_1(\phi)\Delta u(t) + h.o.t.$$

onde $h.o.t.$ representa todos os possíveis termos de mais alta ordem.

Como o treinamento da rede neural pode somente alcançar um mínimo local, a análise de estabilidade realizada, somente determina condições para garantir a **estabilidade local do sistema em malha fechada**. Mas claro que mais investigações são necessárias de forma a provar a estabilidade assintótica global para este tipo de controlador em malha fechada. E que a dificuldade está na busca de uma análise de estabilidade para sistemas gerais de controle não linear. A estrutura é mostrada na figura 3.18.

3.4.6 Proposta de um controlador neural robusto de posição

Kim and Lewis (1999) propuseram uma arquitetura de controlador de posição neural robusto baseado em realimentação de sinais que não exige a medição das velocidades das juntas (o sinal proveniente do tacômetros, sensores de velocidade, estão normalmente con-

taminadas por ruído [4], o que limitaria os ganhos relativos ao erro de velocidade de um controlador (Weihmann, 1999).

Inicialmente duas redes neurais são utilizadas Uma para estimar a velocidade das juntas do manipulador e outra para fechar a malha do controlador baseado na realimentação do sinais de saída do sistema.

Um observador neural é utilizado para estimar a velocidade das juntas, como solução para não se trabalhar com o sinal ruidoso proveniente dos tacômetros do robô. Segundo Kim and Lewis (1999), um observador permite maior exatidão que uma simples diferenciação numérica. Este observador não linear (neural) foi inserido no laço de realimentação para garantir uma estabilidade assintótica local.

Os **pesos sinápticos** das redes do observador e do controlador são **ajustados *on-line***, sem a necessidade de uma fase de aprendizado *off-line*.

Kim and Lewis (1999) provam que todos os sinais envolvidos no sistema em malha fechada composto pelo robô, observador e controlador, são uniformemente finalmente limitados (*uniformly ultimately bounded* - ver definição em (Lewis et al., 1993, Cap. 1)).

Trata-se de uma contribuição ao projeto de seguidores de trajetória não lineares para sistemas robóticos. **Não é necessário** conhecimento algum à respeito da **modelagem dinâmica do robô**, assim a rede neural do controlador independe de modelo (*model-free*) e pode ser aplicável à outros sistemas não lineares que possuam uma estrutura similar à dos robôs manipuladores.

Comparado com um controlador adaptativo tradicional, este não exige que os parâmetros desconhecidos do sistema sejam lineares e não exige o (tedioso) cálculo da matriz regressora, $Y(q, \dot{q}, \ddot{q})$.

Controladores adaptativos tradicionais que usam observadores de estado com modos deslizantes (*sliding*) para estimar a velocidade das juntas, utilizam equações diferenciais que apresentam descontinuidades. Outras estruturas de controladores à estrutura variável empregam observadores de estado baseados em modelo e cujos parâmetros são estimativas fixas. O problema destes controladores é que os observadores propostos assumem que se conhece com exatidão os parâmetros da equação dinâmica do manipulador e assim sendo,

[4] Também no caso do robô real Inter/SCARA, a informação de velocidade é obtida derivando-se numericamente os sinais provenientes dos sensores de posição (*encoders* relativos) porque o sinal proveniente do tacogerador existente vêm muito contaminado por ruídos e componentes em altas freqüências

se exige um conhecimento exato da dinâmica do robô para que estes controladores funcionem bem. Na prática não é possível se obter este conhecimento exato. Desta forma, robustez e capacidade de adaptação são indispensáveis no projeto no controlador destes sistemas.

Usaram uma rede neural para fechar a malha de realimentação dos sinais de saída sem necessidade das informações à respeito das velocidades das juntas. Realizaram simulações sobre um controlador do tipo 2DOF.

3.4.7 Controle de Força e Posição de Robô Manipulador SCARA Utilizando Redes Neurais Artificiais

Battistela (1999) explorou técnicas para controle de posição e controle de força/posição de um robô manipulador SCARA utilizando redes neurais.

Controle neural de posição – Simulações e Experimentações

Realizou simulações e experimentações para controle de posição das 4 juntas do robô Inter/SCARA (descrito à frente no item 4.2). Implementou estratégia de aprendizado baseado no erro de realimentação (*feedback error learning* (Er and Liew, 1997) – estrutura apresentada nos itens 3.3.4 e 3.4.2. Comparou o controlador de posição neural com outros dois controladores: (1) um controlador clássico do tipo PD (Proporcional-Derivativo), sem compensação da gravidade, e (2) um controlador por dinâmica inversa contendo na malha externa um controlador *feedforward* clássico PID (Proporcional-Integral-Derivativo) – INV+PID. Um controlador do tipo PD apresenta erro em regime permanente (problema de regulação), mas garante estabilidade global. Já um controlador PID possui a propriedade de ser localmente estável, resolve o problema de regulação (erro ponto à ponto), mas não apresenta resposta transitória uniforme, ou seja, não desempenha um bom seguimento da trajetória, que piora no caso de incertezas paramétricas e perturbações externas. A rede neural aplicada na arquitetura de aprendizado por realimentação do erro, se mostrou como uma forma interessante de se modelar a dinâmica inversa do manipulador. A estrutura proposta para o controlador neural de posição trabalha em paralelo com um controlador PD convencional, e é mostrada na figura 3.19.

Esta estrutura foi inspirada na estrutura EFELC de Er and Liew (1997) (já descrita na seção 3.4.2 - figura 3.16) e em outras arquiteturas de aprendizado pela realimentação do erro

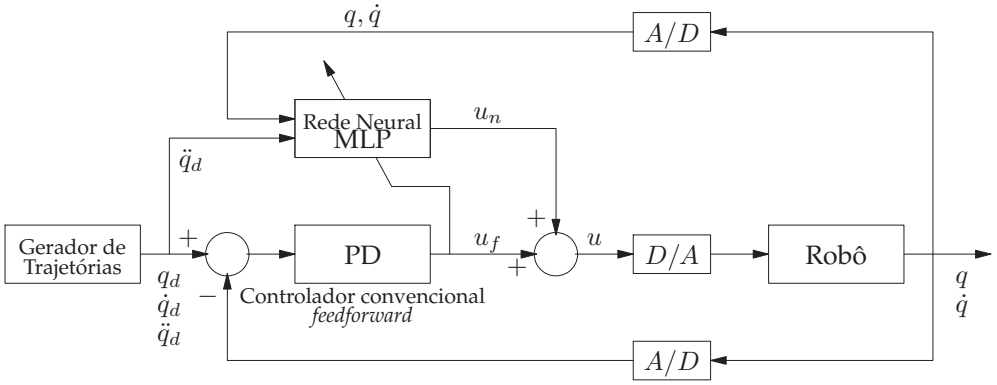


Figura 3.19: Controlador neural de posição proposto por Battistela (1999).

(Warwick, 1996; Morris and Khemaissia, 1996; Fukuda et al., 1996). A rede usa a saída de controle gerada pelo PD, u_f, como sinal de erro para seu treinamento (de modo semelhante à estrutura EFELC proposta por Er and Liew (1997)). Destacou como principal vantagem da estrutura proposta o fato desta **não exigir treinamento *off-line*** da rede neural. A rede neural tenta compensar as não-linearidades do sistema e incertezas paramétricas existentes na sua modelagem, ou seja, desempenha o papel da dinâmica inversa do sistema:

$$\hat{B}(q)\ddot{q} + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q) = \tau$$

onde $\hat{B}$, $\hat{C}$ e $\hat{g}$ correspondem respectivamente aos parâmetros estimados (indicados pelo "^") para a matriz de inércia do manipulador, termos de Corioli e Centrífugos do sistema, e vetor dos termos gravitacionais do sistema; e τ corresponde aos torques que deveriam ser desenvolvidos por cada uma das juntas do manipulador. Todos os dados desta equação se referem ao espaço das juntas do manipulador (posições angulares, q; velocidades angulares, $\dot{q}$ e aclerações angulares $\ddot{q}$ de cada uma das juntas). Normalmente numa tarefa de manipulação envolvendo interação com o meio, as posições para os movimentos não restritos são fornecidos no espaço cartesiano 3D, o que exige transformações de espaços de coordenadas entre o espaço de juntas e o espaço da tarefa (cartesiano).

Battistela utilizou uma rede *feedforward* multicamadas de *perceptrons* (MLP) composta de 12 neurônios na sua camada de entrada (4 dados para posição angular da junta i: q_i; 4 dados para velocidade angular da junta i: $\dot{q}_i$ e outros 4 para aceleração desejada para cada uma das juntas: $\ddot{q}_{di}$ – o índice i se refere ao número da junta, variando de 1 à 4) + 8 neurônios na primeira camada invisível + 5 neurônios na segunda camada invisível (ou intermediária) e

$+$ 4 neurônios na camada de saída da rede (correspondendo aos torques à serem gerados em cada uma das 4 juntas deste robô). Esta arquitetura foi a que apresentou os melhores resultados em testes *off-line*. Os pesos sinápticos foram inicializados com valores randômicos entre ± 0.1, e os *bias* foram inicializados com valores na faixa de ± 0.05. Percebeu que para taxas de aprendizado com valores acima de 0.1, o sistema se instabilizava porque a rede saturava muito rapidamente oscilando entre os valores mínimos e máximos de torques permitidos para cada uma das juntas. Testou 4 diferentes algoritmos de aprendizado: (1) o tradicional *back propagation*, (2) *AB-propagation*, (3) RPROP e (4) *quick propagation*. A idéia era se avaliar algoritmos mais rápidos de treinamento que o tradicional *back propagation*. Não obteve resultados promissores com os algoritmos *RPROP* e *quick propagation*, que apresentaram baixa capacidade de generalização (se mostraram muito sensíveis à escolha do conjunto de treinamento) e resposta muito rápida (a saída da rede "saturava" nos valores máximos de torque permitidos para este robô) o que levava o sistema de controle integrado PD + RN à instabilidade. O problema se deveu ao fato de que nos instantes iniciais, o controlador PD realiza uma ação de controle muito expressiva, o que fazia com que as redes interpretassem este elevado valor como um elevado sinal de erro, ocasionando ajustes relativamente grandes nos pesos da rede, independente de uma baixíssima taxa de aprendizado e de ajustes em outros parâmetros de treinamento da rede. Problema semelhante já foi detectado por Er and Liew (1997) para a estrutura convencional FELC (ver figura 3.15). Optou pelo algoritmo *back propagation* dada sua simplicidade em comparação ao algoritmo *AB-propagation* que exige um parâmetro extra para treinamento da rede. Durante o treinamento *on-line*, a taxa de aprendizado adotada girou em torno de 0.001. Battistela comprovou que valores acima de 0.05 tornavam o sistema oscilatório – a rede ficava com seu aprendizado muito acelerado.

Apesar de a rede iniciar o controle sem nenhum conhecimento à respeito do sistema, a estabilidade inicial fica garantida pela ação de controle PD. **Nas simulações** realizadas para os controladores de posição, o controlador convencional PD exibiu erro em regime permanente na terceira junta porque não compensava os efeitos gravitacionais para esta junta. O controlador por dinâmica inversa funcionou conforme o esperado realizando a correta compensação dos termos não-lineares e os referentes aos acoplamentos existentes entre as juntas. Simulou **variações paramétricas** modificando os momentos de inércia (I_i), massas (m_i) e comprimentos (l_i) para as juntas: $i = 1, 3$ e 4. Os ganhos do controlador PD fo-

ram suficientes para garantir bom desempenho mesmo frente à variações paramétricas. Já o controlador INV+PID (que leva em conta dados do modelo ideal do robô), mesmo com a presença do PID não conseguiu linearizar nem desacoplar o novo comportamento dinâmico do robô, apresentando erros em regime permanente. Simulou também **perturbações**, aplicando no instante $t = 3$ [s], um torque contrário na junta 1 correspondente à 20% do valor máximo capaz de ser desenvolvido por esta junta e no instante de tempo $t = 4$ [s] fez a mesma coisa para a junta 2. Frente às perturbações, o controlador PD não conseguiu responder dentro do tempo de resposta desejado e os erros em regime permanente para a terceira junta ainda foram mais agravados. O controlador INV+PID conseguiu estabilizar o sistema mas às custas de um *overshoot* muito elevado e mesmo assim, também ultrapassou o tempo de resposta desejado. O controlador neural de posição foi capaz de compensar tanto as variações paramétricas, quanto as perturbações, apresentando ainda baixos valores de erro de seguimento de trajetória. Simulou ainda **efeitos de carregamento** na última junta do robô Inter/SCARA, variando as massas e momentos de inércia das terceira e quarta juntas. O controlador PD não foi capaz de compensar o acréscimo no efeito gravitacional, gerando um erro ainda maior em regime permanente para a terceira junta. O controlador INV+PID apresentou erro em regime permanente para a terceira junta. O controlador neural de posição foi capaz de compensar o efeito extra gravitacional, ainda dentro da especificação do tempo de resposta desejado. A rede neural funcionou como um compensador de gravidade, suprimindo as deficiências do controlador convencional PD que trabalha em paralelo. Porém Battistela percebeu que o controlador neural de posição gerava ações de controle mais oscilatórias, mas com menores erros de seguimento de trajetória. E comprovou que o desempenho do controlador por dinâmica inversa (INV+PID), depende muito da precisão do modelo dinâmico do manipulador levado em consideração na sua lei de controle.

Nas **experimentações** sobre o robô Inter/SCARA real, realizou testes com o controlador PD convencional sem compensação de gravidade e com o controlador neural de posição sendo proposto. Limitou ainda as perturbações à uma variação de apenas 10% do valor máximo de torque permitido para a primeira junta. Empregou ainda **dois valores diferentes de taxa de aprendizado** do controlador neural: valores baixos em torno de 0.03 foram utilizados nos instantes iniciais de aprendizado quando a ação do controlador paralelo PD era elevada. Mas quando o efetuador final já estava bem mais próximo do valor final desejado,

a ação deste controlador diminuía (menor ação de controle $\Rightarrow$ menor erro de treinamento) e o aprendizado da rede se tornava muito lento, assim, nesta situação, a taxa de aprendizado era modificada para um valor mais elevado em torno de 0.7. Nos ensaios experimentais realizados, o controlador PD implementado exibiu erro em regime permanente para a terceira junta (efeito gravitacional) e frente à perturbação exercida sobre a primeira junta, não foi capaz de rejeitá-la. Battistela destacou o fato de que não é possível se atingir erro nulo em regime permanente utilizando um controlador PD, mas mesmo assim, este controlador continua sendo muito utilizado na prática quando não se exige precisão superior à 10^{-3} ([rad] para as juntas 1, 2 e 4 e [m] para a junta 3). O controlador neural alcançou maior precisão de posicionamento, com erros menores que 10^{-5}, também porque Battistela limitou o treinamento da rede para valores de erro de posição menores que 10^{-5}. O controlador neural foi capaz de compensar a perturbação na primeira junta porque cada variação de erro desencadeava uma variação na saída do controlador paralelo PD, ocasionando um refinamento da rede fazendo com que esta compensasse a perturbação exercida sobre o manipulador. **Resumindo**: para o caso do controlador neural de posição trabalhando em paralelo com um controlador convencional PD, Battistela comprovou a robustez deste controlador frente à incertezas paramétricas e perturbações, com baixo erro de seguimento de trajetória, se bem que às custas de uma ação de controle mais oscilatória.

Controle híbrido neural de força/posição – Simulações

Battistela (1999) simulou 3 estruturas de controle diferentes para as três primeiras juntas do robô Inter/SCARA (a quarta junta é a responsável pela definição da orientação do efetuador final). O primeiro controlador era um híbrido dinâmico (sua lei de controle realiza a compensação dinâmica do manipulador, portanto exige um modelo dinâmico exato do manipulador, nem sempre disponível na prática), contendo na malha de controle de posição, um controlador convencional PD (seria um controlador INV+PD para malha de controle de posição) e na malha de controle de força, um controlador convencional PID (seria um controlador INV+PID para a malha de controle de força). O segundo controlador simulado foi um híbrido neural levando em conta o modelo dinâmico do manipulador (ver figura 3.20) e o terceiro controlador também era um híbrido neural, porém não exigia um modelo dinâmico do manipulador.

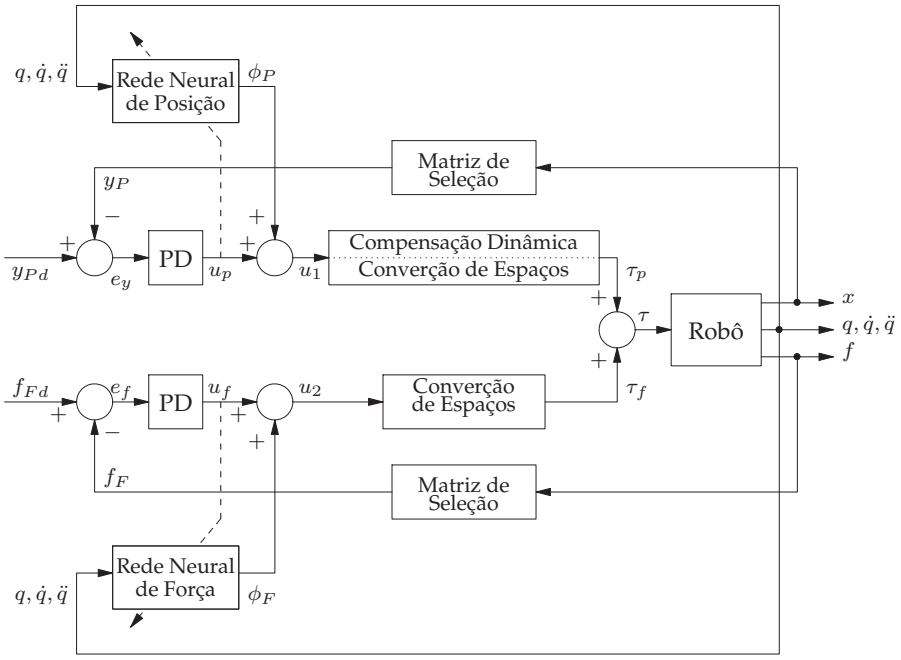


Figura 3.20: Controlador híbrido neural de força/posição considerando modelo dinâmico do manipulador, proposto por Battistela (1999).

Para ambos os controladores neurais empregou controladores PD tanto para a malha de controle de posição quanto para a malha de controle de força. Simulou o manipulador realizando movimento livre descrito por *senos* e *cosenos* no plano XY e movimento restrito no eixo Z, ou seja, aplicando força sobre um meio modelado simplesmente (linearmente) pela constante de rigidez, $K = 10^4$ [N/m].

Os algoritmos de aprendizado acelerados, *quick propagation* e RPROP, tal qual ocorreu no caso do controlador neural de posição, apresentaram comportamento oscilatório indesejável: as saídas das redes saturavam rapidamente nos limites dos torques de cada uma das juntas, em função do elevado valor de atualização do pesos sinápticos das redes, o que levava o sistema à instabilidade. Battistela optou mais uma vez pelo algoritmo de aprendizado *back propagation* dado a necessidade de um aprendizado mais lento e gradual que não saturasse a rede nos instantes iniciais de controle. Para os dois controladores híbrido neurais utilizou redes MLP, sendo que ambas as redes possuíam o mesmo número de neurônios na camada de entrada (9, definidos de forma semelhante ao caso do controlador neural de posição[5], com agora para as 3 primeiras juntas do robô). Já as camadas invisíveis das redes

[5]Observação: no caso do robô manipulador real Inter/SCARA não estão disponíveis acelerômetros nas juntas do robô, como normalmente ocorre em outros robôs industriais, devido ao alto custo deste sensor. No máximo se pode ter acesso às acelerações desejadas para cada uma das 4 juntas do robô, $\ddot{q}_{di}$.

para a malha de controle de posição e malha de controle de força são diferentes. A rede da malha de controle de posição continha 7 e 5 neurônios respectivamente na sua primeira e segunda camadas invisíveis e mais 2 neurônios na sua camada de saída: um para definir o torque para o eixo X e outro para definir o torque para o eixo Y. A rede da malha de controle de força, empregava apenas uma camada invisível com 6 neurônios e 1 neurônio na camada de saída: torque à ser desenvolvido no eixo Z. Como ocorrido no caso do controlador neural de posição, foi obrigado a inicializar os pesos sinápticos e de *bias* das redes com valores randômicos baixos entre ± 0.05. Para a rede da malha de controle de posição, adotou o valor de 0.0005 para a taxa de aprendizado, e para a rede da malha de controle de força trabalhou com o valor de 0.001.

Nas **simulações** realizadas, gerou uma perturbação no instante $t = 2.5$ [s] na junta 1, correspondendo à um torque aplicado no sentido contrário atingindo um valor de 45% do valor máximo gerado por esta junta, e no instante de tempo $t = 3$ [s] introduziu uma segunda perturbação na terceira junta, aplicando um torque reverso no valor de 5% do máximo suportado por esta junta. O controlador híbrido dinâmico convencional apresentou bom desempenho nos instantes iniciais da simulação (sem as perturbações), mas depois não foi capaz de corrigir o erro de posição em regime permanente provocado pelas perturbações. A estratégia de controle por dinâmica inversa (compensação dinâmica) não conseguiu linearizar e desacoplar o novo modelo dinâmico do manipulador. E o controlador PD embutido na malha deste controlador de posição não foi capaz de rejeitar estas perturbações. Este último problema poderia ter sido minimizado com a adoção de um controlador PID. Ainda em relação à malha de controle de posição, o controlador híbrido neural sem modelo dinâmico apresentou comportamento oscilatório nos instantes iniciais (de funcionamento e em seguida às perturbações). Battistela justificou este comportamento pelo fato da malha de controle de posição não prever o modelo dinâmico do manipulador, o que fazia com que esta rede necessitasse de um período de tempo maior para o ajuste de seus pesos sinápticos. Mas mesmo assim, este controlador permitiu alcançar erros de seguimento de trajetória de posição menores que os comparados ao controlador híbrido dinâmico convencional. Em relação à malha de controle de força, o controlador PID embutido na estrutura do controlador híbrido dinâmico tradicional (INV+PID) foi capaz de compensar as perturbações (erro nulo em regime permanente), mas exigiu um tempo maior para correção da sua trajetória

que os controladores híbridos neurais. Os controladores híbridos neurais (com e sem levar em conta o modelo dinâmico do manipulador) conseguiram rejeitar razoavelmente bem as perturbações – as redes se adaptaram rapidamente às novas condições de operação. Simulou também **variações paramétricas** nos momentos de inércia das juntas 1, 3 e 4 e nos comprimentos dos elos das primeira e segunda juntas. Esta variação não afetou o consideravelmente o desempenho de nenhum dos 3 controladores sendo avaliados. Mas Battistela percebeu que o controlador híbrido dinâmico convencional mostrou um erro pequeno em regime permanente, o que não ocorreu com nenhum dos controladores neurais. As redes neurais foram capazes de adaptar o controlador frente à estas variações, anulado o que poderia ser considerado como incertezas de modelagem. Novamente o controlador híbrido neural sem o modelo dinâmico apresentou oscilações de posição no plano XY nos instantes iniciais de funcionamento deste controlador. Em relação ao **esforço de controle**, percebeu que os controladores neurais apresentaram comportamento mais oscilatório nos períodos transitórios – instantes em que as redes estavam ajustando seus pesos para as novas situações. Sendo que o controlador híbrido neural sem o modelo dinâmico, exibiu maiores comportamentos oscilatórios no período inicial de seguimento da trajetória devido ao fato de suas redes ainda não serem capazes de compensar a dinâmica do manipulador. Battistela concluiu que o controlador híbrido neural com compensação dinâmica permitia uma ação de controle mais suave, porém exige o modelo dinâmico do manipulador, mesmo que inexato (situação que freqüentemente ocorre na prática). Verificando o erro médio de seguimento de trajetórias de posição e força, percebeu que as duas estratégias de controladores neurais exibiram erros cerca de 50% menores que o controlador híbrido dinâmico convencional. **Conclusões finais:** o controlador híbrido dinâmico convencional exige o modelo dinâmico exato do manipulador e não é capaz de lidar com perturbações ou variações paramétricas de forma a anular os erros de posicionamento (apesar do PID da malha de controle de força conseguir compensar estas variações para o caso de controle de força). O que indica que uma estrutura de controle de força/posição capaz de se ajustar *on-line* à incertezas de modelagem, variações paramétricas e perturbações, é muito interessante e bem vindo. Surge então a proposta de controladores híbridos neurais: um que ainda exige o modelo dinâmico do manipulador e outro que não. Mesmo o controlador híbrido neural sem modelo dinâmico ainda é capaz de lidar com variações paramétricas e perturbações às custas de uma ação de controle mais

oscilatória e não tão suave nos períodos transitórios como o é capaz de fazer o outro controlador híbrido neural que leva em consideração o modelo dinâmico (mesmo inexato) no manipulador. Porém Battistela citou a provável dificuldade de se provar a estabilidade do sistema: robô manipulador + PD + Rede Neural.

3.5 Controladores *Fuzzy*

A medida que os processos industriais foram se tornando mais complexos, passando a exigir uma flexibilidade maior na especificação de parâmetros e menores custos, os projetistas começaram a considerar técnicas de controle outras que não somente as convencionais. Entre as recentes técnicas computacionais de controle desenvolvidas, controladores *fuzzy* estão sendo mais utilizados industrialmente, principalmente nos casos em que técnicas de controle convencionais são difíceis de aplicar (da Mota Almeida et al., 2000).

Entre as abordagens de controladores *fuzzy* que surgiram, destacamos variações para controladores *fuzzy*-PID.

3.5.1 Controlador *Fuzzy* – Introdução

Um controlador por lógica *fuzzy* (FLC=*Fuzzy Logic Controller*) está baseado em regras heurísticas que faz uso da teoria de conjuntos difusos desenvolvida por Zadeh para traduzir regras de controle lingüísticas numa estratégia de controle coerente (Kosko, 1992; Yager and Filev, 1994). O diagrama em blocos de um controlador *fuzzy* é mostrado na figura 3.21.

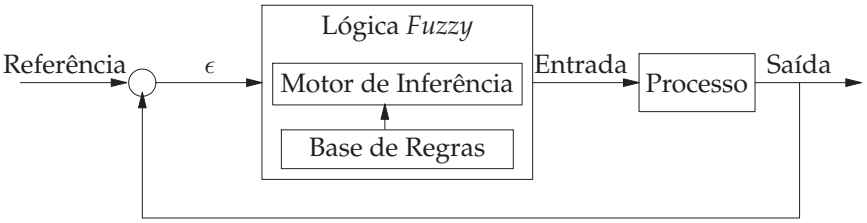


Figura 3.21: Controlador geral por lógica *fuzzy*.

Uma lei de controle *fuzzy* é descrita por uma base de conhecimentos constituída de regras SE ... ENTÃO com predicados vagos e um mecanismo de inferência de lógica *fuzzy*. O principal componente de um controlador *fuzzy* é sua base de regras. Ela é formada por regras que descrevem relações entre a entrada e e a saída u do controlador. Em geral, a lei

de controle de um controlador *fuzzy* geral pode ser representada por:

$$u[k] = \mathcal{F}(e[k], de[k], ie[k])$$

onde a função $\mathcal{F}$ é não linear e descrita pelos parâmetros e regras *fuzzy*. Cada regra do controlador *fuzzy* é caracterizada por uma parte SE, denominada de antecedente e pela parte ENTÃO, chamada de conseqüente. O exemplo de uma base de regras é mostrada na tabela 3.2.

$u[k]$		$e[k]$		
		N	Z	P
$de[k]$	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Tabela 3.2: Exemplo de base de regras *fuzzy*.

As variáveis lingüísticas dos antecedentes e conseqüentes são representadas por funções de pertinência. A figura 3.22 mostra um exemplo de funções de pertinência para a tabela 3.2.

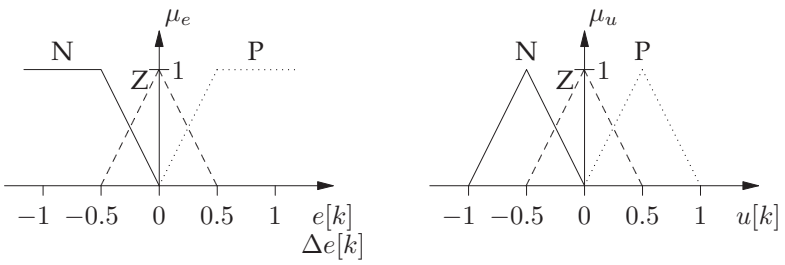


Figura 3.22: Funções de pertinência para $e[k]$, $de[k]$ e $u[k]$.

O motor de inferência de um controlador *fuzzy* é composto pelo: a) processo de "*fuzzyficação*"das variáveis de entrada; b) base de regras e c) processo de "*defuzzyficação*"da saída (resultado das regras). A saída de um controlador geral *fuzzy* é dado por:

$$u[k] = Defuzz\Big\{R \circ Fuzz\{e[k]\} \circ Fuzz\{de[k]\} \circ Fuzz\{ie[k]\}\Big\} \tag{3.7}$$

onde $Fuzz\{\cdot\}$ é o operador de *fuzzyficação*, $Defuzz\{\cdot\}$ é o operador de *defuzzificação*, $\circ$ é o operador de composição das relações *fuzzy* e R trata do operador relacionado ao controlador *fuzzy*.

3.5.2 Controladores PID *fuzzy*

Entre as técnicas de controladores convencionais, os controladores Proporcionais-Integrativos-Derivativos (PID) são os que encontraram maior aceitação e aplicação na indústria nas últimas décadas, seja por causa da sua simplicidade ou robustez . Assim, controladores *fuzzy*-PID foram desenvolvidos inspirados nos controladores convencionais PID, ressaltando que o controlador *fuzzy*-PID pode ser sintonizado baseado nos ajustes de um controlador PID (Qin, 1994; Yager and Filev, 1994). As abordagens para controladores *fuzzy*-PID compreendem:

a) Controlador *fuzzy*-PI + convencional D;

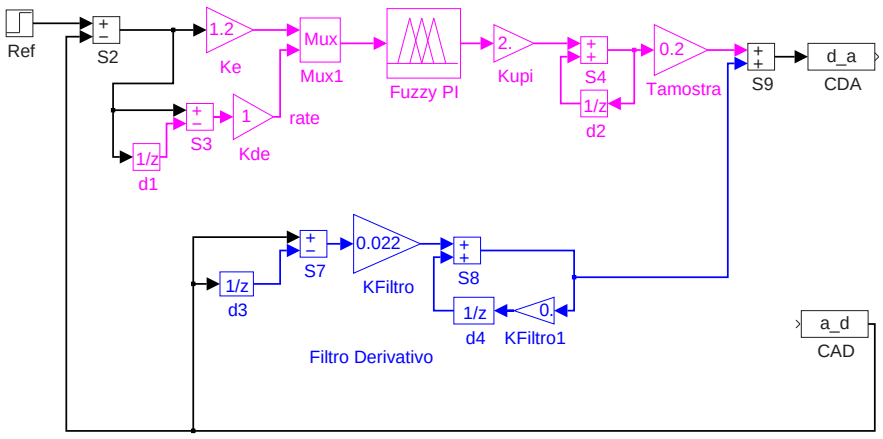
b) Controlador *fuzzy*-PD + convencional I;

c) Controlador *fuzzy*-PD + *fuzzy*-I;

d) Controlador *fuzzy*-PI + *fuzzy*-PD;

3.5.3 Controlador *Fuzzy*-PI + Convencional D

O diagrama em blocos do controlador é mostrado na figura 3.23, e está baseado na estrutura proposta por (Qin, 1994).



Obs: o bloco ”CDA” e ”CAD” correspondem respectivamente ao Conversor de sinal Digital para Analógico e ao Conversor de sinal Analógico para Digital.

Figura 3.23: Diagrama em blocos do Controlador *Fuzzy*-PI + Convencional D.

A lei de controle do controlador *Fuzzy*-PI+D é dada por:

$$u[k] = u[k-1] + TDefuzz\Big\{K_e \circ Fuzz\{e[k]\} \wedge K_{de} \circ Fuzz\{de[k]\}\Big\} + u_D[k] \tag{3.8}$$

onde T é o período de amostragem e:

$$u_D[k] = k_{f1}u[k-1] + k_{f2}(y[k] - y[k-1])$$

e onde k_{f1} e k_{f2} são as constantes do filtro derivativo.

3.5.4 Controlador *Fuzzy*-PD + Convencional I

Este controlador consiste de um *Fuzzy*-PD em paralelo com um controlador Integral tradicional. O diagrama em blocos deste controlador é mostrado na figura 3.24. A ação integral é necessária para evitar erros em regime permanente.

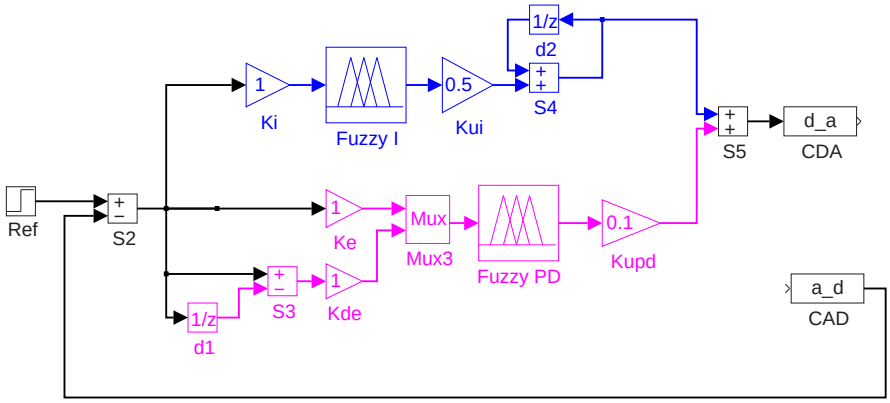


Figura 3.24: Controlador *fuzzy*-PD + *fuzzy*-I.

A lei de controle é dada por:

$$u[k] = u[k-1] + \frac{KT}{T_i}e[k] + Defuzz\Big\{K_e \circ Fuzz\{e[k]\} \wedge K_{de} \circ Fuzz\{de[k]\}\Big\} \tag{3.9}$$

onde K e T_i representam o ganho proporcional e o constante de tempo do integrador respectivamente.

3.5.5 Controlador *Fuzzy*-PD + *Fuzzy*-I

Este controlador foi proposto por (Li, 1997). A lei de controle é resultado do somatório da ação de controle *fuzzy*-PI e *fuzzy*-I. Duas bases regras são necessárias para cada uma das partes deste controlador. A saída deste controlador é dada por:

$$u[k] = u[k-1] + TDefuzz\Big\{K_i \circ Fuzz\{e[k]\}\Big\} + Defuzz\Big\{K_e \circ Fuzz\{e[k]\} \wedge K_{de} \circ Fuzz\{de[k]\}\Big\} \tag{3.10}$$

O diagrama em blocos deste controlador aparece na figura 3.25.

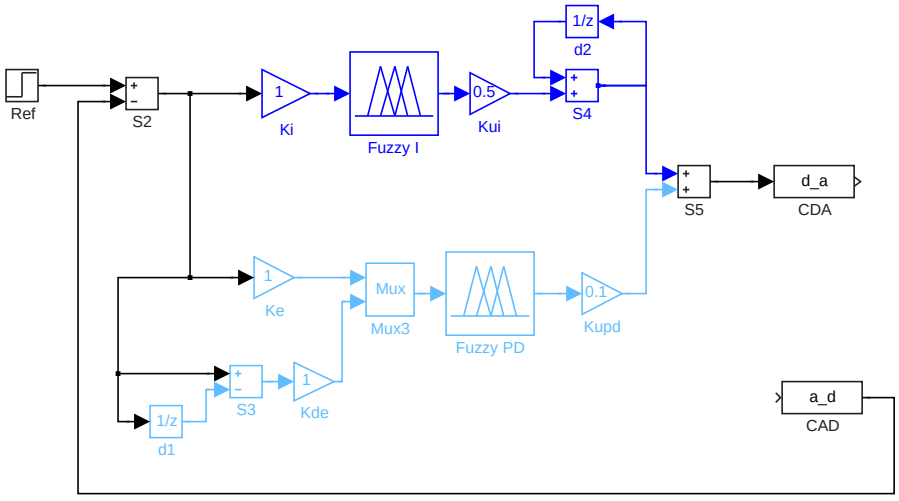


Figura 3.25: Controlador *Fuzzy*-PD + *Fuzzy*-I.

3.5.6 Controlador *Fuzzy*-PI + *Fuzzy*-PD

O projeto deste controlador foi apresentado por (Kwok et al., 1990). Consiste de um *fuzzy*-PI em paralelo com um *fuzzy*-PD. Este controlador é constituído por duas bases de regras: uma para a parte relacionada ao controlador *fuzzy*-PD eoutra ao *fuzy*-PI. O diagrama em blocos deste controlador aparece na figura 3.26.

Um dos problemas críticos no controle *fuzzy* está relacionado com a sintonia e calibração das regras *fuzzy* (Zadeh, 1996). Mas nos útlimos anos, métodos que solucionem em parte este problema vêm sido pesquisadas (Zadeh, 1996; Qin, 1994; Kwok et al., 1990).

3.5.7 Controlador Adaptativo *Fuzzy* com Aprendizado Neural

Lin and Lin (1997) apresentam um controlador neural, usando RN do tipo ART (mul-

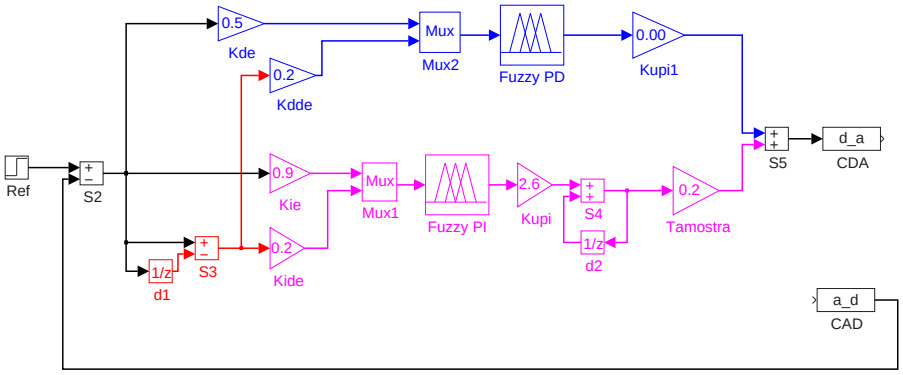


Figura 3.26: Controlador *Fuzzy*-PD + *Fuzzy*-PI.

ticamadas *feedforward*) com algoritmo de aprendizado *on-line* substituindo um controlador lógico *fuzzy*. A rede proposta (FALCON = *Fuzzy Adaptative Learning Control Network*) pode ser comparado em estrutura e propriedades de aprendizagem à um tradicional controlador lógico *fuzzy*. Propuseram um algoritmo *on-line* de aprendizado para a estrutura/parâmetros, que denominaram de FALCON-ART. Este algoritmo gera automaticamente toda a estrutura do controlador FALCON. Ele combina o esquema de aprendizado *backpropagation* para realizar a estimação de parâmetros e o algoritmo *fuzzy* ART para estimar sua estrutura. Este algoritmo possui algumas características importantes como:

1. Antes de tudo, eles particionam o espaço de estados de entrada e espaço do controle da saída usando hiper-*boxes fuzzy* irregulares baseado na distribuição dos dados de entrada. Em muitos sistemas de controle *fuzzy* ou neurais, os espaços de entrada e saída são sempre particionados em "*grids*".

   A medida que o número de variáveis de entrada/saída aumentam, o número de espaços particionados cresce de maneira combinatória. Para se evitar o problema de explosão combinatória, o algoritmo FALCON-ART divide os espaços de entrada/saída de forma flexível, baseado na distribuição do dados utilizados para treinamento.

2. O algoritmo FALCON-ART pode criar e treinar um controlador FALCON de maneira altamente autônoma. No início não existe funções de pertinência (*membership functions*), partições *fuzzy* ou regras difusas. Elas são criadas assim que o primeiro padrão de treinamento aparece. Assim, os usuários deste tipo de controlador **não necessitam** fornecer **nenhum conhecimento** *a priori* **ou** mesmo alguma **informação inicial sobre o sistema à ser controlado**.

Lin and Lin (1997) afirmam que o algoritmo FALCON-ART pode dividir *on-line* os espaços de entrada/saída, sintonizar as funções de pertinência, encontrar as regras difusas mais apropriadas e dinamicamente suprimir regras redundantes à medida que vai recebendo mais dados para treinamento.

Lin and Lin (1997) realizaram simulações numéricas que demonstraram a **quase** efetividade deste controlador nas aplicações simuladas. Simulações realizadas:

1. Identificação de um sistema dinânimo, seguindo o formato:

$$\hat{y}(k+1) = \hat{f}\left(u(k), u(k-1), \ldots, u(k-p+1), y(k), y(k-1), \ldots, y(k-q+1)\right)$$

   onde a planta simulada segue a equação:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k)$$

   e cuja entrada é definida como: $u(k) = sin(2\pi K/100)$.

2. Predição de uma série temporal caótica; Utilizaram a série temporal de Mackey-Glass, gerada pela seguinte equação diferencial:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

   onde: $\tau > 17$ (adotaram $\tau = 30$ nas simulações deles).

3. Controle de retrocesso de um caminhão (*control for backing up the truck*), baseado no problema proposto por Nguyen and Widrow (1990).

4. Controle de um sistema de barra e bola (*ball and beam system*).

Não foi observado nada à respeito da carga numérica exigida por este controlador, nem sobre sua complexidade de implementação.

3.5.8 Controle de Força *Fuzzy* para Robôs Industriais

Lin and Huang (1998) desenvolveram uma estrutura de controle de força hierárquico constituída de um sistema de controle em alto-nível baseado em lógica *fuzzy* e outro de

baixo-nível, já existente, para controle de movimento do robô manipulador. Um esquema de controle de força *fuzzy* foi adotado para se adaptar às várias condições de contato. A capacidade de adaptação do sistema foi possível através do ajuste do fator de escala do controlador lógico *fuzzy*. Aplicaram a estrutura proposta num robô industrial da Mitsubishi: MELFA RV-M1 equipado com sensor de força/torque. Testaram esta proposta fazendo o robô manter se deslocar mantendo uma certa força, sobre 4 superfícies de características diferentes: mola, esponja, borracha1 e borracha2. Para o caso do ensaio sobre a borracha1 perceberam um grande *overshoot* e a ocorrência de um ciclo-limite. Para os outros casos, perceberam um desempenho satisfatório. Num outro ensaio, introduziram perturbação no sistema (forçando o sistema com a mão) que o sistema *fuzzy* foi capaz de rejeitar, reagindo de forma mais suave (sem *overshoot*) e menos erro em regime permanente que um simples e tradicional controlador PI ou PID. Também testaram o controlador de força *fuzzy* numa tarefa de inserção de pinos (*peg-in-hole*. O controlador foi usado para ajustar a orientação do pino no encaixe, mantendo como torque desejado um valor nulo; esta estratégia foi últil para lidar com erros de posicionamento. Conseguiram completar a tarefa de inserção.

Capítulo 4

Descrição do Robô e dos Controladores Propostos

4.1 Introdução

Este capítulo apresenta as implementações realizadas tanto para os controladores convencionais de posição quanto para os controladores integrados de posição (que mesclam um controlador convencional com alguma rede neural). Faz o mesmo também com relação aos controladores de posição/força explorados neste trabalho. E termina apresentando índices de desempenho propostos neste trabalho como uma tentativa de análise mais objetiva dos resultados que podem ser obtidos.

4.2 Descrição do robô Inter/Scara

O robô a ser utilizado nos experimentos foi projetado e construído pelo Instituto de Robótica da Universidade Técnica Federal de Zurique (ETH), na Suíça (ver figura 4.1) e foi adquirido em parceria entre o Departamento de Automação e Sistemas (DAS) e Departamento de Engenharia Mecânica (EMC) para fins de pesquisa. Sua principal característica é ser de **arquitetura aberta**, ou seja, possibilita a programação e implementação de algoritmos de controle e não somente a programação de trajetória ou seqüencias de comandos como na maioria dos robôs industriais. Atualmente se encontra instalado no Laboratório de Automação Industrial (LAI) do DAS.

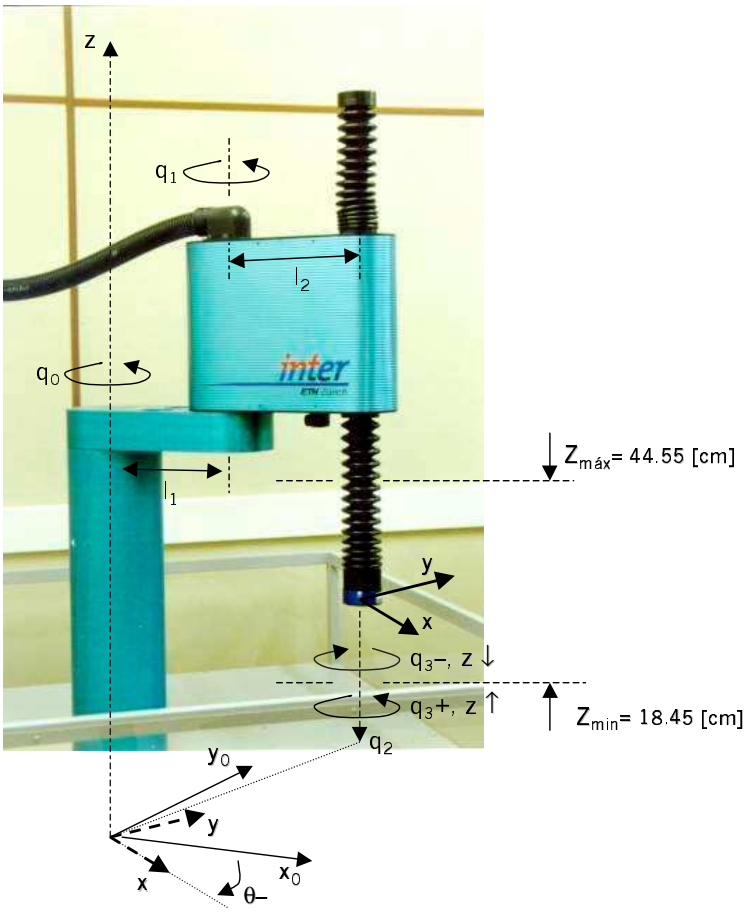


Figura 4.1: Robô Inter/Scara e seu sistema de coordenadas.

Este é um robô do tipo SCARA (*Selective Compliant Articulated Robot Arm*), com 4 graus de liberdade (4DOF). As duas primeiras juntas são de rotação, girando em torno de eixos verticais e trabalhando portanto num plano horizontal (plano XY), como se fosse um robô 2DOF (com 2 graus de liberdade). A terceira junta é de translação, também dita prismática permitindo deslocamentos no sentido vertical, ao longo do eixo z. E a quarta junta, de rotação, permite definir a orientação (ângulo θ) do efetuador final deste robô em relação ao eixo vertical Z. Esta estrutura permite grande rigidez no sentido vertical e certa complacência no plano horizontal.

O robô está equipado com o sensor de força JR3 (fabricante homônimo) que possibilita capturar 6 tipos de informações: forças e momentos em cada um dos 3 eixos – $f_{Sens} = [\ f_x \quad f_y \quad f_z \quad \mu_x \quad \mu_y \quad \mu_z\ ]^T$, ver figura 4.2 (maiores detalhes recorrer à *http://www.jr3.com/* ou (Weihmann, 1999)).

Os sinais são medidos através de *strain-gages* montados em ponte, amplificados e con-

Figura 4.2: Sensor de força JR3 instalado no robô.

vertidos para informações digitais de força e momentos pelo próprio circuito eletrônico localizado dentro do sensor. Os valores máximos de sinais que podem ser medidos através deste sensor são mostrados na tabela 4.1.

Forças [N]		
f_x	f_y	f_z
100	100	200

Momentos [Nm]		
μ_x	μ_y	μ_z
6,3	6,3	6,3

Tabela 4.1: Capacidade de carregamento do sensor de força JR3.

A figura 4.3 mostra as ferramentas já desenvolvidas para uso com este robô, como resultado dos trabalhos de Bier and Guenther (2000).

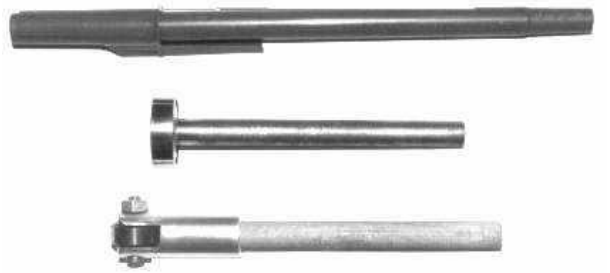


Figura 4.3: Ferramentas implementadas para o robô.

A interface entre o usuário e este robô é realizada por meio do sistema operacional em tempo real orientado a objetos: XO/2 [1] (Siegwart et al., 1998; Golin et al., 2000; Weihmann, 1999). Este sistema foi desenvolvido no Instituto de Robótica do ETH (Instituto Federal de Tecnologia de Zurich, Suiça) e é programado numa linguagem de alto nível chamada XO/2, que é uma extensão da liguagem Oberon orientada a objetos sucessora do Modula2 e Pascal.

[1] Cabe ressaltar que durante o desenvolvimento deste trabalho, o sistema XO/2 sofreu uma atualização, juntamente com seu software original utilizado para comandar o robô, passando a ser usada a versão:"Oberon System 3 for Windows^{TM} Win32 2.3 (25.2.1999) on Windows NT 5.0". Desde fins de 2002 o XO/2 já está na sua 4^a versão (favor verificar em: http://www.oberon.ethz.ch/).

Esta linguagem é bastante recente e ainda está em fase de aperfeiçoamento mas vêm sendo aplicada com sucesso nos sistemas mecatrônicos sendo desenvolvidos por este instituto de robótica em Zurique (Siegwart et al., 1998). O XO/2 é ainda um sistema de desenvolvimento híbrido que roda num computador hospedeiro (*host*) com plataforma Macintosh, Sun e PC (Siegwart et al., 1998). Suporta a família de processadores Motorola 68.XXX ou PowerPC (que é o caso da CPU contida no gabinete de controle do robô Inter/SCARA).

No caso do robô Inter/SCARA, sua CPU trabalha com a versão do XO/2 Real-Time direcionada para rodar sobre a plataforma PowerPC. Esta CPU interage com outro sistema XO/2 versão hóspede para Windows, que roda em microcomputadores PC (plataforma Intel). Esta versão é preparada para trocar dados com a CPU do robô via rede Ethernet e ainda é capaz de gerar código executável para o PowerPC através de um *cross*-compilador. Cada nova modificação em algum código do robô é editada no computador PC , compilada de forma a gerar código para o PowerPC, e "empacotada" na forma de um único arquivo chamado "Megula" que é automaticamente lido pela CPU do robô toda vez que seu gabinete é (re-)inicializado. A conexão entre o computador hospedeiro (*host*) e o gabinete do robô é realizada inicialmente via porta serial (para configuração inicial) e posteriormente através de rede local rodando protocolo TCP/IP (o gabinete do robô trabalha com um número fixo de IP). Cabe ressaltar que existem versões ditas "nativas" do XO/2 para Windows preparadas para rodar sobre o Windows, ou seja, gerando código executável diretamente para plataforma Intel, sendo previsto até mesmo interfaces com as API's e DLL's mais comuns do Windows – esta versão é conhecida pelo nome de XOberon System 3 Plug-In. A versão 4 do XO/2 pode até mesmo rodar de maneira autônoma como Sistema Operacional sobre a plataforma Intel, sem necessidade do Windows.

Siegwart et al. (1998) destacam as duas mais importantes características do sistema XO/2: 1) a interface entre o processador alvo e o computador hospedeiro é feita através de uma janela terminal (figura 4.4, janela "Log03.150.162.140.200"), e 2) "*linkagens*"[2] incrementais permitem a adição e remoção (*on-line*) de tarefas sem a interrupção de outras que já estejam sendo processadas. Os comandos fornecidos pelo usuário normalmente são agrupados num mesmo arquivo do tipo "Tool" conforme pode ser visto na figura 4.4, janelas: "Inter.Tool" (comandos básicos de inicialização e movimentação do robô),

[2] "linkar" = interligação dos módulos compilados entre si.

"TesteArquivos.Tool" (comandos para captura de dados do robô) e "Force.Tool" (comandos para testar controle de força). Informações complementares sobre o sistema XO/2 podem ser obtidas via consulta a uma lista de usuários e desenvolvedores deste sistema em: *Oberon@inf.ethz.ch* originalmente criada pelo *staff* do instituto ETH e consultas no site: http://www.lists.inf.ethz.ch/mailman/listinfo/oberon ou no site www.XO2.org.

No caso do robô Inter/SCARA, as rotinas de controle, *Watchdog* (teste de comunicação entre gabinete do robô e PC *host*), rotinas de proteção/segurança, sincronização dos sensores (inicialização do robô), acionamento dos motores e comunicação com o robô Inter/SCARA são realizadas usando a linguagem de programação que acompanha o sistema XO/2 (Weihmann, 1999; Golin et al., 2000). No conjunto de rotinas para o XO/2 que chegou acompanhando o robô Inter/SCARA (e no "pacote" da versão que foi atualizada), foram previstas apenas as rotinas necessárias para controle no espaço de juntas e no espaço operacional, usando controlador do tipo PD que trabalha no espaço de juntas. Entretanto não

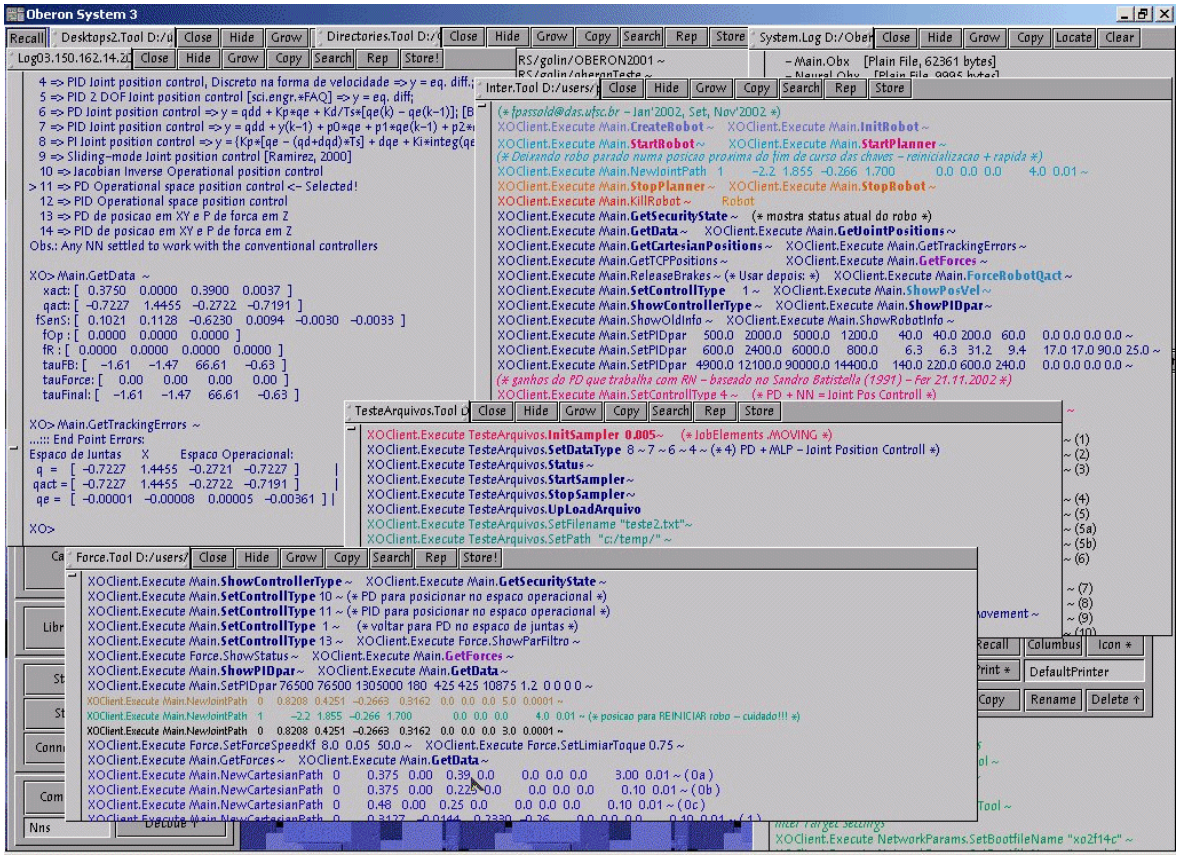


Figura 4.4: Figura ilustrando ambiente de operação do XO/2.

existia nenhuma rotina já pronta para rodar algoritmo de controle híbrido de posição/força.

Alguns dados paramétricos do robô foram fornecidos pelo fabricante mas com elevado grau de incerteza, especialmente o momento de inércia e centros de massa de cada junta (Weihmann, 1999), o que inviabiliza de certa forma, simulações usando ferramenta numérica como o MATLAB/Simulink, uma vez que os ganhos encontrados para controladores simulados desta forma não podem ser diretamente aproveitados num experimento real com o robô (as diferenças se tornam muito grandes).

As juntas de rotação deste robô são equipadas com transmissões do tipo "*harmonic drive*", que se caracteriza por ser composta por engrenagens com formato elíptico entre o eixo do motor e o da junta, o que faz com que, durante uma rotação, haja sempre apenas dois pontos de contato, ou seja, garante um atrito mínimo relativo a transmissão do torque do motor para a junta. Já a terceira e quarta juntas deste robô são compostas por um conjunto mecânico conhecido pelo nome de "*ball-screw-spline*". Este conjunto é o responsável pelo deslocamento vertical da terceira junta e rotação (mudança de orientação) da última junta. A terceira junta, vertical, contém no seu interior um fuso com uma ranhura helicoidal que faz contato com dois pares de ranhuras axiais distintos: o "*ball-screw*" e o "*ball-spline*". O movimento deste fuso é provocado pela rotação aplicada em separado ou em conjunto nestes pares de ranhuras. Isto implica num **acoplamento mecânico entre a terceira e quarta juntas deste robô**. Ou seja, se for aplicada tensão apenas no motor da junta 2, este conjunto se desloca no sentido vertical, o que também provoca uma mudança na orientação (giro) da última junta (3). Por causa deste acoplamento mecânico entre as duas últimas juntas, já existe alguma compensação prevista no *software* que acompanha este robô[3], à fim de garantir apenas movimento vertical (deslocamento da terceira junta) ou apenas rotacional (mudança na orientação final do efetuador, ou giro da última junta). É interessante ressaltar que este acoplamento nas últimas duas juntas acarreta dificuldades extras na implementação de algoritmos de controle de força/posição que levem em consideração as 4 juntas deste robô, além de acarretar uma dificuldade extra na modelagem e simulação de controladores para as 4 juntas dada ainda a incerteza de paramétrica. Mais informações a respeito deste robô podem ser encontradas em (Weihmann, 1999; Golin et al., 2000).

[3]O acoplamento mecânico existente entre as últimas 2 juntas deste robô é "desacoplado" por software apenas para comandos enviados à terceira junta (q_2) no espaço de juntas ou comandos em z enviados no espaço operacional. Mas comandos enviados no espaço de juntas para q_3 alteram a altura do robô em z, no espaço operacional. Maiores detalhes podem ser encontrados em (Golin et al., 2000).

São expostos a seguir os controladores utilizados neste trabalho com o robô Inter/SCARA.

4.3 Rotina de Controle

A rotina original de controle do robô Inter/SCARA implementada no XO/2 previa somente controle PD de velocidade no espaço de juntas: PROCEDURE (h: ControlFBHdl) Run() – módulo: Robot.Mod, configurada como uma tarefa de tempo-real para ser rodada originalmente a cada 1 [ms], tendo sido reservado um mínimo de 200 [ms] no gerenciador de tarefas do XO/2 para execução desta tarefa e *deadline* (tempo limite para execução da tarefa) especificado para 0.5055 [ms] (isto é, se a execução desta tarefa não terminar num prazo inferior ao *dealine* especificado, uma exceção à nível de *software* ocorre, e a tarefa é colocada em modo suspenso, ou seja, deixa de ser executada – o que neste caso, pode significar a completa perda de controle sobre a movimentação do robô). Esta tarefa é a responsável também por ler os encoders do robô, capturar as informações vindas do sensor de força do robô, calcular as velocidades atuais desenvolvidas pelas juntas, $\dot{q}$ (através de derivada numérica simples, sem uso de filtro), atualizar o erro de posicionamento no espaço de juntas, $\tilde{q}$, e originalmente ainda aplicar a lei de controle baseado em PD de velocidade e atualizar a posição atual do robô no espaço operacional, x, calculando a cinemática direta do robô ($x = K(q, dx)$, levando em conta o *offset* (dx), ocasionado pela introdução de uma ferramenta ao robô). A figura 4.5 mostra na forma de um diagrama em blocos simplificado a maneira como é realizado o controle em tempo-real do robô Inter/Scara.

Esta rotina foi expandida de forma a permitir a seleção *on-line* de outros algoritmos de controle. Para tanto foi acrescentado o atributo ControllerType ao objeto robot. Esta rotina de controle foi expandida então com o algoritmo mostrado na figura 4.6, página 109 a seguir.

Para poder ser utilizada a nova tarefa de tempo-real relacionada com o controlador, certos métodos do objeto robot tiveram de ser expandidos ou mesmo criados. O Módulo original ScaraRobot.Mod foi expandido para dar suporte ao cálculo dos novos estados necessários do robô, conforme segue:

- (robot: ScaraRobot) **DirKinematicsFull***(VAR q, dx, qd, x, xd: ARRAY OF LONGREAL);
  Atualiza: $x = K(q) + K_{Offset}(q, dx)$, e; $\dot{x} = J_{A_{Full}}(q, dx)\,\dot{q}$. O código para a

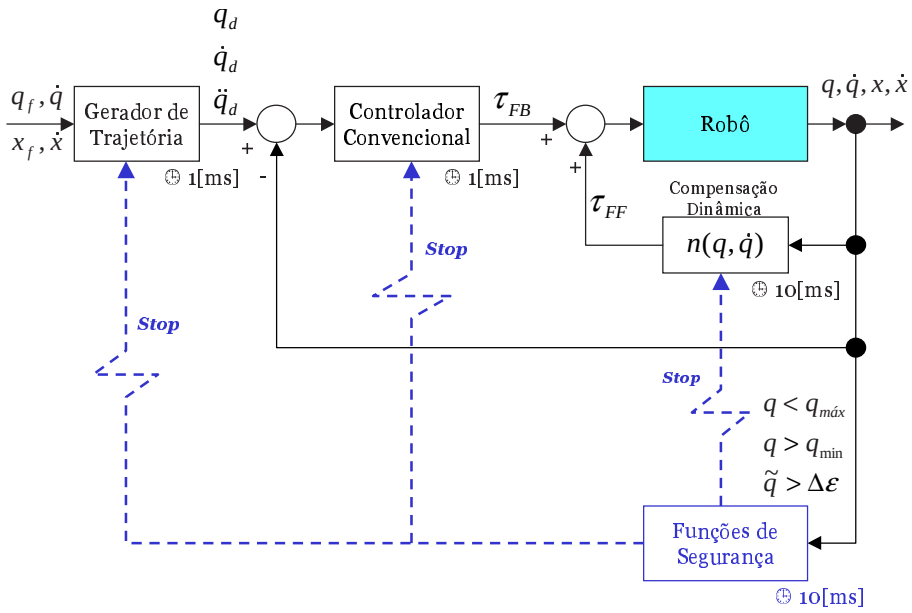


Figura 4.5: Diagrama em blocos do controle em tempo-real feito no robô Inter/Scara.

determinação de $\ddot{x}$ também já está previsto nesta nova rotina.

- (robot: ScaraRobot) **EvaluateInvJaFull*** (VAR q, dx: ARRAY OF LONGREAL);
  Atualiza a nova matriz de parâmetros do objeto robot relacionada com o cálculo da Jacobina Inversa: robot.InvJaFull – ver equação A.11 no Apêndice A. Esta matriz é necessária para o cálculo de: $\Delta q = J^{-1}_{A_{Full}}(q)\,\Delta x$.

- (robot: ScaraRobot) **InvKinematicsFull*** (VAR x, dx, xd, xdd, q, qd, qdd: ARRAY OF LONGREAL): LONGINT;
  Completa o cálculo da cinemática inversa do robô, determinando:

  a) $q = K^{-1}(x)$: já existente no método original InvKinematics (ver (Weihmann, 1999; Golin et al., 2000)). É utilizado o método geométrico para determinação da cinemática inversa do robô (ver em (Sciavicco and Siciliano, 1996; Barrientos et al., 1997)). No caso do robô estar operando muito próximo do limite do seu espaço operacional exterior, um mensagem de advertência (o *flag* robot.securityState.desc=Robot.outOfWorkspaceErrDesc), é gerada para indicar que não foi possível determinar a cinemática inversa do robô, neste caso, é retornado como novo valor de q_1, o valor calculado no período de amostragem anterior! Esta rotina leva em consideração o acoplamento mecânico existente entre as duas

PROCEDURE (h: ControlFBHdl) **Run**();

⋮

IF robot.**ControllerType** ≠ 9 **THEN** (* *controle no espaço de juntas....* *)

CASE robot.ControllerType **OF**

0: (* *Simples controlador proporcional: estável* *)

$$\tau_{FB} = B(q)\, K_p\, \tilde{q}$$

| 1: (* *Controlador original do robô... PD de velocidade* *)

$$\tau_{FB} = B(q)\,(K_p\tilde{q} - K_d\dot{q}_d)$$

| 2: (* *PID baseado na equação do PD + ação integral sobre erro* *)

$$\tau_{FB} = B(q)\left(K_p\tilde{q} - K_i\int\tilde{q}\,dt + K_d\dot{\tilde{q}}\right)$$

⋮

| 6: (* *PD discreto [Batistella, 1999, eq. 4.7, pag. 54 e eq. 5.1, pag. 73]* *)

$$\tau_{FB} = B(q)\left[\dot{q}_d + K_p\tilde{q} + K_d T_s\left(\tilde{q}[k] - \tilde{q}[k-1]\right)\right]$$

| 7: (* *PID discreto [Batistella, 1999, eq. 5.3, pag. 74]* *)

$$\ddot{q}_u[k] = \ddot{q}_u[k-1] + \ddot{q}_d + p_0\tilde{q}[k] + p_1\tilde{q}[k-1] + p_2\tilde{q}[k-2]$$

$$\tau_{FB} = B(q)\,\ddot{q}_u[k]$$

onde: $p_0 = K_p\left(1 + K_d/T_s\right)$

$p_1 = K_p\left(1 + 2K_d/T_s - K_i T_s\right)$

$p_2 = \left(K_p K_d\right)/T_s$

| 8: (* *PI discreto [Batistella, 1999, pag. 130]* *)

$$\tau_{FB} = B(q)\left\{K_p\left[\tilde{q} - T_s\left(\dot{q}_d + \dot{q}\right)\right] + \left(\dot{q}_d - \dot{q}\right) + K_i T_s\int\tilde{q}\,dt\right\}$$

| 9: (* *sliding mode control [Ramirez et al., 2000]* *)

$$\tau_{FB} = -K\cdot\mathrm{sat}(s,\varepsilon)$$

onde: $\mathrm{sat}(s,\varepsilon) = \begin{cases}\mathrm{sign}(s), & se\ |s| \geq \varepsilon \\ s/\varepsilon, & se\ |s| < \varepsilon\end{cases}$

END (* *fim do case por tipo de controlador no espaço de juntas* *)

(* *fim dos casos para controladores no espaço de juntas* *)

ELSE (* *do IF do teste de controladores: espaço de juntas × espaço operacional* *)

(* *Segue controladores no Espaço Operacional* *)

$$\left.\begin{aligned} x &= K(q) + K_{Offset}(q, dx) \\ \dot{x} &= J_{A_{Full}}(q, dx)\,\dot{q}\end{aligned}\right\}$$ (* *método: robot.DirKinematicsFul* *)

$$\tilde{x} = x_d - x$$

CASE robot.ControllerType **OF**

10: (* *Controlador por Jacobiana Inversa no Espaço Operacional [Sciavicco e Siciliano, 1996, fig. 6.29]* *)

$$\Delta q = J_A^{-1}(q)\,\tilde{x}$$

$$\tau_{FB} = K_p\,\Delta q$$

| 11, 13: (* *Controlador PD de Posição no Espaço Operacional* *)

$$\tau_{FB} = J_{A_{Full}}\left(K_p\tilde{x} - K_d\dot{x}\right)$$ (* *[Sciavicco e Siciliano, 1996, eq. (6.102)]* *)

| 12, 14: (* *Controlador PID de posição no espaço operacional* *)

$$\tau_{FB} = J_{A_{Full}}\left(K_p\tilde{x} + K_i\int\tilde{x}\,dt - K_d\,\dot{\tilde{x}}\right)$$

ELSE (* *controlador Proporcional no Espaço Operacional* *)

$$\tau_{FB} = K_p J_{A_{Full}}^T(q)\,\tilde{x}$$

END

END (* *fim do teste para controladores convencionais* *)

⋮

Figura 4.6: Controladores convencionais de posição implementados no robô Inter/SCARA.

últimas juntas do robô Inter/SCARA;

b) $\dot{q} = J_{A_{Full}}^{-1}(q)\,\dot{x}$: não existente na programação original do robô;

c) $\ddot{q} = J_{A_{Full}}^{-1}\big(\ddot{x} - \dot{J}_{A_{Full}}\,\dot{q}\big)$: também não disponível no código original do robô.

A dedução das duas últimas equações para o caso específico do robô Inter/SCARA pode ser acompanhado no Apêndice A (equações A.14 e A.15 respectivamente).

4.4 Controladores Convencionais de Posição

4.4.1 Controlador PD no Espaço de Juntas

O robô Inter Scara vêm acompanhado de fábrica apenas de um controlador PD no Espaço de Juntas, baseado na equação 2.2, mas sem o termo relativo à compensação gravitacional já que as duas últimas juntas deste robô foram projetadas de forma a compensar mecanicamente este efeito (arranjo mecânico conhecido como sistema *"ball-screw-spline"* – ver (Weihmann, 1999)). Desta forma a lei de controle deste controlador se torna:

$$u = K_p\tilde{q} + K_d\dot{\tilde{q}} \tag{4.1}$$

que é aplicada acompanhada de opcional compensação dinâmica (termo $\hat{\eta}(\cdot)$) na equação a seguir:

$$\hat{B}(q)u + \hat{\eta}(q, \dot{q}, \ddot{q}) = \tau \tag{4.2}$$

No sistema implementado no XO/2 que veio instalado com o robô, o termo $\hat{\eta}(\cdot)$ calcula os torques necessários para realizar a compensação dinâmica do robô, (Hüpi and Gruener, 2001), implementada na forma de uma tarefa em tempo real: (h: ControlFFHdl) Run() que a cada 1 [ms] realiza o cálculo:

$$\tau_{FF} = m\,\ddot{q} + d\,\mathrm{Sgn}(\dot{q}) + k\,q = \hat{\eta}(\cdot) \tag{4.3}$$

onde: m corresponde aos momentos de inércia das juntas; d se refere ao um fator de amortecimento; e k se refere à constante de rigidez. Estes 3 parâmetros compõem o vetor $p = [m \quad d \quad k]$ relacionado com a estimação de parâmetros do robô. Por *default* este parâmetro é inicializado com zero ($p = 0$), ou seja, não é realizada compensação dinâmica até

que o usuário ative as equações relacionadas com a adaptação dinâmica destes termos, introduzindo valores não nulos no vetor de ganhos de adaptação dos parâmetros, λ, que também por *default* inicia nulo ($\lambda = 0$). As equações relacionadas com a adaptação paramétrica (ajuste do vetor p) foram implementadas na forma de outra tarefa em tempo-real:(h: ControlAdaptHdl) Run() que roda à cada 10 [ms]. Para maiores detalhes acerca do procedimento opcional para compensação dinâmica que acompanha a nova versão do código para o robô Inter/Scara são indicados os itens 1.5.3.5 e 1.5.3.6 de (Hüpi and Gruener, 2001). Como os vetores p e λ partem nulos com o início da operação do robô, estes termos permanecem nulos enquanto se utiliza este controlador PD com uma rede neural, isto é, durante o uso da RN com um controlador convencional a compensação dinâmica permanece desabilitada.

Segundo Hüpi and Gruener (2001), esta rotina de compensação dinâmica visa gerar o torque extra necessário para: (a) no modo estático do robô equilibrar os efeitos gravitacionais e (b) durante a movimentação do robô suprimir os atritos envolvidos nas juntas do robô. Em aplicações de controle de força, normalmente as velocidades de deslocamento nas direções que não são restringidas são baixas (o robô pode estar realizando uma tarefa de desbaste, e neste caso a força de contato contra o meio e velocidade com que é realizada o desbaste varia de acordo com o material (Kiguchi and Fukuda, 1997) e principalmente neste caso (de controle de força) pode ser interessante algum mecanismo de compensação de atritos relacionados com as juntas do robô e no contato do robô com o meio.

Para este controlador foram mantidos os ganhos já sintonizados de fábrica para este robô, conforme demonstrado na tabela 4.2. Note que este tipo de controlador não garante erro nulo em regime permanente e sua estabilidade é apenas global (Sciavicco and Siciliano, 1996).

Parâmetro	Junta 0	Junta 1	Junta 2	Junta 3
K_p	4900	12100	90000	14400
K_d	140	220	600	240

Tabela 4.2: Ganhos (originais) do Controlador PD de posição no espaço de juntas.

4.4.2 Controlador PID no Espaço de Juntas

Simplesmente à lei de controle anterior (eq. 4.1) foi acrescentada ação integral sobre o erro de posição no espaço de juntas, baseado na equação:

$$u = K_p\,\tilde{q} + K_i \int \tilde{q}\,dt + K_d\,\dot{\tilde{q}} \tag{4.4}$$

Esta lei de controle foi implementada de forma digital na forma que segue:

$$u_j[k] = K_{p_j}\,\tilde{q}_j\,[k] + K_{i_j} \sum_{k=0}^{n} \tilde{q}_j[k]T + K_{d_j} \frac{(q_j[k] - q_j[k-1])}{T} \tag{4.5}$$

onde o índice j refere-se a junta sendo controlada ($j = 1 \ldots 4$). Esta implementação se baseou no PID de modo posicional conforme apresentado no Apêndice B, equação B.3.

Para suprimir efeitos de saturação na ação integral, um simples algoritmo de saturação foi aplicado (figura 4.7). Os valores da ação integral foram limitados ao valores: iLimit=10000.

```
robot.integ= robot.integ + robot.qe;
IF robot.integ > 0) AND (robot.integ > iLimit) THEN
   robot.integ= iLimit;
ELSIF (robot.integ < 0) AND (robot.integ < -iLimit) THEN
   robot.integ= -iLimit;
END;
```

Figura 4.7: Saturador para ação integral excessiva.

A sintonia deste controlador será feita com base nos parâmetros sugeridos por Ziegler-Nichols (Seborg et al., 1989; Franklin et al., 1994; Nise, 2000) para ajuste de controladores PID (ver Apêndice B).

4.4.3 Controlador por Modos Deslizantes no Espaço de Juntas

Foi implementado um modelo de controlador robusto trabalhando por modos deslizantes (*sliding-mode control*), baseado no trabalho já realizado por Ramírez et al. (2000) para este mesmo robô. A lei de controle deste controlador é dada pela equação:

$$u = -K\,Sat\Big(\frac{s}{\epsilon}\Big) \tag{4.6}$$

onde para redução do efeito de *chattering* foi definida a função de saturação, $Sat(\cdot)$, como sendo:

$$Sat\Big(\frac{s}{\epsilon}\Big) = \begin{cases} \text{sign}\big(s\big), & \text{se } |s| \geq \epsilon \\ \frac{s}{\epsilon}, & \text{se } |s| < \epsilon \end{cases} \tag{4.7}$$

e como projeto da superfície de deslizamento para este controlador, a função s foi definida como:

$$s(q, \dot{q}, t) = c(q - q_d) + (\dot{q} - \dot{q}_d) \tag{4.8}$$

onde:
- c é um vetor 4×1 (um parâmetro para cada junta) e se refere ao ganho proporcional sobre o erro de posição no espaço de juntas ($c > 0$);
- ϵ também um vetor de dimensão 4 responsável por definir a valor da camada limite relacionada com o projeto do controlador por modos deslizantes.
- K também um vetor de dimensão 4, relacionado com os ganhos do controlador.

4.4.4 Controle por Jacobiana Inversa no Espaço Operacional

A lei de controle para este controlador foi baseada em Sciavicco and Siciliano (1996):

$$u = K_p \, J_A^{-1}(q) \, \big(x_d - x\big) \tag{4.9}$$

4.4.5 Controle PD no Espaço Operacional (PD-Op)

A lei de controle utilizada para implementar o PD no espaço operacional é a mesma já descrita no Capítulo 2, equação 2.5 (pág. 15) com exceção do termo relativo à compensação gravitacional, $g(q)$, que já é realizado de forma mecânica no robô Inter/SCARA:

$$u = J_A^T(q) \, K_p \, \tilde{x} - J_A^T \, K_d \, J_a(q) \, \dot{q} \tag{4.10}$$

a parte da expressão acima relacionada com o termo $\tilde{x}$, expandida especificamente para o caso do robô Inter/Scara, gera o seguinte impacto na ação de controle:

$$\underbrace{\begin{bmatrix} (-l_1 s0 - l_2 s01 + k_1)K_{p_0} & (l_1 c0 + l_2 c01 + k_2)K_{p_1} & 0 & K_{p_3} \\ (-l_2 s01 + k_1)K_{p_0} & (l_2 c01 + k_2)K_{p_1} & 0 & K_{p_3} \\ 0 & 0 & K_{p_2} & 0 \\ k_1 K_{p_0} & k_2 K + p_1 & k K_{p_2} & K_{p_3} \end{bmatrix} \cdot \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix}}_{J_A^T(q)\cdot K_P \cdot \tilde{x}} \tag{4.11}$$

e a parte da equação 4.10 relacionada com o termo $\dot{q}$, expandida, causa este outro impacto na ação de controle:

$$\underbrace{\begin{bmatrix} (-l_1 s0 - l_2 s01 + k_1)K_{d_0} & (l_1 c0 + l_2 c01 + k_2)K_{d_1} & 0 & K_{d_3} \\ (-l_2 s01 + k_1)K_{d_0} & (l_2 c01 + k_2)K_{d_1} & 0 & K_{d_3} \\ 0 & 0 & K_{d_2} & 0 \\ k_1 K_{d_0} & k_2 K + d_1 & k K_{d_2} & K_{d_3} \end{bmatrix} \cdot \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}}_{J_A^T \cdot K_D \cdot J_a(q) \cdot \dot{q}}$$

4.4.6 Controlador PID no Espaço Operacional (PID-Op)

Em relação à lei de controle anterior, à equação 4.10 foi acrescentado a ação integral sobre o termo $\tilde{x}$, implementado de forma numérica, ficando a lei de controle na forma:

$$u = J_A^T(q) \left(K_p \, \tilde{x} + K_i \int \tilde{x} \, dt \right) - J_A^T(q) \, K_d \, J_A(q) \cdot \dot{q} \tag{4.12}$$

Pretendeu-se com a ação integral, cancelar o erro em regime permanente no espaço operacional.

4.5 Controlador Integrado de Posição

Propõe-se neste trabalho utilizar uma rede neural para realizar a compensação dinâmica dentro de uma malha de controle de posição convencional. Isto é, a rede neural trabalha em paralelo com um controlador convencional de posição (de preferência algum que já vinha sendo utilizado) e passam a ser gerados 2 torques: um proveniente da rede neural: τ_{NN} e

outro proveniente do controlador convencional: τ_{FB}. A soma destes dois torques, compõe o toque final, τ_{Final} que é enviado a cada uma das juntas do robô. Cada junta possui seu proprio controlador convencional, um P ou PD ou PID convenientemente sintonizado para cada junta do robô. A esta estrutura é adicionada apenas uma rede neural que recebe como informações de entrada de dados: $x_{NN} = [\, q \quad \dot{q} \quad \ddot{q}_d \,]$. São passados os valores instantâneos e atuais das posições e velocidades angulares de cada uma das juntas do robô. Apenas para o vetor de aceleração é passado o valor desejado, $\ddot{q}_d$, e não o valor atual devido à falta de um acelerômetro (sensor) presente nas juntas do robô. Cabe ressaltar que mesmo o vetor de velocidade atuais do robô, $\dot{q}$ é determinado através de um simples derivada numérica sem o uso de algum filtro para limitar o ruído induzido nesta forma de medição, já que na prática, o tacogerador presente nas juntas do robô se revelou muito ruidoso e testes experimentais concluíram ser preferível o uso de simples derivada numérica sobre as informações geradas diretamente à partir dos *encoders* presentes nas juntas do robô (Golin et al., 2000).

O que distingue esta aplicação de rede neural para a área de controle de outras formas de implementação está no sinal utilizado para realimentação de seus pesos sinápticos. Diferente de outras abordagens mais convencionais, que normalmente usariam, neste caso, o erro de seguimento de trajetória, esta rede utiliza como informação de erro para ajuste de seus pesos o próprio valor gerado pelo controlador convencional com o qual trabalha em paralelo, numa arquitetura conhecida como "aprendizado baseado no erro de realimentação"(*feedback error learning*). Esta arquitetura já se mostrou bastante promissora em outros trabalhos apenas com variações no "pacote" de entrada de dados da rede (Battistela, 1999; Er and Liew, 1997; Watanabe, 1996; Carelli et al., 1995). Finalmente a figura 4.8 mostra na forma de diagrama de blocos, o controlador integrado que foi implementado no robô Inter/Scara.

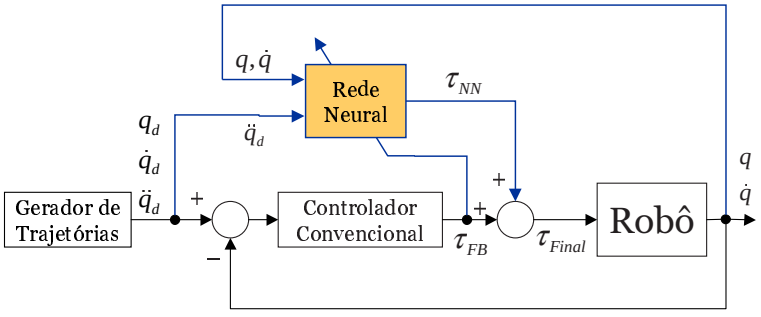


Figura 4.8: Controlador integrado de posição implementado

Para evitar saturação nos atuadores de potência do robô, o torque final, τ_{Final}, enviado para cada uma das juntas, passa por um limitador conforme aparece ilustrado na figura 4.9. Desta forma, caso ocorra uma saturação da rede neural, o controlador convencional pode contrabalançar o torque errôneo gerado pela rede neural. Uma rede neural satura quando sua taxa de aprendizado inicia com um valor muito elevado. Neste caso, o neurônio que teve sua saída saturada, mantém sua saída fixa num dos extremos possíveis de saída (± 1) e neste caso, não há mais possibilidade de recuperação para a rede a não ser reinicializar seus pesos e recomeçar o treinamento com taxas de aprendizado menores.

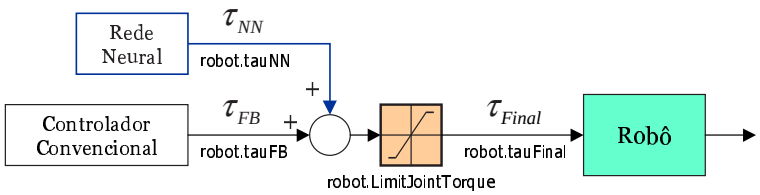


Figura 4.9: Composição dos torques finais enviados às juntas do robô.

Além disto, foi implementado outro mecanismo adicional mais complexo (por *software*), ver figura 4.10, capaz de "desconectar" uma rede neural da malha de controle com base num *flag* (Net.Problems) que confirma que o objeto relacionado com a rede neural foi corretamente inicializado (existe) e que a rede não se encontra com algum neurônio saturado. Neste caso, a rede neural não é mais utilizada na malha e controle. Não é feito o processamento *forward* da rede e nem a retropropagação de erro para ajuste dos pesos da rede (etapa *backward*). Neste caso, apenas o controlador convencional continua responsável pelo controle do processo e o usuário é alertado sobre este problema com uma mensagem de erro na tela de Log150.162.14.59.xx do sistema.

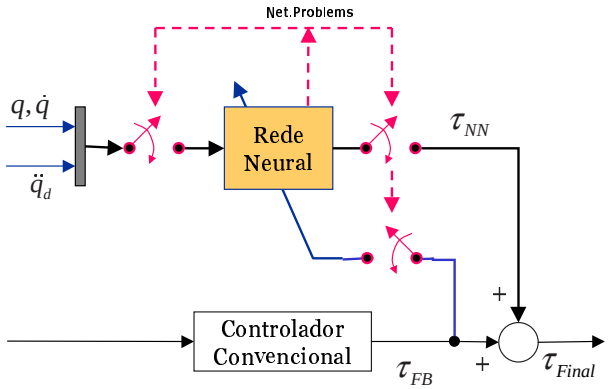


Figura 4.10: Mecanismo de "desconexão" da rede em caso de problemas.

Dependendo do tipo de rede neural adotada, pode ser necessário um escalonamento dos dados de entrada. Redes RBF trabalham diretamente sobre os valores brutos dos dados de entrada fornecidos para a mesma, já que sua camada intermediária, contendo as funções gausssianas vai ser a responsável por "mapear" os dados de entrada. Já a rede MLP exige que os dados de entrada variem na faixa entre -1 à +1. A figura 4.11 mostra o processamento realizado sobre os dados de entrada/saída da rede MLP.

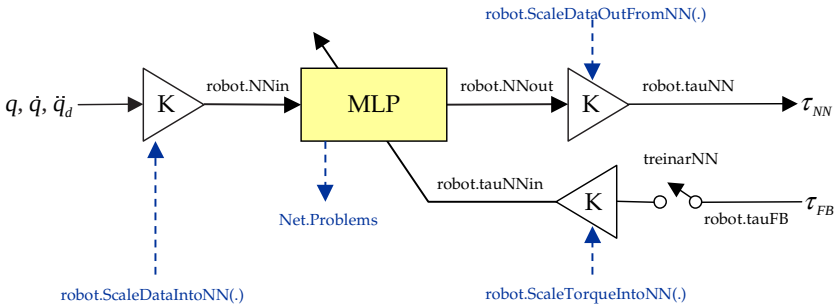


Figura 4.11: Fluxo de dados envolvendo rede MLP para controle de posição.

A figura 4.12 mostra o mesmo fluxo de dados envolvendo uma rede RBF.

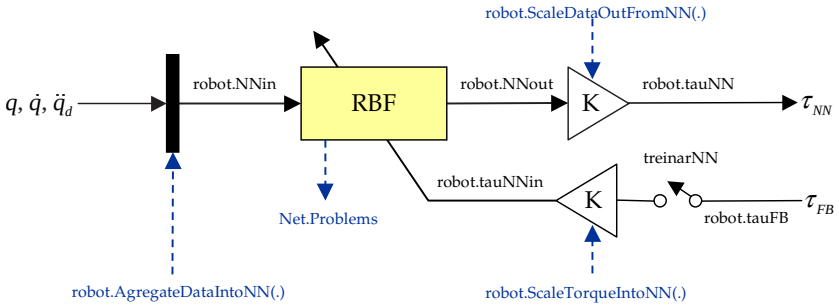


Figura 4.12: Fluxo de dados envolvendo rede RBF para controle de posição.

O método (robot: Robot) ScaleDataIntoNN(·) escalona e organiza os dados de entrada para a rede MLP conforme mostram as equações:

$$x_{NN_i} = q_{Offset_i} + q_{Escala_i}\, q_i \tag{4.13}$$

$$x_{NN_{i+4}} = \dot{q}_{Escala_i}\, \dot{q}_i \tag{4.14}$$

$$x_{NN_{i+8}} = \ddot{q}_{Escala_i}\, \ddot{q}_{d_i} \tag{4.15}$$

onde x_{NN_i} se refere ao i-ésimo elemento de entrada da rede, i se refere ao número da junta ($i = 0:3$) e os termos q_{Offset_i}, q_{Escala_i}, $\dot{q}_{Escala_i}$ e $\ddot{q}_{Escala_i}$ são calculados pelo método (robot:

Robot) InitDataScale da seguinte forma:

$$q_{Offset_i} = \frac{-\text{robot.qmax[i]} - \text{robot.qmin[i]}}{\text{robot.qmax[i]} - \text{robot.qmin[i]}} \tag{4.16}$$

$$\dot{q}_{Escala_i} = \frac{2.0}{1.1\big(\text{robot.qdmax[i]} - \text{robot.qdmin[i]}\big)} \tag{4.17}$$

$$q_{Escala_i} = \frac{2.0}{\text{robot.qmax[i]} - \text{robot.qmin[i]}} \tag{4.18}$$

$$\ddot{q}_{Escala_i} = \frac{2.0}{1.1\big(\text{robot.qddmax[i]} - \text{robot.qddmin[i]}\big)} \tag{4.19}$$

A tabela 4.3 mostra os valores mínimos e máximo relacionados com informações de cada uma das juntas do robô.

Junta	q_{min}	$q_{máx}$	$\lvert\dot{q}_{máx}\rvert$	$\lvert\ddot{q}_{máx}\rvert$	$\lvert\tau_{máx}\rvert$
0	-2.25	2.25	3.00	80.0	333.0
1	-1.90	1.90	3.00	100.0	157.0
2	-0.49	-0.21	0.88	12.0	877.0
3	-2.40	2.40	20.00	500.0	16.7

Tabela 4.3: Valores máximos e mínimos relacionados com cada uma das juntas do robô.

Tanto para a rede RBF quanto para a rede MLP é necessário re-escalonar o torque de saída calculado pela rede:

$$\tau_{Escala_i} = \frac{\text{robot.taumax[i]} - \text{robot.taumin[i]}}{2.0} \tag{4.20}$$

A tabela 4.4 resume os fatores de escala empregados para operar com as redes neurais. As redes começaram a apresentar bom desempenho depois que seus pesos sinápticos w_{ij}, foram inicializados com valores aleatórios na faixa de ± 0.05 e seus pesos de *Bias* foram inicializados aleatóriamente na faixa de ± 0.005. Estes valores podem parecer baixos, mas isto se deve ao fato de que os dados de entrada e saída da rede passam por um "fator de escala". Note que um valor de saída da rede como $y_0 = 0.05$ resulta um torque de saída da rede de: $\tau_{NN_0} = 0.05 \times \tau_{Escala_0} = 16.55$[Nm].

4.5.1 Rede MLP para controle de posição

Foi implementada uma rede do tipo *Perceptrom* Multicamadas (MLP) contendo 12 neurônios na sua camada de entrada (3 vetores de entrada de dados de dimensão 4 ($q, \dot{q}, \ddot{q}_d$) + 1 neurônio de *Bias* = 13), 8 na primeira camada invisível (8 + 1 de *Bias* = 9), 5 (5 + 1 de *Bias*

Junta	$\mathbf{q_{Offset}}$	$\mathbf{q_{Escala}}$	$\dot{\mathbf{q}}_{\mathbf{Escala}}$	$\ddot{\mathbf{q}}_{\mathbf{Escala}}$	$\tau_{\mathbf{Escala}}$
0	0.0000	0.4444	0.3030	0.0114	333.0
1	0.0000	0.5263	0.3030	0.0091	157.0
2	2.5000	7.1429	1.0331	0.0758	877.0
3	0.0000	0.4167	0.0455	0.0018	16.7

Tabela 4.4: Fatores escala empregados com as redes neurais.

= 6) na segunda camada invisível e 4 neurônios de saída necessários para gerar o torque de saída necessário para comandar cada uma das 4 juntas do robô à exemplo do que já havia sido utilizado por Battistela (1999). Esta rede é referenciada no restante do trabalho como "MLP2c"' (rede multicamadas de *perceptrons* com 2 camadas ocultas).

Esta rede composta por $13 \times 9 \times 6 \times 4$ neurônios em cada uma de suas camadas, totaliza um conjunto de: $(13 \times 8 = 104) + (9 \times 5 = 45) + (6 \times 4 = 24) = 173$ conexões sinápticas (pesos) no total. A cada instante de amostragem são realizadas 440 [Flops][4] relativas ao modo de processamento direto (etapa de *forward*) da rede e mais 1092 [Flops] relativos à etapa de aprendizado da rede (etapa de *backpropagation*), o que significa que quando a rede necessita ser treinada, todo o processamento envolvendo a rede MLP exige 1532[Flops]. Ou seja, quando o processamento da rede inclui a etapa de aprendizado são realizados quase 3,5 vezes mais Flops do que simplesmente quando se realiza o processamento direto da rede. Nos cálculos anteriores estão incluídas as entradas de *Bias* e correspondentes pesos sinápticos.

Para a etapa de treinamento desta rede (*backward*) foi adotado o algoritmo de retropropagação de erros ou *back-propagation* expandido com o termo *momentum*. Detalhes da implementação deste algoritmo estão no Apêndice C. Haykin (1999) e Keller (1999) citam algumas vantagens obtidas com a expansão do algoritmo *back-propagation* usando o termo *momentum*:

a) o termo *momentum* permite incrementar a velocidade do aprendizado da rede e ainda evitar que nos casos de uma taxa de aprendizado elevado, a rede entre num estado oscilatório. Taxas de aprendizado elevadas podem até acelerar o aprendizado da rede, mas seu impacto sobre a forma drástica como os pesos sinápticos da rede são atualizados podem tornar a rede instável (oscilatória);

b) a inclusão do termo *momentum* tende a acelerar o algoritmo *back-propagation* para gradi-

[4] *Floating Point Operations* = operações em ponto flutuante.

entes de erro descendentes cada vez menores enquanto acrescenta um efeito estabilizante nos casos em que a derivada do erro muda consecutivamente de sinal entre interações de aprendizado (Haykin, 1999); e;

c) melhor ainda, a inclusão do termo *momentum* no algoritmo *back-propagation* previne a rede de estabilizar seu aprendizado em torno de um mínimo local (indesejável) na superfície de erros.

Também foi previsto o teste com uma rede MLP contendo apenas uma camada invisível com 8 neurônios (rede: "MLP1c"). A idéia era verificar se duas camadas ocultas são importantes para acumular algum aprendizado dinâmico para a rede.

4.5.2 Rede RBF para controle de posição

A rede RBF trabalha com o mesmo vetor de entrada de dados usado para a rede MLP, ou seja, $x_{NN} = [q \quad \dot{q} \quad \ddot{q}_d]$. Assim, a camada de entrada desta rede é formada por $(3 \times 4) = 12$ elementos passivos de entrada. A camada intermediária desta rede é organizada na forma de 3 **classes de dados de entrada**: uma para cada tipo de dado de entrada; especificamente: 1) posições angulares atuais; 2) velocidades angulares atuais e 3) acelerações angulares desejadas.

Organizar os dados de entrada na forma de diferentes *classes de entrada de dados* empregada neste trabalho foi inspirada no trabalho desenvolvido por (Fritzke, 1997; Bouchaffra, 2001; Kiguchi and Fukuda, 1997). Fritzke (1997) mostra que a camada intermediária da rede RBF com suas m_1 funções gaussianas pode ser comparada à etapa de *fuzzyficação* dos dados de entrada usando m_1 funções de pertinência. A camada de saída da rede RBF pode ser comparada às regras processadas pelo motor de inferência de um sistema *fuzzy* de primeira ordem baseado no modelo de Sugeno (Bouchaffra, 2001). Cada peso sináptico pode ser comparado com a parte *IF* da regra relacionada com a saída j do sistema. O produto aritmético relacionado com o cálculo do valor de saída da rede (equação 4.24) corresponde ao processamento *AND* de sistemas *fuzzy*. Apenas no caso de sistemas neuro-*fuzzy* o modo de calcular a saída do sistema pode incluir o uso de uma função de transferência ou uma camada extra responsável pela *defuzyficação* dos valores de saída alcançados pela camada intermendiária da rede RBF (Kiguchi and Fukuda, 1997). A idéia base que inspirou a criação

de diferentes *classes* de entrada de dados foi baseado em trabalhos anteriores já desenvolvidas pelo autor deste trabalho com controladores PID-*fuzzy* (da Mota Almeida et al., 2000).

O espaço de entrada de cada um dos vetores de entrada de dados foi mapeado usando m_1 funções gaussinas de Green (Haykin, 1999), expressa pela equação:

$$\mathbf{G}(\mathbf{x}, \mathbf{t}_i) = \exp\left(-\frac{1}{2 \cdot \sigma_i^2} \cdot \|\mathbf{x} - \mathbf{t}_i\|^2\right) \tag{4.21}$$

onde $\mathbf{G}(\cdot, \cdot)$ representa o vetor de funções gaussianas, $\mathbf{x}$ equivale ao vetor de entrada de dados; $\mathbf{t}_i$ representa o vetor de centro da i-ésima função gaussina; σ_i representa a largura da i-ésima função gaussiana ($\sigma_i > 0$); i varia de 1 até m_1 funções gaussianas desejadas para representar o vetor de entrada $\mathbf{x}$.

Neste trabalho, foi implementada uma classe (P.O.O.[5]) relativa à implementação da rede RBF, permitindo trabalhar com até 9 funções gaussianas na sua camada intermediária ($m_1 = 5 \ldots 9$).

A equação 4.21 é repetida para todas as classes de entrada de dados e aplicada sobre todas as dimensões do vetor de entrada de dados correspondente à classe de entrada de dados atualmente sendo processada. O que no caso de uma rede RBF para controle de posição no espaço de juntas, contendo 5 funções gaussinas ($m_1 = 5$), implica numa camada intermediária contendo: 3 *classes* $\times$ 4 *d.o.f.* $\times$ 5 *funções gaussianas* = 60 neurônios (mais 1 se for considerado o de *Bias* para a camada de saída).

Os centros das funções gaussianas foram fixadas baseadas na quantidade de funções gaussinas desejadas e nos intervalos mínimos e máximos de variação de cada vetor de entrada, o que permite determinar a máxima distância Euclidiana entre os centros escolhidos, d_{max}. As larguras das funções, ou desvios padrões, σ, podem ser fixadas de acordo com a equação (Gabrijel and Dobnikar, 1997; Haykin, 1999):

$$\sigma = \frac{d_{max}}{\sqrt{2 \cdot m_1}} \tag{4.22}$$

onde d_{max} equivale à máxima distância Euclidiana entre cada par de centros consecutivos das funções e que foi calculado como: $d_{max} = (x_{\max} - x_{\min})/(m_1 - 1)$.

[5]P.O.O. = Programação Orientada a Objetos

Os dados de entrada para esta rede foram organizados da seguinte forma:

$$\begin{aligned} x_{NN_i} &= q_i \\ x_{NN_{i+4}} &= \dot{q}_i \\ x_{NN_{i+8}} &= \ddot{q}_{d_i} \end{aligned} \tag{4.23}$$

onde $i = 0 \ldots m_0$, m_0 corresponde à dimensão do vetor de entrada. Neste caso, todos os vetores de entrada têm dimensão igual com valor $m_0 = 4$ correspondendo às 4 juntas do robô.

A figura 4.13 mostra como fica a distribuição das funções de base radial para o caso de $m_1 = 5$ para os dados referentes à junta 2.

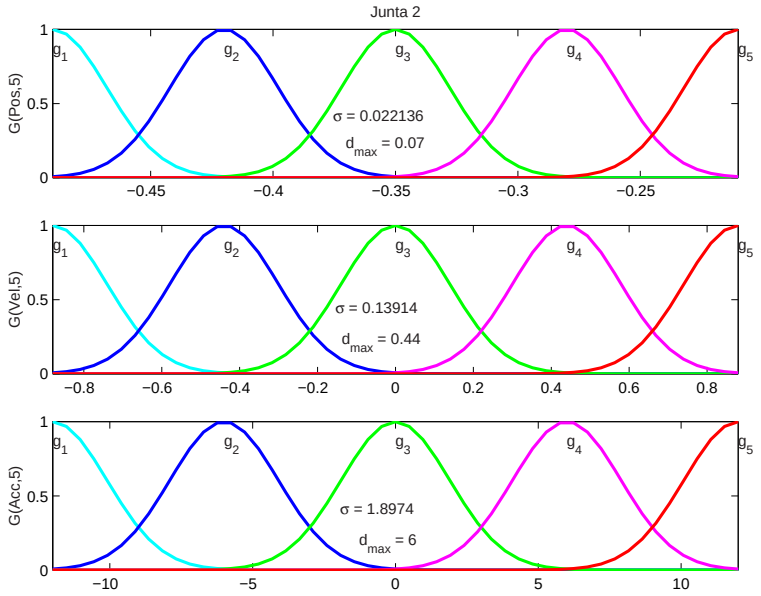


Figura 4.13: Exemplo de distribuição de funções de base radial para dados da junta 2.

A tabela 4.5 mostra como fica organizado o vetor $\mathbf{t_i}$ quando $m_1 = 5$ (usamos 5 funções gaussianas para representar os dados de entrada). Mostra também como ficaram os valores calculados para o vetor d_{max} e para o vetor de desvio padrão (ou, larguras), σ, para as m_1 funções gaussianas utilizadas para caracterizar cada vetor de entrada de dados $\mathbf{x}$.

Por fim, a camada de saída da rede RBF é formada por 4 elementos, cada um para gerar o torque para cada junta do robô, cujas saídas são calculadas como:

$$y_j = \sum_{i=1}^{m} w_{ij}\, \varphi_i \tag{4.24}$$

Classe 0 → q (Posições Angulares Instantâneas):								
Junta	Funções Gaussianas						$d_{máx}$	σ
0	$t_i = [$	-2.25	-1.12	0.00	1.12	2.25	]	1.125	0.356
1	$t_i = [$	-1.90	-0.95	0.00	0.95	1.90	]	0.950	0.300
2	$t_i = [$	0.49	-0.42	-0.35	-0.28	-0.21	]	0.070	0.022
3	$t_i = [$	-2.40	-1.20	0.00	1.20	2.40	]	1.200	0.379
Classe 1 → $\dot{q}$ (Velocidade Angulares Instantâneas):								
Junta	Funções Gaussianas						$d_{máx}$	σ
0	$t_i = [$	-3.00	-1.50	0.00	1.50	3.00	]	1.500	0.474
1	$t_i = [$	-3.00	-1.50	0.00	1.50	3.00	]	1.500	0.474
2	$t_i = [$	-0.88	-0.44	0.00	0.44	0.88	]	0.440	0.139
3	$t_i = [$	-20.00	-10.00	0.00	10.00	20.00	]	10.000	3.162
Classe 2 → $\ddot{q}_d$ (Acelerações Angulares Desejadas):								
Junta	Funções Gaussianas						$d_{máx}$	σ
0	$t_i = [$	-80.00	-40.00	0.00	40.00	80.00	]	40.000	12.649
1	$t_i = [$	-100.00	-50.00	0.00	50.00	100.00	]	50.000	15.811
2	$t_i = [$	-12.00	-6.00	0.00	6.00	12.00	]	6.000	1.897
3	$t_i = [$	-500.00	-250.00	0.00	250.00	500.00	]	250.000	79.057

Tabela 4.5: Funções gaussianas usadas para mapear os vetores de entrada da rede RBF usando 5 funções.

onde $j = 1 \ldots 4$ corresponde ao sinal de saída da rede à ser transformado (depois de escalonado) no torque para a junta j; m corresponde ao número de neurônios presentes na camada intermediária da rede (a camada que contem as funções gaussianas); e φ_i corresponde à saída das funções gaussianas presentes na camada intermediária da rede e é determinado da seguinte forma:

$i = -1$
$k = -1$
for $c = 0$ **to** $classes$-1 **do**
  for $j = 0$ **to** $Elementos\{classe_c\} - 1$ **do**
    $k = k + 1$
    for $m = 0$ **to** $m_1 - 1$ **do**
      $i = i + 1$
      $\varphi_i = G(x_k, t_{cjm})$
    end for
  end for
end for

onde: c varia o número da classe de entrada de dados; $Elementos\{classe_c\}$ se refere à quantidade de elementos, ou dimensão do vetor de entrada atual; k varia o número do elemento de entrada passivo da rede RBF; m varia o número da função gaussiana atualmente sendo processada sobre a variável x_k de entrada de dados. A abordagem acima é útil e propositalmente genérica para aplicação posterior da rede RBF para controle de força onde os vetores

de entrada de dados tem suas dimensões variando de 3 à 6. No caso da rede RBF com 5 funções gaussianas, i varia de: $i = 0 \ldots 59$. Isto significa que cada um dos neurônios da camada de saída de rede possui conexão com todos os neurônios da camada anterior, a intermediária, sem considerar a classe de dados do vetor de entrada ou a qual junta ele se refere.

Um exemplo de rede RBF implementada aparece na figura 4.14. Ela utiliza 5 funções gaussianas para mapear os dados de entrada, totaliza na sua camada oculta: 3×4 (graus de liberdade) $\times$ 5 (funções gaussianas) $+1$ (neurônio de *Bias*) $= 61$ neurônios, composta por $61 \times 4 = 244$ conexões sinápticas (pesos) no total.

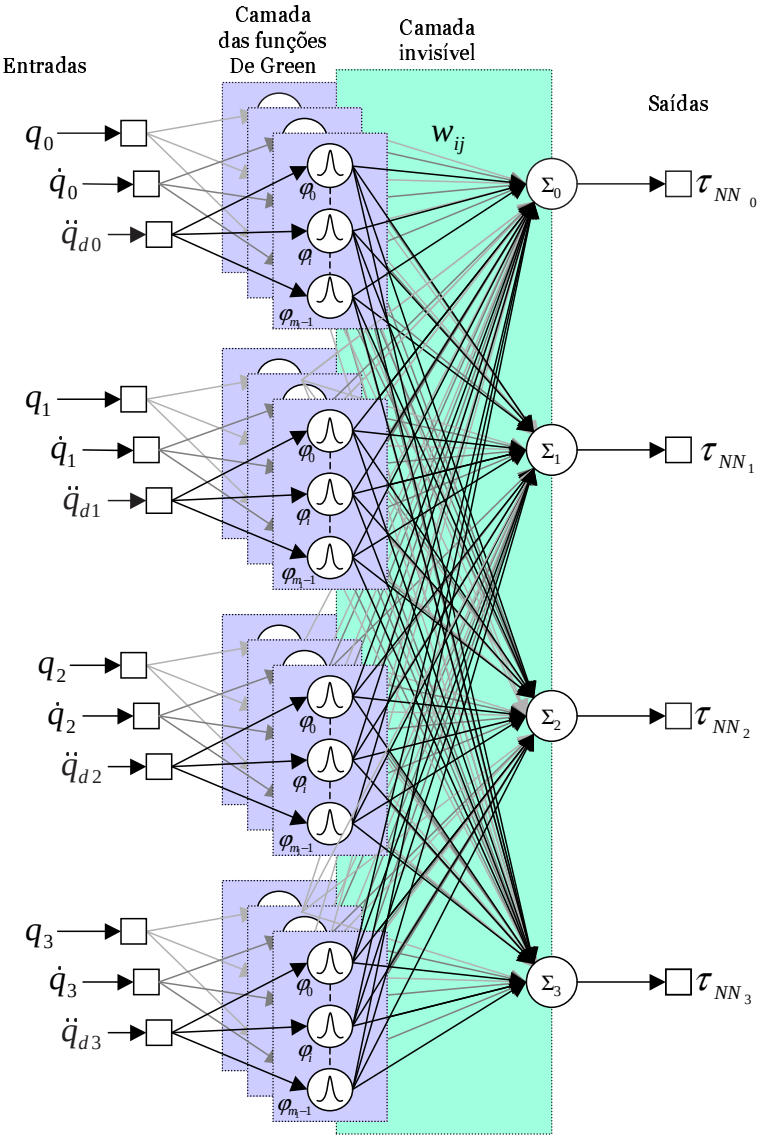


Figura 4.14: Estrutura de rede neural RBF para controle de posição ($m_1 = 5$ neste caso).

As redes RBF utilizadas neste trabalho permitem ajuste *on-line* apenas dos pesos sinápticos da camada de saída da rede, baseado no mesmo algoritmo do gradiente descendente ou *back-propagation* já utilizado para treinar as redes MLP. A rede RBF implementada depois de inicializada não modifica mais a posição de seus centros nem a largura de suas funções. Foi implementada desta forma para evitar o acúmulo de no mínimo mais 2 parâmetros de ajuste para operação desta rede, relacionados com taxas de aprendizado para os centros e largura de cada uma das funções gaussianas utilizadas (além é claro de implicar em mais tempo de processamento necessário para determinação destes ajustes). Se for desejado ajuste para as funções gaussianas da rede ver (Gabrijel and Dobnikar, 1997).

4.6 Controle Híbrido de Posição/Força

Sempre que se realiza controle de força num robô manipulador, ele é realizado no mínimo em um dos graus de liberdade disponíveis do robô; nos outros graus de liberdade continua sendo realizado controle de posição. Este é o conceito básico da técnica de controle de força/posição conhecida como "controle híbrido" que divide os graus de liberdade de um robô em dois subespaços ortoganais entre si: subespaço de controle de posição e subespaço de controle de força. A divisão destes espaços é realizada conforme especificado por uma matriz conhecida como matriz de Seleção S. Esta técnica não especifica o tipo particular de controlador de posição ou de força que deve ser utilizado (Raibert and Craig, 1981; Zeng and Hemani, 1997; McCarragher et al., 1997; Sciavicco and Siciliano, 1996; Vukobratovič and Stojić, 1995; Volpe, 1990).

Neste trabalho optou-se por realizar controle de força apenas na direção Z e controle de posição das direções restantes: X, Y e orientação do efetuador final do robô (ângulo θ). Sendo assim, a matriz de seleção S fica definida como:

$$S = \begin{bmatrix} \mathbf{0} & 0 & 0 & 0 \\ 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{0} \end{bmatrix}$$

onde os elementos da diagonal principal nulos indicam a direção onde será realizada con-

trole de posição; os elementos da diagonal principal iguais à 1, indicam controle de posição naquela direção. Assim é estabelecida uma malha de realimentação para controle de posição nas direções ortogonais às quais é realizada a realimentação para controle de força, conforme mostra a figura 4.15.

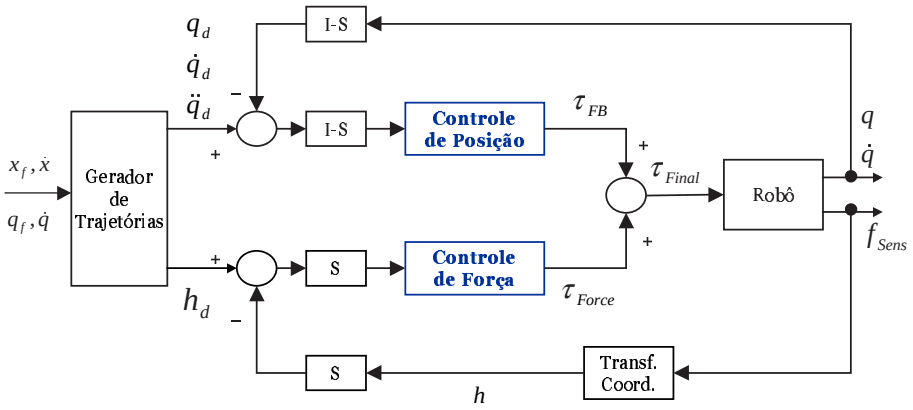


Figura 4.15: Controle híbrido implementado no XO/2.

Para a malha de controle de posição estão previstos o uso dos controladores de posição já explorados anterioremente e o uso do controlador integrado de posição na malha de controle de posição conforme já mostrado na figura 4.8. Para a malha de controle de força está previsto o uso dos controladores convencionais de força Proporcional e PI já demonstrados anteriormente.

4.7 Controladores Convencionais de Força

Os controladores implementados aqui realizam controle de força explícito já que fazem uso dos dados captados à partir do sensor de força JR3 disponível no robô Inter/Scara, a princípio da forma indica no diagrama de blocos mostrado na figura 4.16.

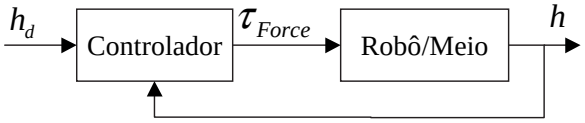


Figura 4.16: Diagrama em blocos de controle de força explícito

Os controladores convencionais previstos para trabalharem nesta malha são o Proporcional e o PI. O controlador com ação derivativa foi abandonado em função do sinal carregado de ruídos proveniente do sensor de força.

Todos os controladores de força foram implementados para realizar controle de força na direção Z apenas, ou seja, o robô faria contato contra uma superfície apenas na direção Z do seu espaço operacional. O robô ainda pode se mover nas outras direções: X e Y e modificar a orientação do seu efetuador final (ângulo θ).

A referência para este controlador ou força de contato desejada contra o meio é especificada no espaço operacional (de coordenadas de base do robô). O torque gerado pelo controlador de força pode então ser passado diretamente para a junta 2 pois esta é a única junta relacionada com os movimentos na direção Z.

Neste trabalho não foi dedicada atenção especial para o **controle do impacto**. Apenas foi implementada uma **técnica para detectar o contato**, ativando ou desativando o controlador de força no caso de ser detectada uma força de contato contra o sensor de força, na direção Z, superior à: f_{THR} variando entre 0.5 à 1.0 [N]. Quando $f_{Sens_z} < -f_{THR}$, o controle de força é realizado, senão, é chaveado o controlador de posição. Isto faz com que seja mantido algum controle sobre as juntas do robô mesmo no caso de perda de contato do robô contra o meio. Naturalmente que esta não é a melhor técnica para evitar perda de contato ou aproximação suave contra a peça.

A tarefa relacionada com as funções de segurança do robô (PROCEDURE (h: ControlSecurityHdl) Run()) foi ampliada para considerar ou não erros máximos de seguimento de trajetória (atributo: robot.checkTrackError[i]:= FALSE). O usuário pode ainda modificar o valor limite destes erros máximos de seguimento de trajetória (por exemplo: XOClient.Execute Main.**SetErroMax** 5.0 5.0 20.0 5.0 $\sim$). O objetivo neste caso é possibilitar o teste dos algoritmos de controle de força já que originalmente a tarefa relacionada com as funções de segurança do robô continuamente verificava os erros atuais de seguimento de trajetória. Como no caso de controle de força não é controlada a posição na direção em que é realizado o controle de força, sem esta estratégia seria impossível realizar controle de força sem que esta função de segurança do robô fosse ativada (o robô paralisa suas operações neste caso).

A tarefa de segurança foi ampliada também para depois de inicializado o robô, verificar continuamente os valores captados pelo sensor de força do robô a fim de **detectar casos de colisão**. No caso do sensor de força detectar forças superiores à 80[N] (correspondendo à 80% do valor máximo permitido para o sensor de força nas direções X e Y) ou momentos superiores à 5.1 [Nm] (também 80% do valor máximo suportado pelo sensor de

força), as tarefas de tempo reais relacionadas com o Controlador principal são paralisadas (robot.controlFBTask.Stop), é enviado torque nulo para as juntas do robô e os freios são liberados para possibilitar uma reação "passiva" do robô para o caso da colisão (lei da força e reação). O objetivo aqui é preservar o sensor de força contra problemas verificados com alguma operação incorreta com o robô principalmente no caso de controle de força.

4.7.1 Controlador Proporcional de Força

É o controlador mais simples de ser realizado para a malha de realimentação relacionada com o controle de força. Trata-se de computar somente:

$$\tau_{Force} = K_p \left(h - h_d \right) \tag{4.25}$$

4.7.2 Controlador PI de Força

Ao controlador anterior é simplesmente adicionada ação integral sobre o sinal de erro de força. A lei de controle está baseada em:

$$\tau_{Force} = K_p \, \tilde{h} + K_i \int \tilde{h} \, dt \tag{4.26}$$

onde $\tilde{h} = h - h_d$. Esta lei de controle foi implementada na forma digital como:

$$\tau_{Force}[k] = K_p \tilde{h}[k] + K_i \sum_{k=0}^{\infty} \tilde{h}[k] T \tag{4.27}$$

onde: T se refere ao período de amostragem adotado para a malha de controle de força.

4.8 Controladores Integrados de Força

A primeira arquitetura de rede neural prevista para atuar na malha de controle de força, **RNf-1**, usa como dados de entrada o vetor de entrada $x_{NN} = [\, q \quad f_{Sens} \quad h_d \,]$. Como esta rede acabou demonstrando um desempenho muito aquém do esperado, outras 3 diferentes estruturas de entrada de dados foram avaliadas, à saber:

1. **RNf-2**: trabalha apenas com h_z e $\tilde{h}_z$;

2. **RNf-3**: trabalha com informações atrasadas de h_z + h_{d_z};

   e,

3. **RNf-4**: trabalha com h_z, $\tilde{h}_z$ e $\dot{q}_2$;

Os itens à seguir informam os detalhes relacionados com cada um dos tipo de rede implementados e testados.

4.8.1 Primeira Versão da RN para Controle de Força (RNf-1)

A primeira versão prevista para a rede neural para a malha de controle de força trabalha com os seguintes dados de entrada: $x_{NN} = [\,q \quad f_{Sens} \quad h_d\,]$. A dimensão de cada um dos vetores de entrada de dados é mostrada na figura 4.17.

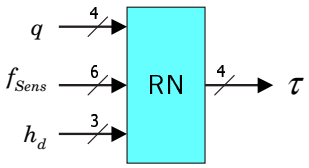


Figura 4.17: Dados de entrada e de saída da rede RNf-1.

Se esperava que através dos dados de entrada: q e f_{Sens}, a rede fosse capaz de inferir internamente, informação capturada do sensor de força, do espaço-sensor para o espaço-operacional ($h = \mathcal{N}(f_{Sens},\, q)$).

4.8.2 Segunda Versão da RN para Controle de Força (RNf-2)

Como a primeira versão não funcionou a contento, foi imaginado se uma abordagem mais simples para os dados de entrada da rede permitira que a rede realizasse a compensação dinâmica na malha de controle de força. Esta rede era composta de apenas 2 neurônios de entrada, tanto para rede MLP quanto para a rede RBF. A rede MLP testada continha 3 neurônios na sua única camada oculta. Ambos modelos de redes geravam apenas uma variável de saída que era o torque à ser enviado para a junta 2 (a prismática) responsável por realizar controle de força na direção Z apenas. A figura 4.18 mostra os dados de entrada e de saída desta rede.

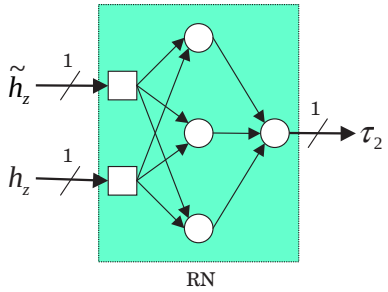


Figura 4.18: Dados de entrada e de saída para RNf-2.

4.8.3 Terceira Versão da RN para Controle de Força (RNf-3)

Como também a segunda versão imaginada para a entrada de dados da rede não se mostrou eficaz, foi testada uma nova proposta inspirada na modelagem ARMA (*AutoRegressive Moving Average*) adotada por outros controladores neurais, já revisada co Capítulo 3, seção 3.3.6. Esta abordagem consiste em passar informações atrasadas da entrada e da saída do processo para a rede, de forma que a mesma tente identificar o sistema à ser controlado.

Nesta nova abordagem foram passadas 5 amostras atrasadas da informação de força na direção Z já traduzida para o espaço operacional, mais a informação atual da força executada na direção Z (também já no espaço operacional) e mais a força desejada na direção Z. Acabando por compor uma rede composta por 7 neurônios na sua entrada e apenas 1 neurônio na sua camada de saída, já que mais uma vez esta rede foi projeta para auxiliar na malha de controle de força atuando apenas na direção Z (o torque gerado por esta rede é adicionado ao torque gerado pelo controlador convencional de força para a direção Z e finalmente encaminhado para a junta 2 do robô, a prismática). A figura 4.19 mostra o fluxo de dados envolvidos nesta versão da rede para controle de força. Esta versão foi testada apenas com uma rede MLP contendo 3 neurônios na sua única camada oculta.

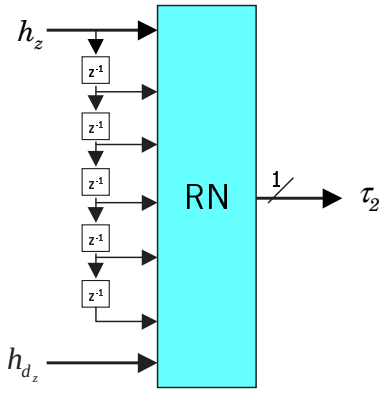


Figura 4.19: Dados de entrada e de saída para RNf-3.

4.8.4 Quarta Versão da RN para Controle de Força (RNf-4)

Esta foi a última tentativa de gerar um ”pacote de entrada de dados” eficaz para a rede neural responsável pela malha controle de força. Inspirado no fato de que certos controladores de força também utilizam como informação de entrada para sua lei de controle de força, a ”velocidade” com que o robô entra no meio, esta informação também foi passada para a rede (mas apenas velocidade no espaço de juntas da junta 2) juntamente com a informação atual de força sendo gerada na direção Z (já traduzida do espaço-sensor para o espaço operacional) e informação da força desejada para contato na direção Z (fornecida no espaço operacional). Esta abordagem foi testada apenas com a rede MLP contendo 1 camada oculta apenas com 5 neurônios e apenas 1 neurônio na sua camada de saída, responsável por gerar o torque para a junta 2. A figura 4.20 mostra o fluxo de dados gerado por esta rede.

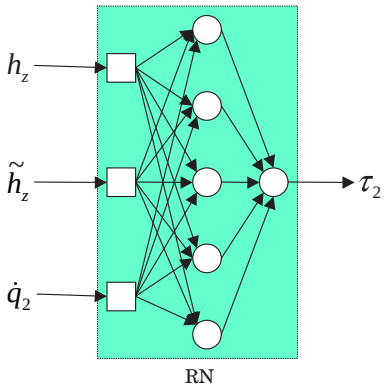


Figura 4.20: Dados de entrada e de saída para RNf-4.

4.9 Índices de Desempenho

Neste trabalho foram adotados diferentes critérios (índices de desempenho) de forma a tentar avaliar de maneira mais objetiva o desempenho obtido com os diferentes controladores. Estes critérios são avaliados por sobre o erro de seguimento de trajetória e por sobre o esforço de controle.

4.9.1 Índices em relação aos Erros de seguimento de trajetória

R1: Erro de Posicionamento final: $\tilde{q}_f$ ou $\tilde{x}_f$:

$$\tilde{q}_f = \tilde{q}_{t\to\infty}$$

$$\tilde{x}_f = \tilde{x}_{t\to\infty}$$

este critério informa apenas o erro de posição em regime permanente.

R2: Erro máximo: $MÁX\{\tilde{q}\}$ ou $MÁX\{\tilde{x}\}$:

$$MÁX\{\tilde{q}\} = \max\{\tilde{q}[kini : kfim]\}$$

$$MÁX\{\tilde{q}\} = \max\{\tilde{x}[kini : kfim]\}$$

Retorna o maior erro de seguimento de trajetória verificado.

R3: Erro Integral Absoluto: $IAE\{\tilde{q}\}$ ou $IAE\{\tilde{x}\}$:

$$IAE\{\tilde{q}\} = \sum_{i=kini}^{kfim} \left|q[k] - q_d[k]\right|$$

$$IAE\{\tilde{x}\} = \sum_{i=kini}^{kfim} \left|x[k] - x_d[k]\right|$$

Acumula de modo absoluto os erros de seguimento de trajetória. Quanto menor este índice, melhor o seguimento de trajetória para o controlador em questão sendo avaliado.

R4: Erro Médio Absoluto: $MED\{\tilde{q}\}$:

$$MED\{\tilde{q}\} = \bar{\tilde{q}} = \frac{\sum_{k=kini}^{kfim} \big(q[k] - q_d[k]\big)}{N}$$

Este valor tenta estimar o valor médio do erro de seguimento de trajetória. Faz mais sentido quanto menor a *variância* do erro de seguimento de trajetória (ver a seguir).

R5: Erro Quadrático Médio: $MSE\{\tilde{q}\}$:

$$MSE\{\tilde{q}\} = \frac{\sum_{k=kini}^{kfim} \Big(q[k] - q_d[k]\Big)^2}{N}$$

Soma acumulada do quadrado dos erros de seguimento de trajetória. Maximiza a soma acumulada de erros de seguimento de trajetória.

R6: Variância do Erro: $VAR\{\tilde{q}\}$:

$$VAR\{\tilde{q}\} = s^2\{\tilde{q}\} = \frac{\sum_{k=kini}^{kfim} \Big(\tilde{q}[k] - \bar{\tilde{q}}\Big)^2}{N-1}$$

Este índice objetiva estimar a dispersão do erro de seguimento de trajetória em torno do valor médio de erro encontrado para execução da trajetória. Quanto maior este índice, significa que o controlador fez a saída daquela junta variar muito em torno do valor da referência desejado a cada instante de tempo. A raiz quadrada deste valor, $s = \sqrt{s^2\{\tilde{q}\}}$, ou desvio padrão, permite estimar que 95% dos valores de erro de seguimento de trajetória se encontram na faixa entre $\bar{\tilde{q}} - 2s$ e $\bar{\tilde{q}} + 2s$. Quanto menor este índice, melhor a capacidade do controlador sendo avaliado em manter aquela junta mais próxima do valor desejado de posição à cada instante de tempo.

4.9.2 Índices de Desempenho em relação ao Esforço de Controle realizado

R7: Maior valor de torque enviado às juntas: $M\acute{A}X\{\tau\}$:

$$M\acute{A}X\{\tau\} = \max\{\tau[kini : kfim]\}$$

Maior ação de controle enviada a certa junta do robô. Permite identificar casos de sobrecarga na ação de controle, se bem que neste caso, por questões de segurança, foi implementado por software um "saturador" para a ação de controle a fim de evitar a atuação das proteções eletrônicas existentes no robô Inter/Scara – o que na prática exige que todo o gabinete do robô (CPU) seja reinicializado depois que uma certa pausa foi respeitada a fim de descarregar capacitores internos relacionados com os drivers de potência dos motores de cada uma das juntas do robô.

R8: Média do valor de torque enviado às juntas: $MED\{\tau\}$:

$$MED\{\tau\} = \bar{\tau} = \frac{\sum_{k=kini}^{kfim} \tau[k]}{N}$$

Valor médio da ação de controle enviado à certa junta. Este índice existe exclusivamente em caráter informativo ou de médio interesse uma vez que pode perder seu valor em função de que na área de robôs manipuladores não se trabalha com entradas degraus e o sinal referência de entrada pode variar tanto de maneira positiva quanto negativa durante o intervalo do teste – se reportar a discussão realizada no item à seguir.

R9: Variação média do torque enviado às juntas: $VARi\{\tau\}$:

$$VARi\{\tau\} = \frac{\sum_{k=kini+1}^{kfim} \Big|\tau[k] - \tau[k-1]\Big|}{N-1} \tag{4.28}$$

Uma tentativa para estimar a variação contida na ação de controle. Quanto menor este índice, menos oscilatório deveria ser a ação de controle e portanto melhor o controlador. Este índice tenta estimar a variação da ação de controle entre dados consecutivos da ação de controle. Note que se este índice fosse medido usando o termo *variância*, o que seria estimado é a variabilidade da ação de controle relativa a um ponto de referência, neste caso, o valor médio da ação de controle, um valor sem sentido principalmente porque no caso de robôs manipuladores um comando de movimentação aplicado para certa junta pode implicar numa ação inicial de controle positiva e logo após, uma ação de controle "negativa". O valor médio poderia então chegar a ser nulo ou ligeiramente maior que zero o que não reflete de maneira nenhuma o esforço de controle realizado. É

que no caso de robôs manipuladores não se costuma introduzir entradas degrau ao sistema. Aqui, a referência varia continuamente no tempo e na maior parte das vezes, de forma contínua e suave no tempo. Principalmente nos casos de comandos de operação do robô especificados no espaço operacional pode-se perceber muitas mudanças abruptas da referência de posição para cada uma das juntas (a ação de controle é executada sobre o "espaço de juntas"), ou seja, neste caso, o valor médio da ação de controle perde sentido. Outros autores como Henriques et al. (1998) utilizaram uma pequena variação sobre a forma como determinar um índice para medir a variação da ação de controle, neste caso, fazendo:

$$SSCC = \sum_{k=1}^{N} (u[k] - u[k-1])^2$$

o fator "^{2}" naturalmente que ressalta as variações na ação de controle entre instantes consecutivos de amostragem.

Obs: $kini$ e $kfim$ se referem aos pontos inicial e final da trajetória atualmente sendo avaliada;

$N = kfim - kini + 1$, se refere ao número de pontos existente dentro da amostra de dados sendo avaliada.

Os resultados obtidos com todos os controladores aqui descritos são apresentados e discutidos nos Capítulos 5 e 6 à seguir.

Capítulo 5

Resultados dos Controladores de Posição

5.1 Introdução

Este capítulo apresenta os resultados obtidos com as implementações realizadas tanto para os controladores convencionais de posição quanto para os controladores integrados de posição (que mesclam um controlador convencional com alguma rede neural). Foram realizados ensaios de movimentações simples e ensaios com uma perturbação presente no meio da execução da trajetória. Todos os ensaios realizados terminam com uma tabela comparando os resultados obtidos conforme os índices de desempenho selecionados neste trabalho e também com um item correspondente à conclusões preliminares onde se apresenta uma rápida discussão dos resultados obtidos com cada tipo de ensaio (sem perturbação e com perturbação). Ao final, o capítulo termina com uma discussão final sobre os resultados obtidos com a aplicação de redes neurais na arquitetura proposta para controle de posição.

5.2 Ajuste do PID

Ao princípio dos testes práticos foi realizada uma nova tentativa para sintonização dos controladores convencionais de posição previstos para uso com as redes neurais. Foi utilizado o método de Ziegler-Nichols para sintonia deos controladores PID (detalhes sobre o ajuste destes controlares por Ziegler-Nichols se encontram no Apêndice B). Os ensaios para

sintonia dos PIDs para cada uma das juntas foram realizados todos no espaço de juntas, uma junta de cada vez.

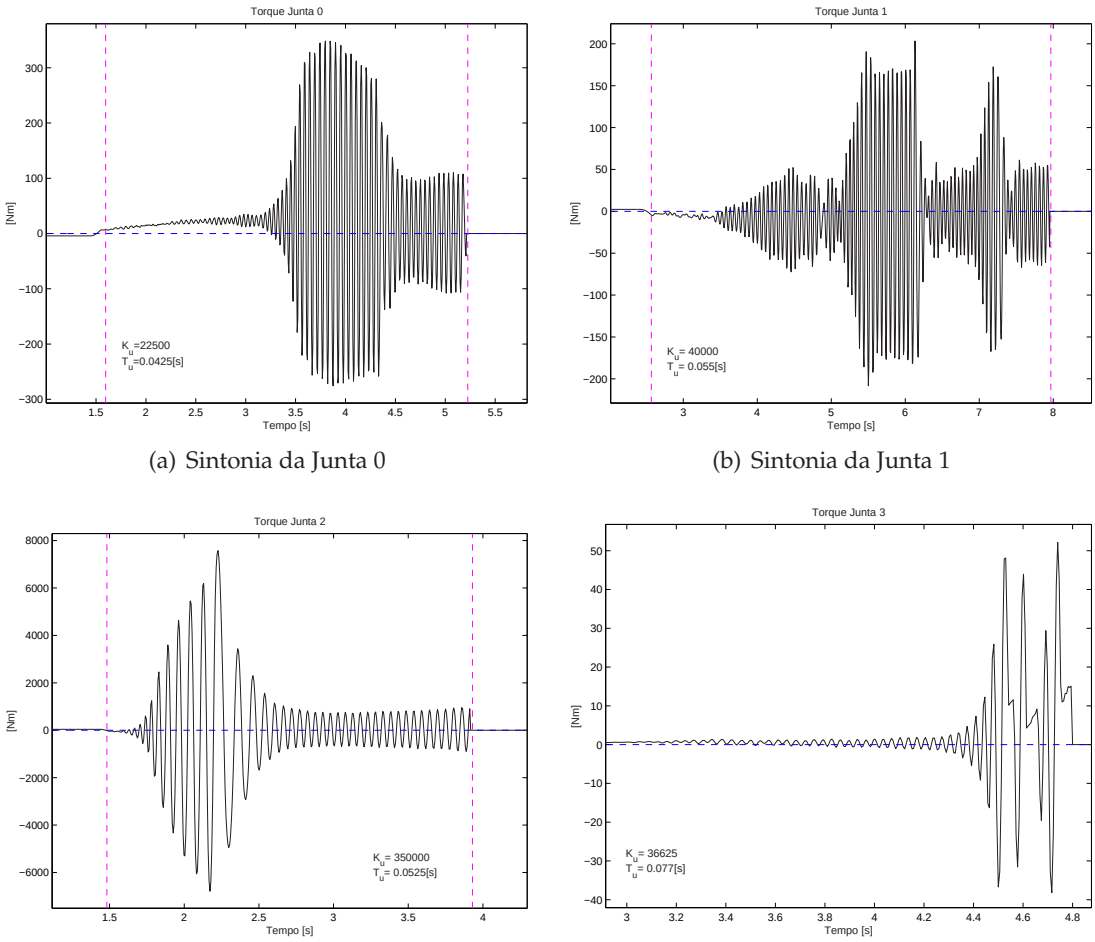
A figura 5.1 mostra os ganhos máximos encontrados (K_u=*ultimate gain*) para cada uma das juntas do robô e o período em que ocorreu a oscilação neste caso (T_u=período da oscilação), parâmetros necessários para realizar a sintonia dos PID's conforme o método sugerido por Ziegler-Nichols (item B.2 do Apêndice B). Estes valores serviram para definir os ganhos à *priori* dos controladores PID à serem adotados. Por fim, foi constatado que estes ganhos variam conforme a região de operação do robô e por ter sido constatado a presença de oscilações (malha fechada com subamortecimento) em certas trajetórias optou-se por adotar para os ganhos K_p e K_d, os mesmos valores usados originalmente pelo controlador PD que veio de "fábrica". Mas para os ganhos K_i (ação integral) foram adotados os valores encontrados através dos ensaios de Ziegler-Nichols para ajuste dos PIDs. A tabela 5.1 mostra na última linha os ganhos que foram adotados para operação com controlador PID no espaço de juntas.

	Parâmetros				
Controlador		Junta 0	Junta 1	Junta 2	Junta 3
Ajuste do PID	K_u	22500	40000	350000	21975
(ganhos máx.)	T_u	0.0425	0.055	0.0525	0.077
PID (ganhos)	K_p	13500	24000	210000	21975
por	K_d	120	300	2300	350
Ziegler-Nichols	K_i	478	1200	9200	1410
PD (ganhos)	K_p	4900	12100	90000	14400
de "fábrica"	K_d	140	220	600	240
	K_p	4900	12100	90000	14400
PID adotado	K_d	120	300	2300	350
(ganhos)	K_i	478	1200	9200	1410

Tabela 5.1: Ajuste dos controladores convencionais de posição (espaço de juntas) usados neste trabalho.

5.3 Ajuste das Redes Neurais

Antes de iniciar uma avaliação do desempenho dos controladores integrados se fez necessário encontrar os parâmetros mais adequados para o aprendizado das redes neurais utilizadas.



(a) Sintonia da Junta 0

(b) Sintonia da Junta 1

(c) Sintonia da Junta 2

(d) Sintonia da Junta 3

Figura 5.1: Sintonia dos controladores PID para cada uma das juntas do robô.

5.3.1 Parâmetros de aprendizado para redes MLP

Para as redes MLP foram testadas taxas de aprendizado, η=0.001 até η=0.04. O valor que permitiu resultados com menor variação dos erros ($VAR\{\tilde{q}\}$) foi com η=0.035. Também foi verificado o desempenho da rede frente ao termo *momentum* (α). Foi verificado que para valores abaixo de $\alpha < 0.3$ a rede parece "esquecer" aprendizado relevante realizado nos instantes anteriores de amostragem porque aumenta o erro de seguimento de trajetória e a variação dos erros. Entretanto valores superiores a $\alpha > 0.7$ praticamente "congelam" a capacidade de aprendizado da rede, fazendo cair o desempenho da rede (a rede se torna "lenta"). Valores elevados para o termo *momentum* fazem com que a rede leve em maior consideração o aprendizado já realizado nas interações anteriores, tornando a rede "cega" para a necessidade de novos ajustes, principalmente no caso de mudança na configuração

do robô (robô partindo para execução de novo movimento). O valor que trouxe um melhor compromisso estabilidade $\times$ capacidade de aprendizado foi de α=0.5.

5.3.2 Parâmetros de aprendizado para redes RBF

A rede RBF adotada utiliza o algoritmo de *back-propagation* para ajustar os pesos da sua camada de saída. Taxas de aprendizado, $\eta < 0.0002$ praticamente cancelam o efeito preponderante da rede sobre o torque final enviado às juntas do robô no controlador integrado. O aprendizado da rede se torna muito lento assim com seu efeito sobre o torque final que é enviado às juntas do robô. Foram testados valores ligeiramente maiores que $\eta = 0.0005$ que apesar de incrementar o desempenho da rede, levam a respostas mais oscilatórias. O valor considerado mais adequado para a taxa de aprendizado para as redes RBF foi $\eta = 0.0005$. Mesmo com este valor sendo inferior à taxa praticada para as redes RBF, pôde ser percebido que a rede RBF reage mais prontamente que uma rede MLP. Comportamento completamente dentro do esperado dada a característica de mapeamento direto das variáveis de entrada realizadas pelas funções gaussinas da sua camada intermediária sem nenhuma necessidade de ajuste. Note que neste trabalho optou-se por trabalhar com centros e larguras fixas, definidas à priori para as funções gaussianas. Para o termo *momentum* foi definido o valor α=0.5 com sendo o que permitiu uma melhor relação entre velocidade de resposta da rede e capacidade adequada de adaptação, num comportamento semelhante ao presenciado com as redes MLP.

5.4 Ensaios Propostos

Dois tipos principais de testes foram realizados com as RNs para controle de posição, de forma a avaliar seu desempenho: 1) testes simples de movimentação e, 2) teste para verificar a capacidade de rejeição à perturbações.

Para o primeiro tipo de teste foram previstos duas trajetórias diferentes visando explorar diferentes regiões do espaço de trabalho do robô. Estas duas trajetórias se referem a movimentos simples especificados no espaço de juntas sem nenhuma perturbação presente durante seu percurso. O objetivo inicial aqui é simplesmente avaliar os primeiros resultados obtidos com a inclusão das redes neurais no controle de posição integrado e comparar os

resultados obtidos com o dos controladores convencionais.

A seguir, os mesmos testes são repetidos para uma trajetória incluindo uma perturbação no decorrer do percurso. A trajetória normal em XY foi interceptada por uma cinta elástica. O objetivo agora foi avaliar a capacidade de rejeição à perturbação dos controladores tanto convencionais quanto dos integrados sendo propostos, como também avaliar algum **efeito "residual" de memória** no caso dos testes com os controladores integrados neurais. Foi verificado o que acontece a primeira vez que controlador integrado se depara com a perturbação assim como se melhora seu desempenho se o movimento é repetido algumas vezes com a perturbação presente. A seguir, a perturbação é retirada e é verificado se a primeira vez que a rede repete a trajetória sem a perturbação, permanece algum efeito "residual" no aprendizado realizado pela rede e como evolui (ou não) o desempenho da rede no caso de se repetir esta trajetória mais vezes sem a perturbação presente. O objetivo deste teste é avaliar a forma como as redes lidam com perturbações.

Os três tipos de ensaios realizados terminam cada um com suas próprias conclusões preliminares e este capítulo termina com o item 5.7 (pág. 210), "Discussão final", que traz um resumo das conclusões obtidas à partir dos ensaios realizados com os controladores integrados de posição.

Todos os controladores de posição avaliados neste capítulo trabalharam com a mesma freqüência original de amostragem adotada pelo controlador PD de "fábrica" que acompanhou o robô Inter, ou seja, f_s= 1 [KHz], ou T_s=1 [ms].

5.5 Ensaios com Trajetórias Simples (sem perturbação)

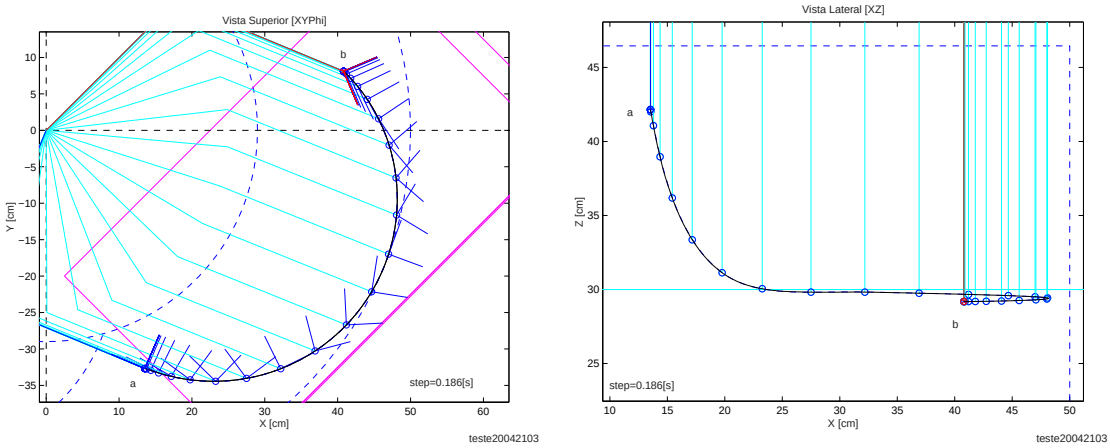
Foram previstos mais de um tipo de trajetória visando verificar o desempenho alcançado pelos controladores em diferentes regiões do espaço de trabalho do robô: Trajetória "A" e Trajetória "B" que operam sobre diferentes regiões do espaço de trabalho do robô.

5.5.1 Testes com a Trajetória "A"

Dados da Trajetória "A"

O perfil da trajetória "A" no espaço operacional aparece na figura 5.2. As linhas tracejadas da figura 5.2 demarcam a região de operação do robô (espaço operacional do mesmo);

as outras linhas mais claras da figura mostram o posicionamento de cada uma das juntas do robô; o traço contínuo mais forte é o que mostra a trajetória executada pelo robô, incluindo sua orientação final (pequenos eixos perpendiculares partindo da pequena ”bolota” que caracteriza o ponto atingido pelo efetuador final do robô). O início do movimento é indicado pela letra ”a” e o final pela letra ”b”. No caso de mais pontos intermediários pertencentes a uma mesma tarefa, letras subseqüentes são utilizadas, por exemplo: ”c”, ”d”’, etc. O retângulo claro que aparece nesta figura se refere a uma placa suporte de inox instalada na horizontal dentro da região de operação do robô. É usada mais tarde nos testes de controladores de forca para apoiar os meios de contato que serão usados. A informação ”step” constante nesta figura se refere ao intervalo de tempo no qual foi mostrada nesta figura cada configuração intermediária assumida pelas juntas do robô durante a execução de uma trajetória. As linhas auxiliares mostradas nesta figura e nas próximas servem apenas como uma referência sobre a forma e a região de operação de cada tarefa sendo executada pelo robô dentro do espaço operacional real disponível para o mesmo. As próximas figuras que mostram a execução de uma tarefa seguem o mesmo padrão gráfico. Os comandos utilizados para gerar esta trajetória foram especificados no espaço de juntas conforme mostrado na tabela 5.2. Os perfis de velocidades e acelerações no espaço de juntas gerados para esta trajetória aparecem na figura 5.3 (pág. 144).



(a) Vista XYθ (b) Vista XZ

Figura 5.2: Trajetória executada para o Movimento ”A”.

Repare pela figura 5.3 que o gerador de trajetórias original utilizado não comanda simultaneamente as 4 juntas do robô. A terceira junta do robô (junta 2, que define a altura em Z

Dados	Junta 0	Junta 1	Junta 2	Junta 3
q_a	-1.9635[rad]	1.5708[rad]	-0.2495[m]	1.5708[rad]
	-112.50^o	90.00^o	-0.2495[m]	90.00^o
q_b	0.7854[rad]	-1.1781[rad]	-0.3700 [m]	-0.7854[rad]
	45.00^o	-67.50^o	-0.3700[m]	-45.00^o
$\dot{q}_{máx}$	1.2320 [rad/s]	-1.2320[rad/s]	-0.1552[m/s]	-1.1160[rad/s]
	70.59[^{o}/s]	-70.59[^{o}/s]	-0.1552[m/s]	-63.94[^{o}/s]
$\ddot{q}_{máx}$	1.0000[rad/s^2]	1.0000 [rad/s^2]	0.4021 [m/s^2]	1.0000 [rad/s^2]
	57.30[^{o}/s^2]	57.30[^{o}/s^2]	0.4021 [m/s^2]	57.30[^{o}/s^2]

Tabela 5.2: Dados da trajetória executada para Movimento "A".

alcançada pelo mesmo) é acionada antes das outras juntas. Repare também a forma com as juntas são desaceleradas ("ruído"). Infelizmente não foi possível implementar com sucesso outro gerador de trajetórias que evitasse *jerks* elevados e movimentasse de modo realmente simultâneo todas as 4 juntas do robô. Estava previsto a geração de quadrifólios e espirais em XY, mantendo altura constante em Z e orientação final do robô constante como demonstra a figura 5.4.

Tabela resumo dos testes realizados

A tabela 5.3 (na pág. 145) resume os dados dos ensaios realizados.

Gráficos dos Erros de Posicionamento

A partir da página 146, a figura 5.5 mostra os erros de seguimento de trajetória para primeira junta (Junta 0) e as figuras 5.6, 5.7 e 5.8, para as juntas 1, 2 e 3 respectivamente.

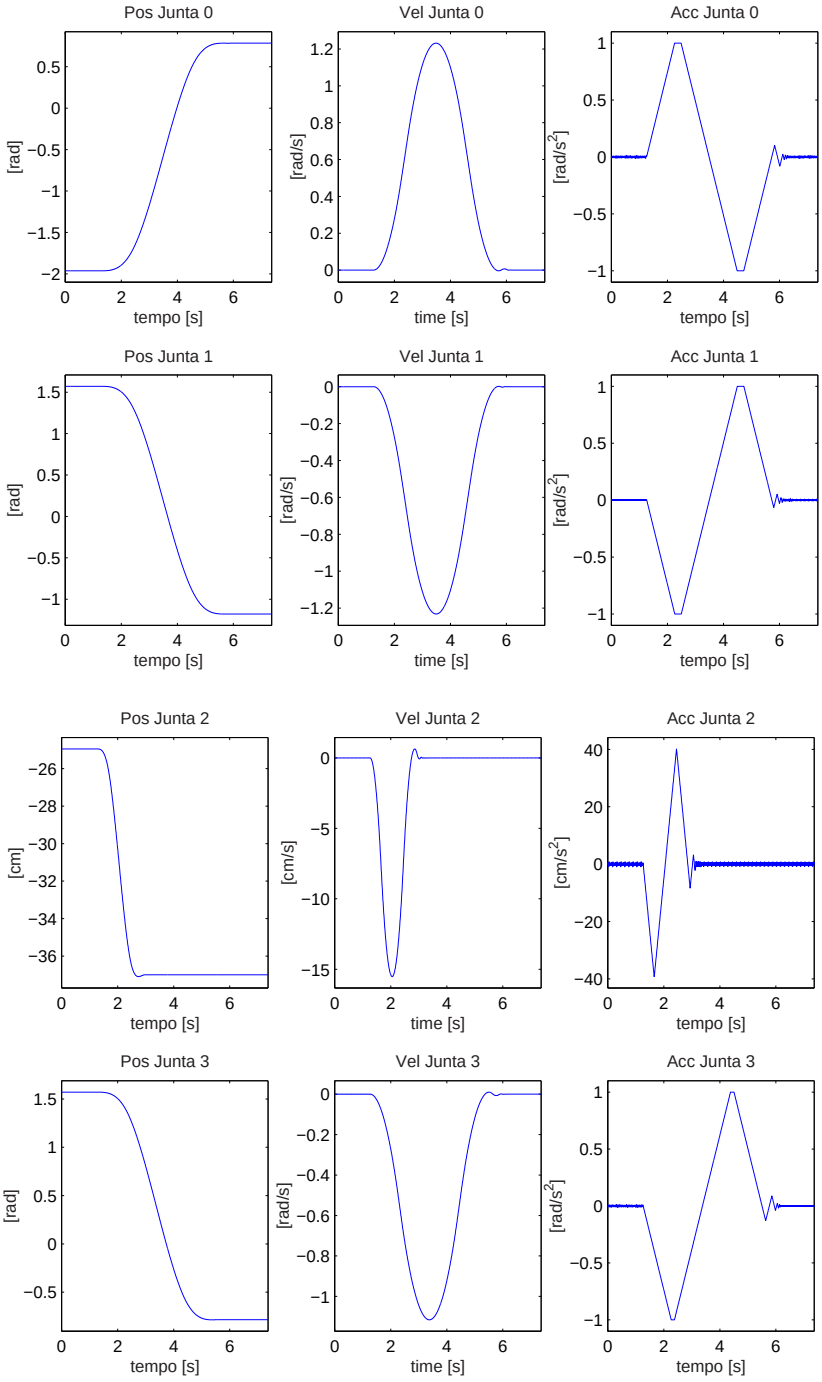
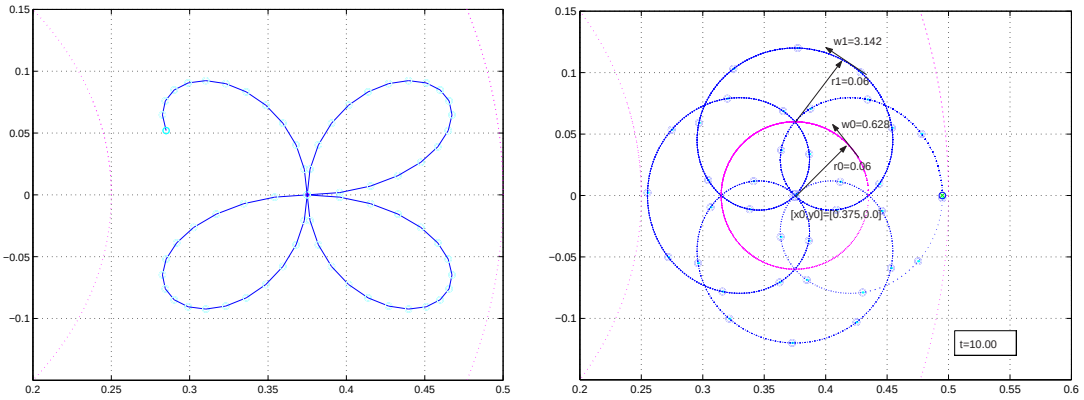


Figura 5.3: Perfis da trajetória do Movimento ″A″.

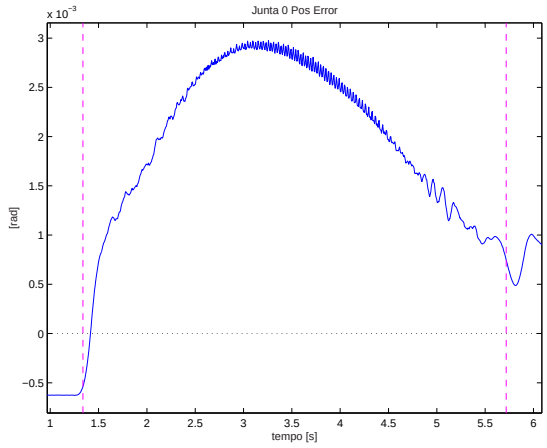


(a) Trajetória de quadrifólio em XY. (b) Trajetória de espirais em XY

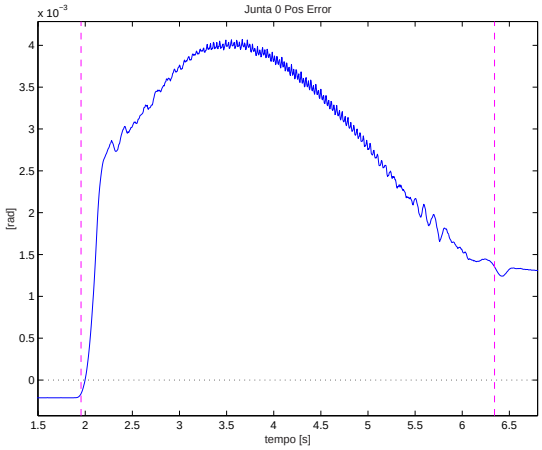
Figura 5.4: Outras trajetória previstas (mas não realizadas).

	Ensaio (Arquivo)	Controlador	Parâmetros
a)	(20042103)	PD de "fábrica"	$K_p = [$ 4900 12100 90000 14400] $K_i = [$ 140 220 600 240]
b)	(21042049)	PID	$K_p = [$ 4900 12100 90000 14400] $K_d = [$ 120 300 2300 350] $K_i = [$ 478 1200 9200 1410]
c)	(21041709)	PD + MLP2c	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
d)	(22040053)	PID + MLP2c	η=0.01, α=0.5 $12 \times 8 \times 5 \times 4$
e)	(21040004)	PD + RBF(5)	η=0.0001, α=0.5 5 funções gaussianas 61 PEs na camada intermediária
f)	(22010200)	PID + RBF(5)	η=0.005, α=0.5 5 funções gaussianas 61 PEs na camada intermediária
g)	(21040102)	PD + RBF(7)	η=0.0001, α=0.5 7 funções gaussianas 85 PEs na camada intemediária
h)	(22040226)	Modos Deslizantes	$c = [$ 10 10 50 10] $\epsilon = [$ 0.3 0.3 1.2 1.2] $K = [$ 50 40 250 5]

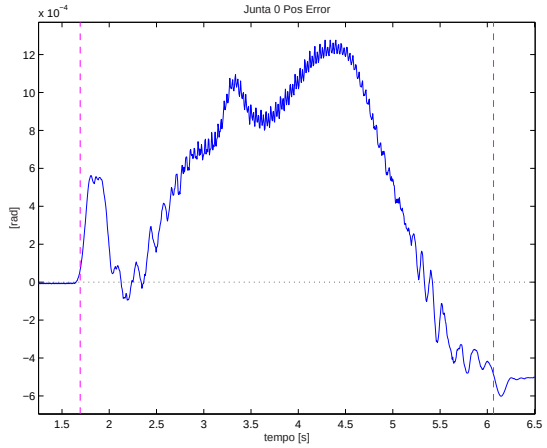
Tabela 5.3: Principais ensaios realizados para o Movimento "A".



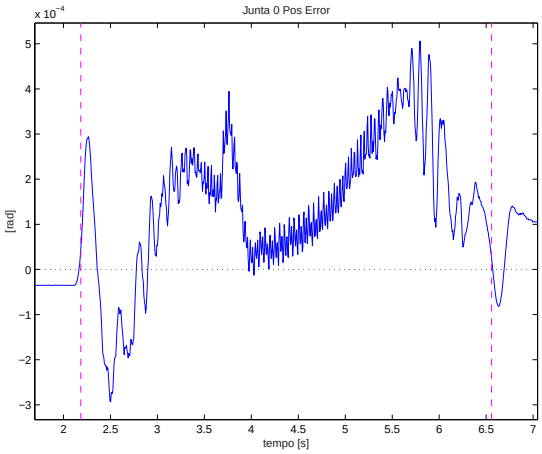
(a) PD de ”fábrica”.



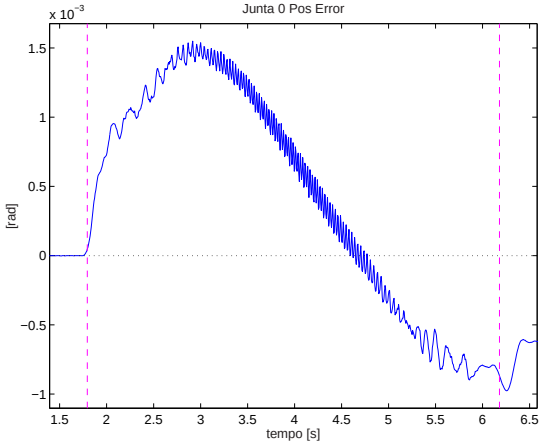
(b) PID.



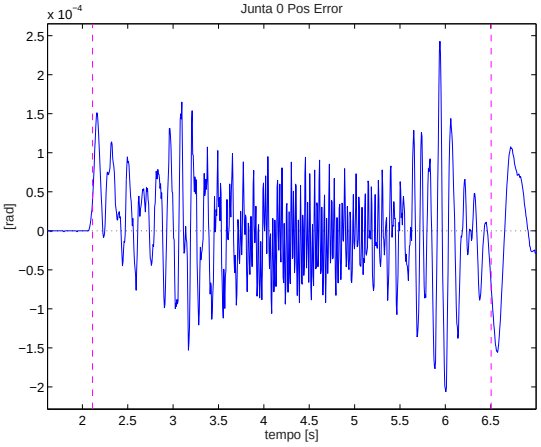
(c) PD + MLP2c (η=0.01 e α=0.7).



(d) PD + MLP2c (η=0.01 e α=0.5).

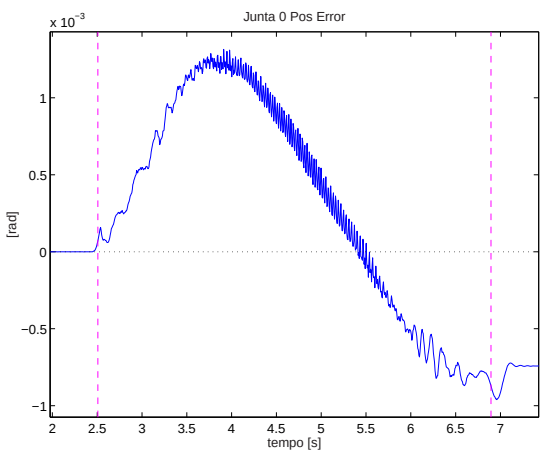


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

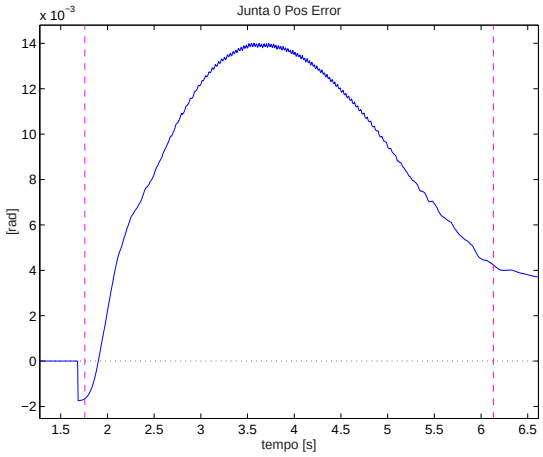


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.5: Erros de seguimento de trajetória para a Junta 0 (Movimento ”A”).

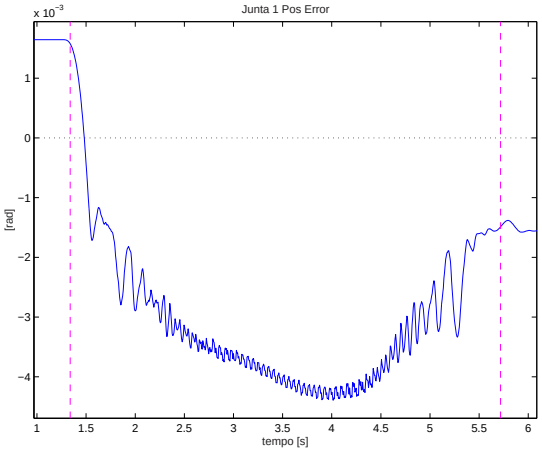


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

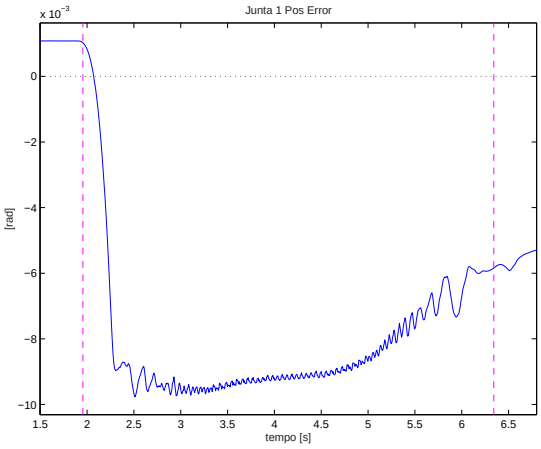


(h) Modos Deslizantes.

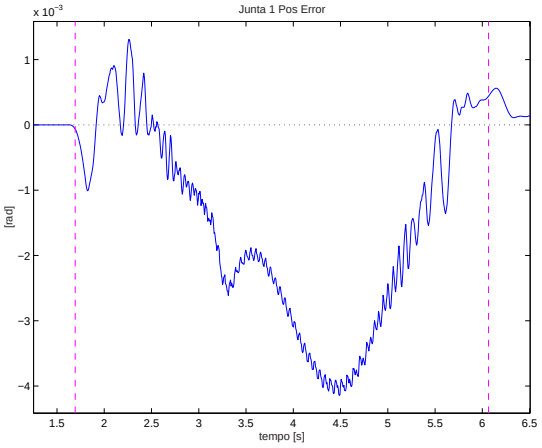
Figura 5.5: (Cont.) Erros de seguimento de trajetória para a Junta 0 (Movimento ″A″).



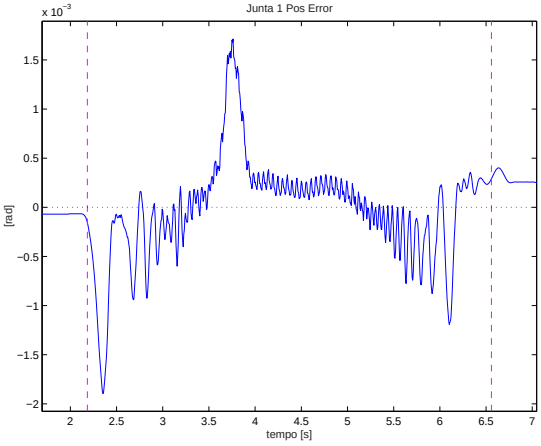
(a) PD de "fábrica"



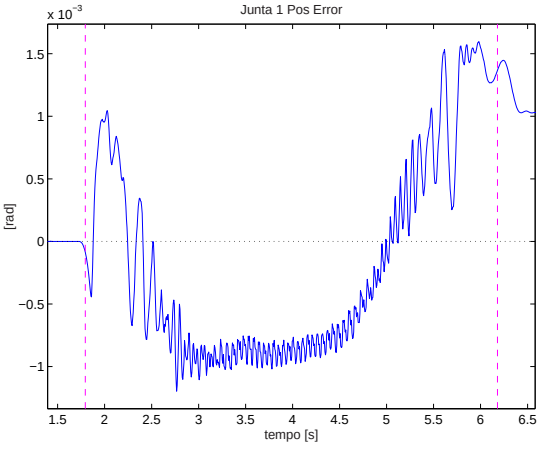
(b) PID



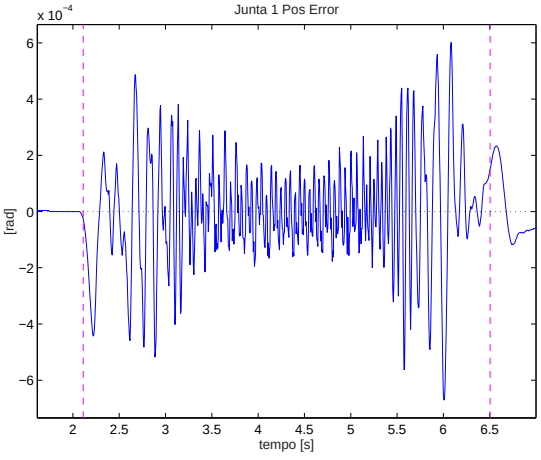
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

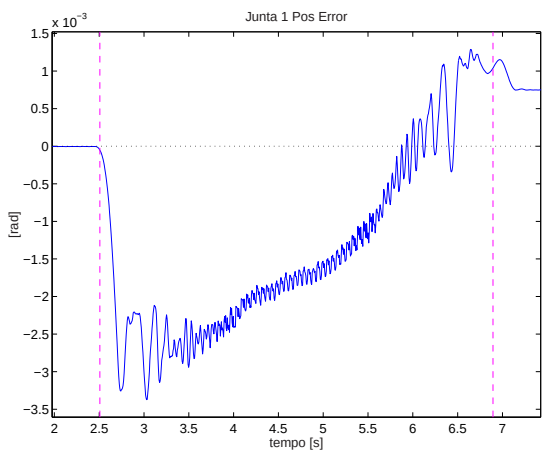


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

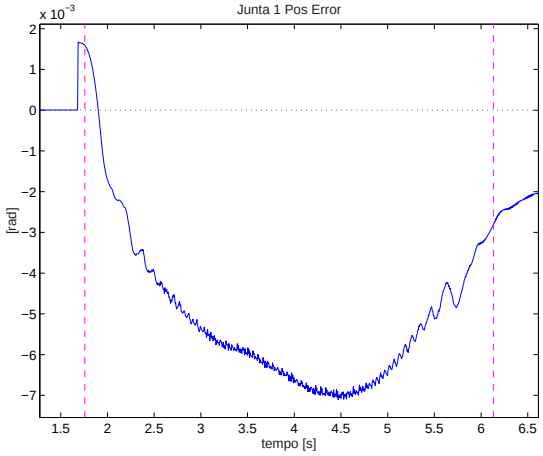


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.6: Erros de seguimento de trajetória para a junta 1 (Movimento "A").

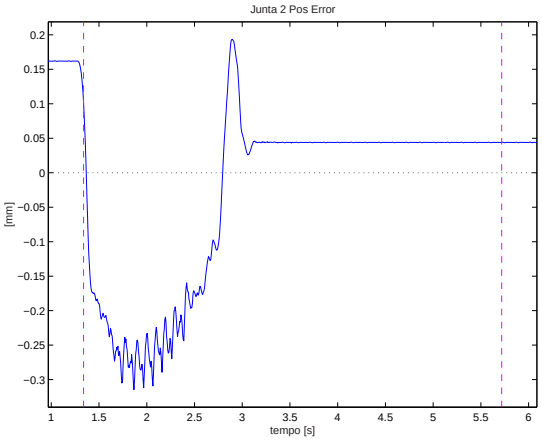


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

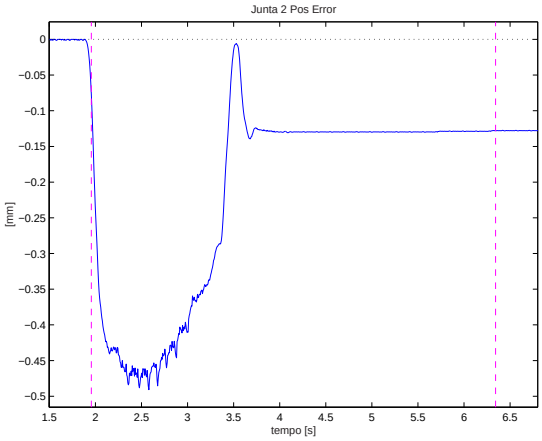


(h) Modos Deslizantes.

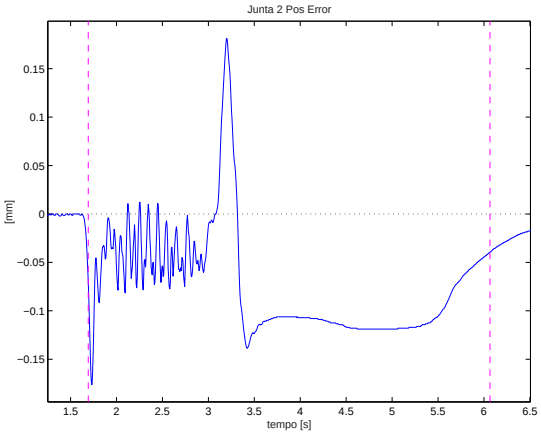
Figura 5.6: (Cont.) Erros de seguimento de trajetória para a Junta 1 (Movimento ″A″).



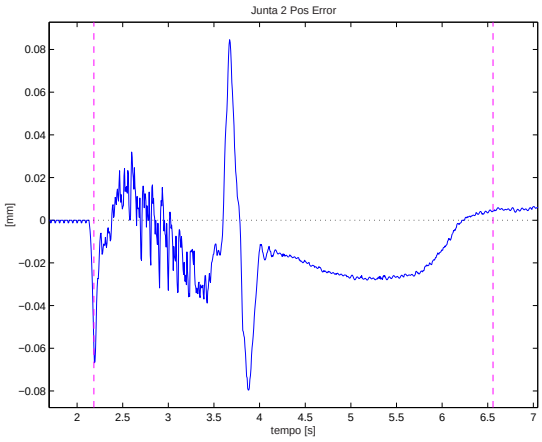
(a) PD de ″fábrica″



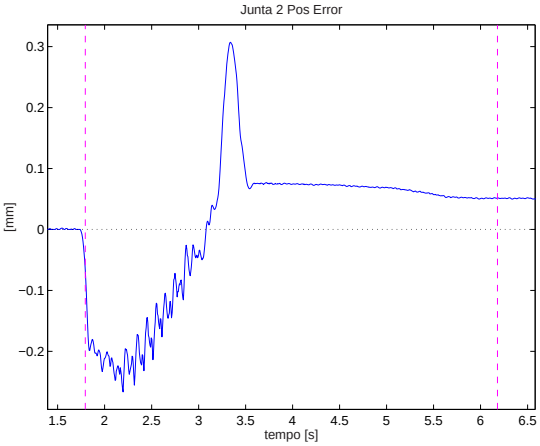
(b) PID



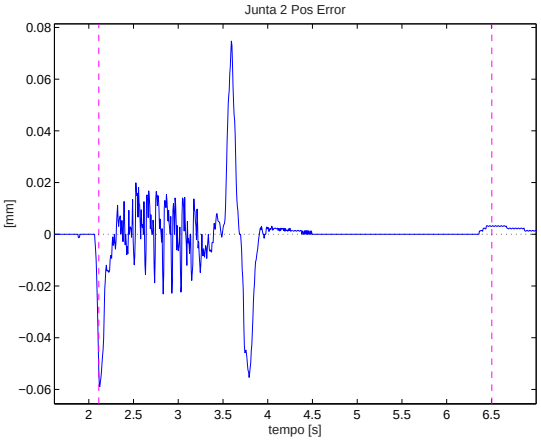
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

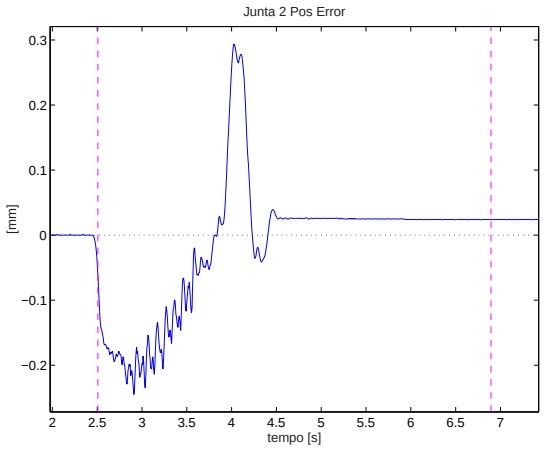


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

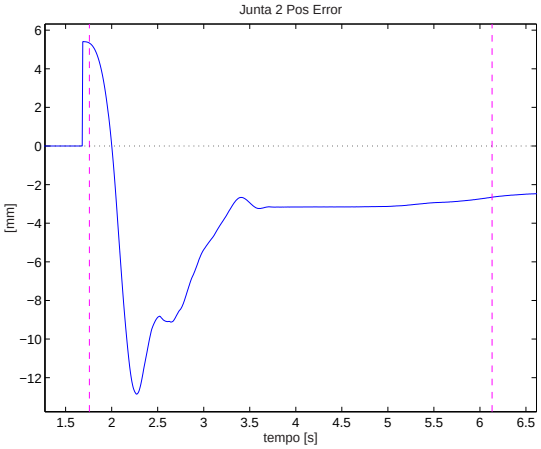


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.7: Erros de seguimento de trajetória para a Junta 2 (Movimento ″A″).

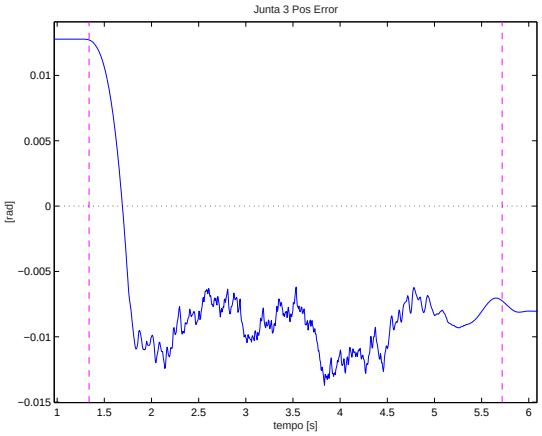


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

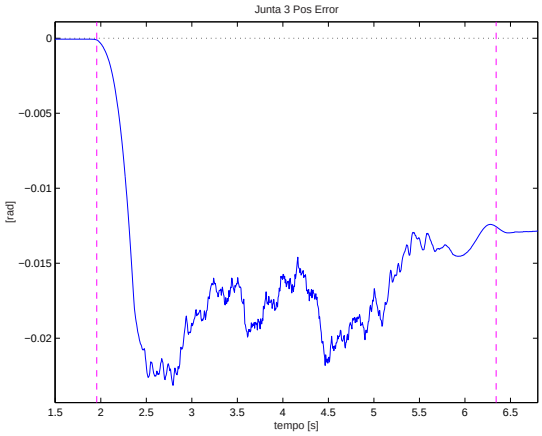


(h) Modos Deslizantes.

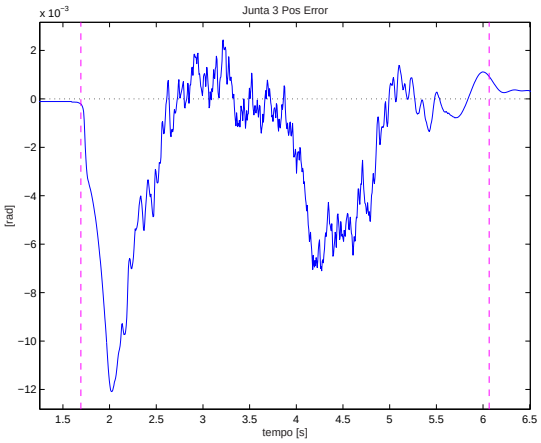
Figura 5.7: (Cont.) Erros de seguimento de trajetória para a Junta 2 (Movimento ″A″).



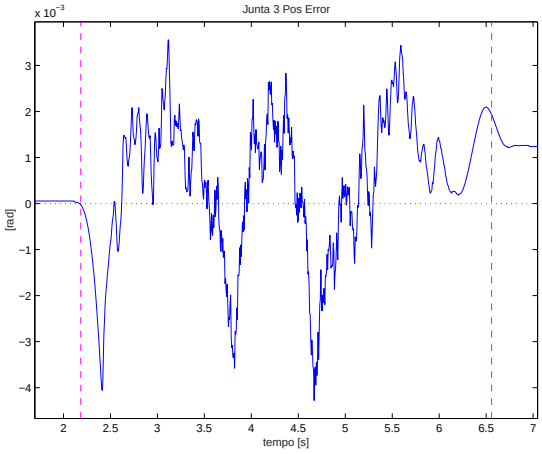
(a) PD de ″fábrica″



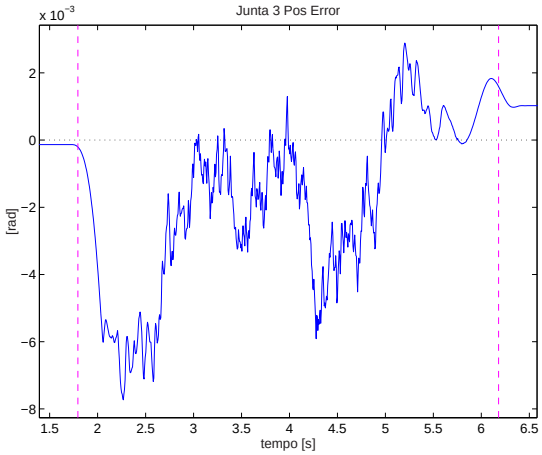
(b) PID



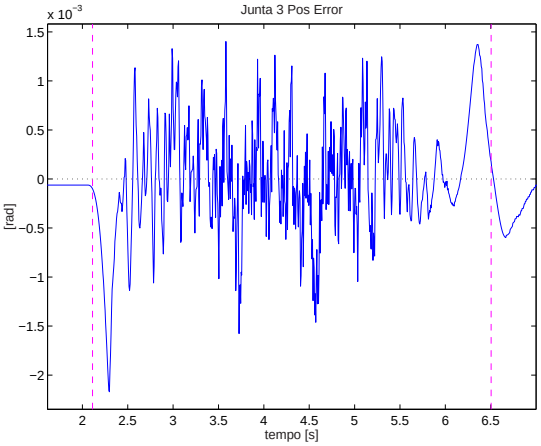
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

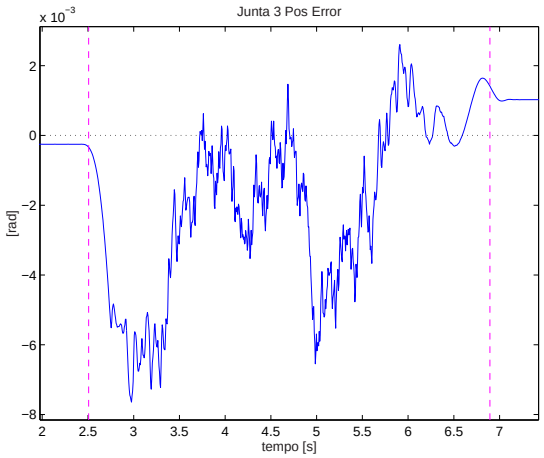


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

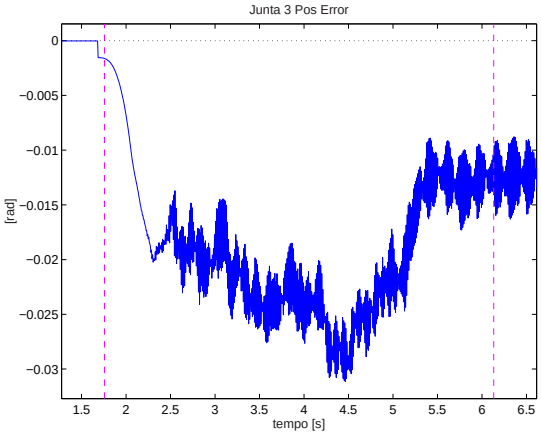


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.8: Erros de seguimento de trajetória para a Junta 3 (Movimento ″A″).



(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

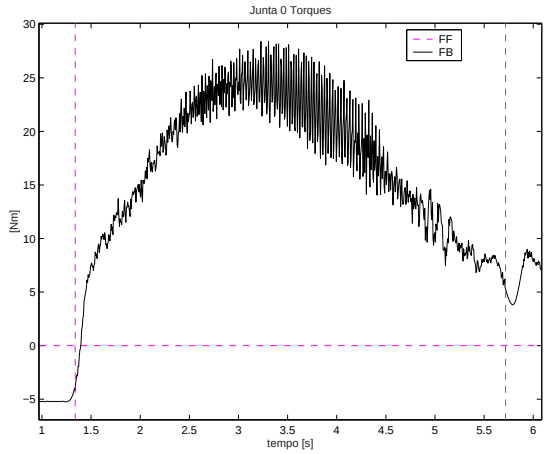


(h) Modos Deslizantes.

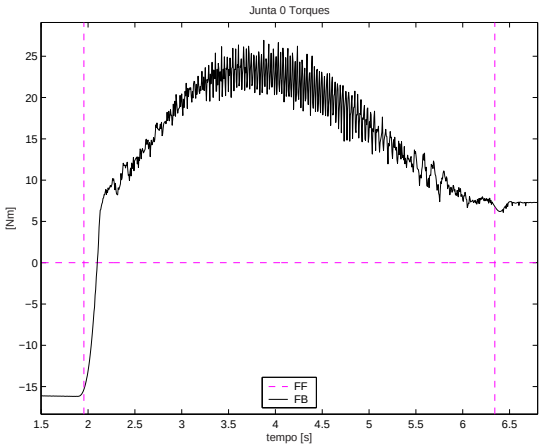
Figura 5.8: (Cont.) Erros de seguimento de trajetória para a Junta 3 (Movimento ″A″).

Gráficos dos Torques Desenvolvidos

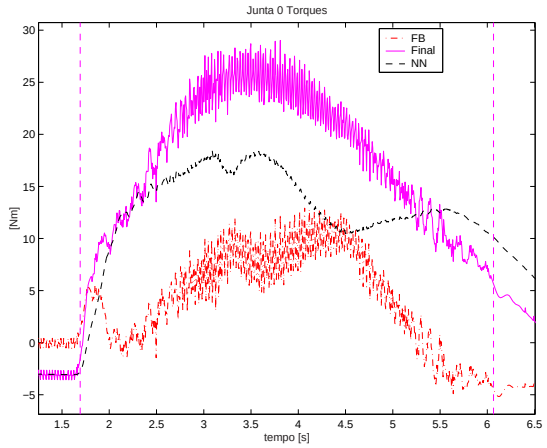
A figura 5.9 mostra os erros de seguimento de trajetória para primeira junta (Junta 0) e as figuras 5.10, 5.11 e 5.12, para as juntas 1, 2 e 3 respectivamente.



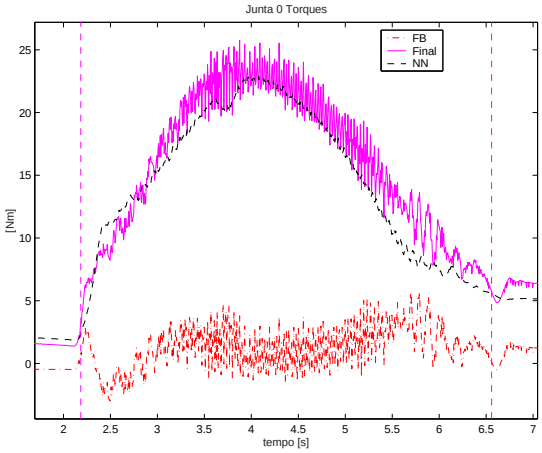
(a) PD de fábrica



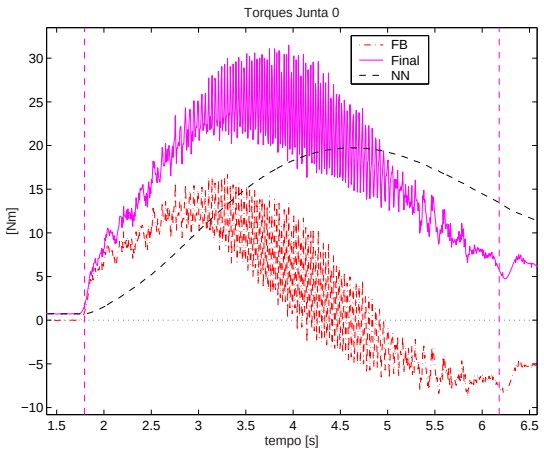
(b) PID



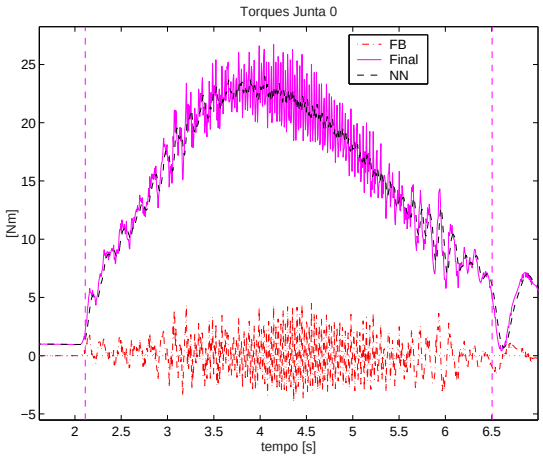
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

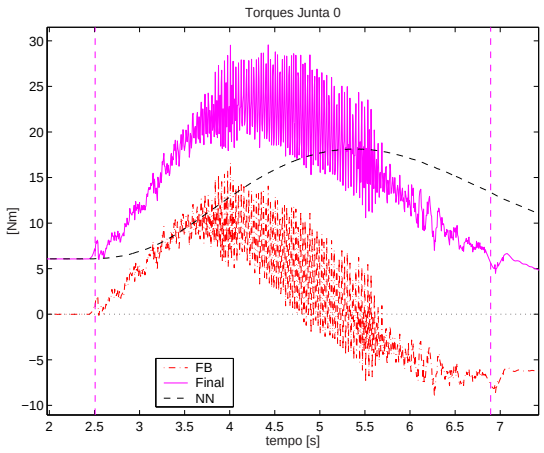


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

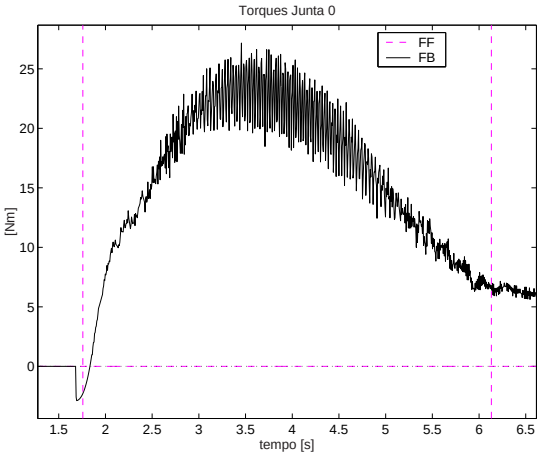


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.9: Torques enviados à junta 0 (Movimento ”A”).

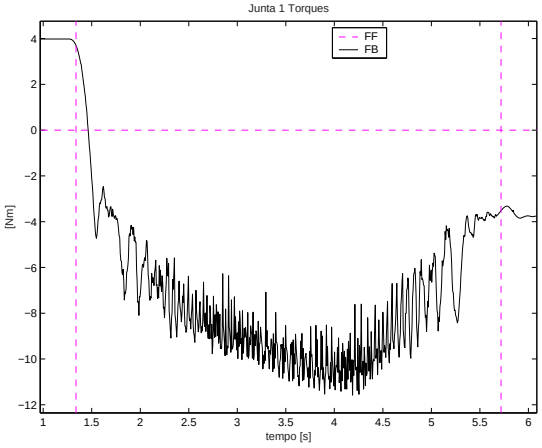


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

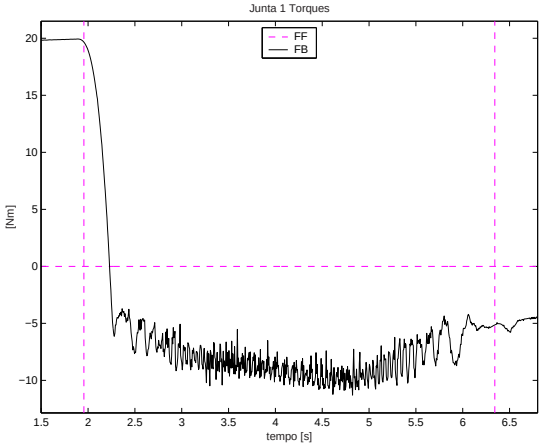


(h) Modos Deslizantes.

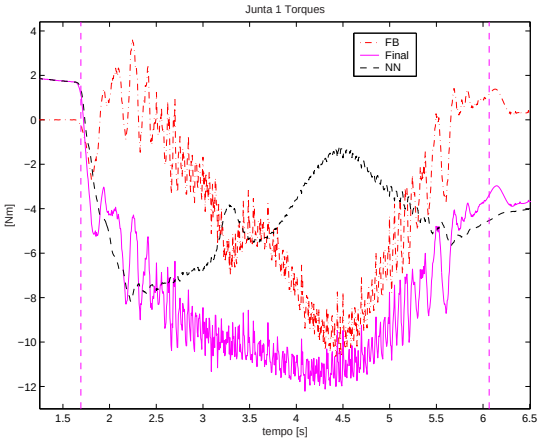
Figura 5.9: (Cont.) Torques enviados à junta 0 (Movimento ″A″).



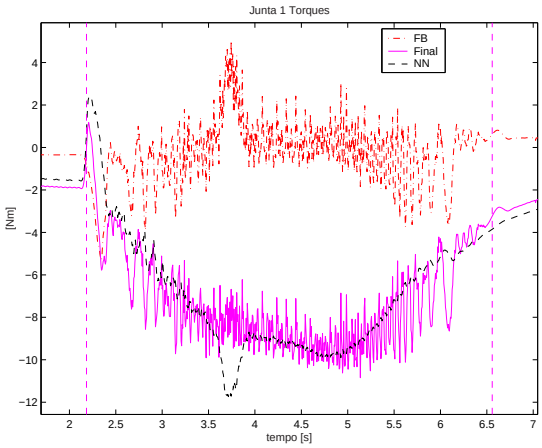
(a) PD de fábrica



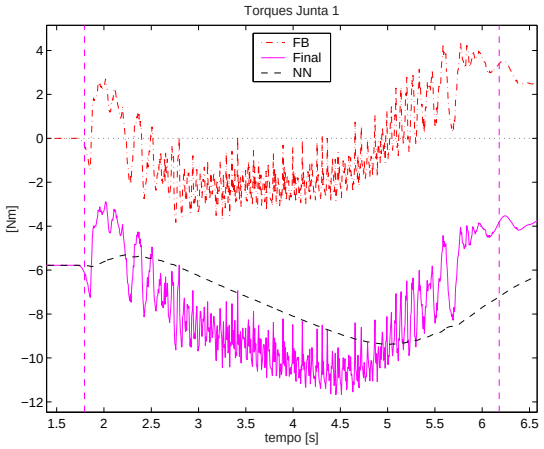
(b) PID



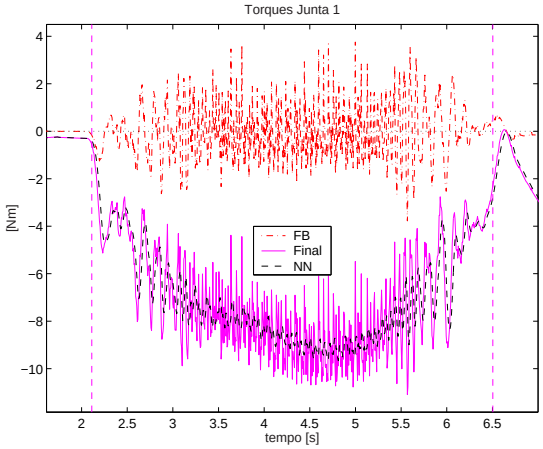
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

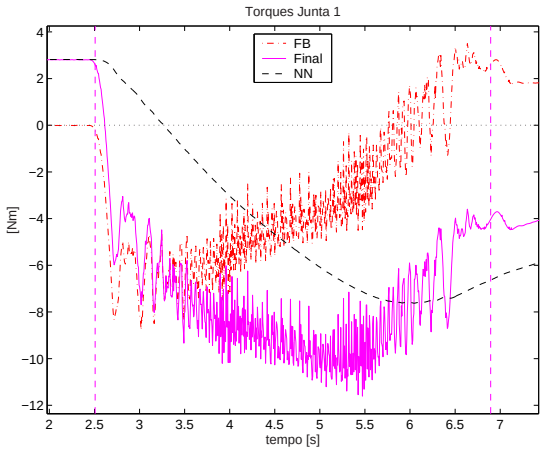


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

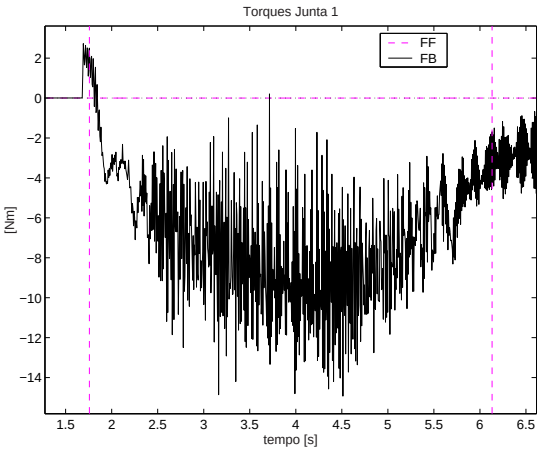


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.10: Torques enviados à junta 1 (Movimento ”A”).

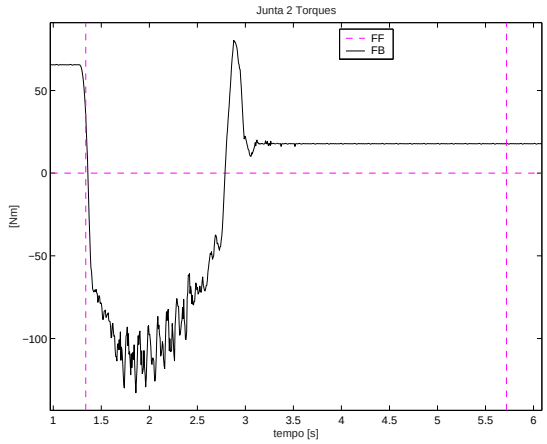


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

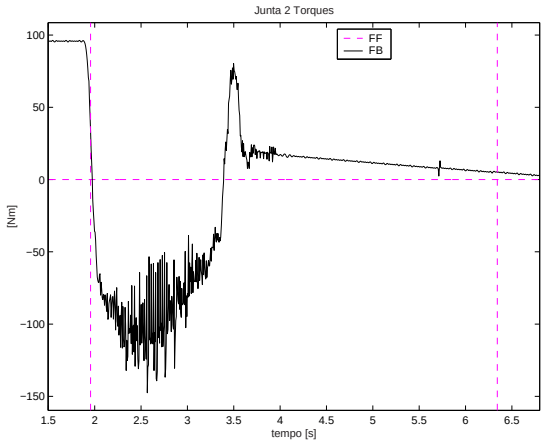


(h) Modos Deslizantes.

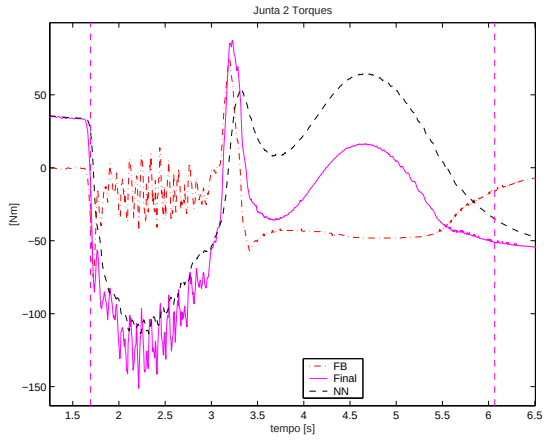
Figura 5.10: (Cont.) Torques enviados à junta 1 (Movimento ″A″).



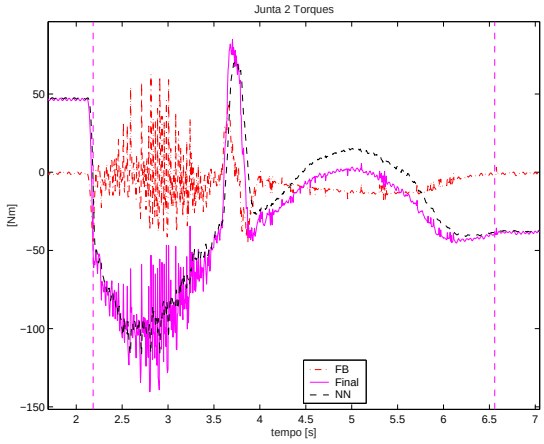
(a) PD de fábrica



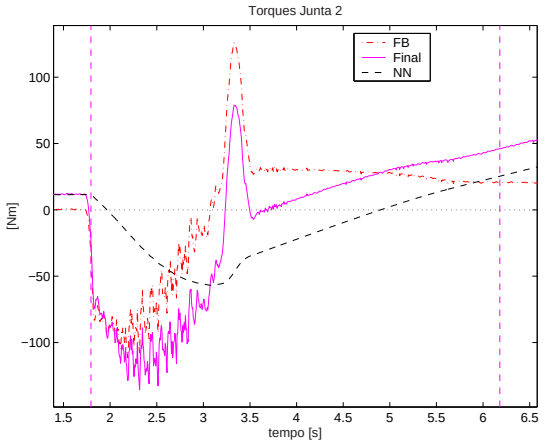
(b) PID



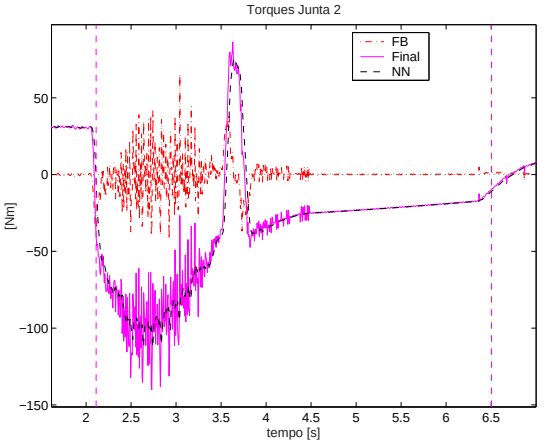
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

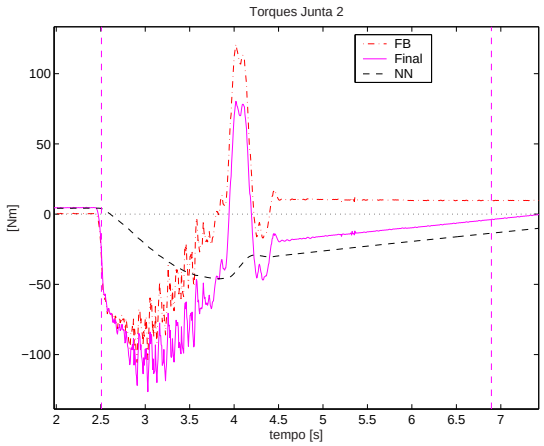


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

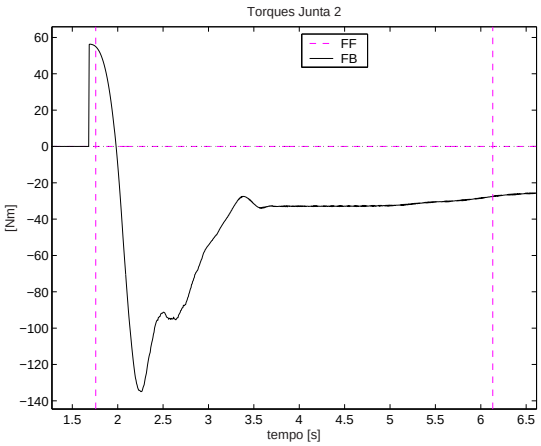


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.11: Torques enviados à junta 2 (Movimento ”A”).

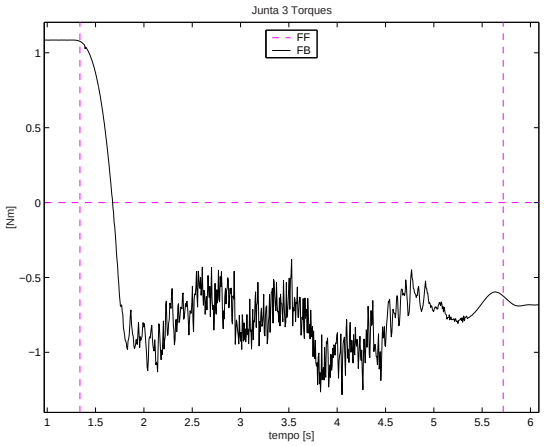


(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).

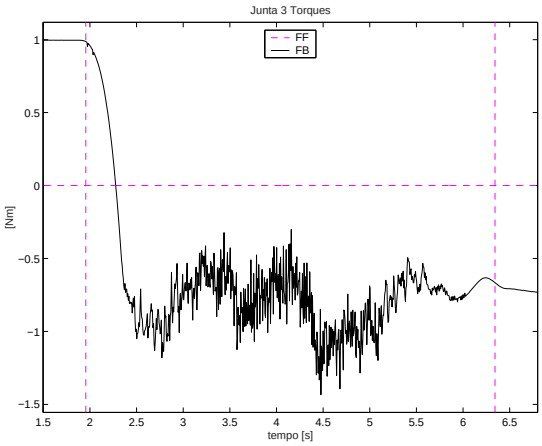


(h) Modos Deslizantes.

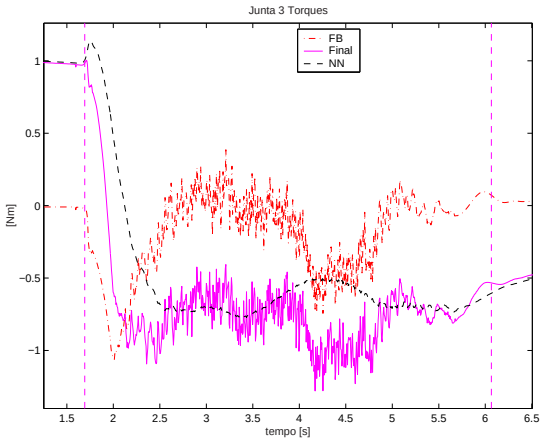
Figura 5.11: (Cont.) Torques enviados à junta 2 (Movimento ″A″).



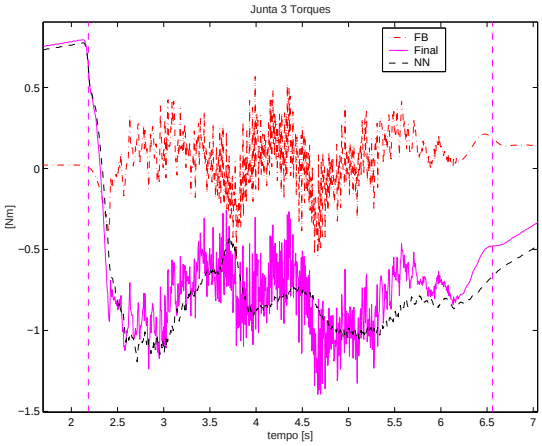
(a) PD de fábrica



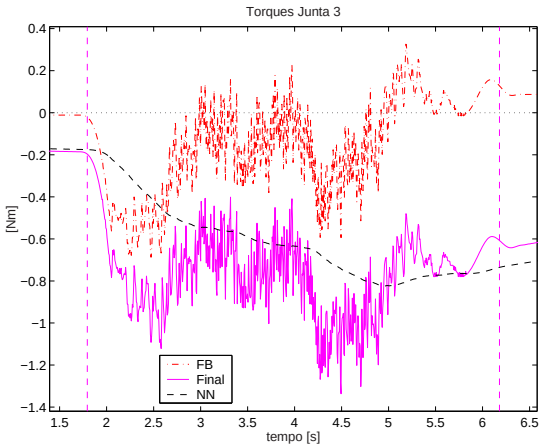
(b) PID



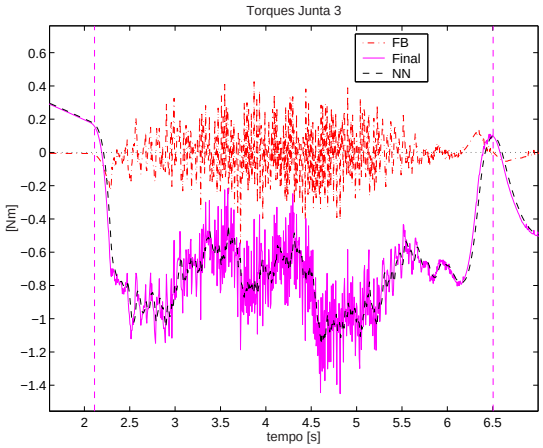
(c) PD + MLP2c (η=0.01 e α=0.7)



(d) PD + MLP2c (η=0.01 e α=0.5)

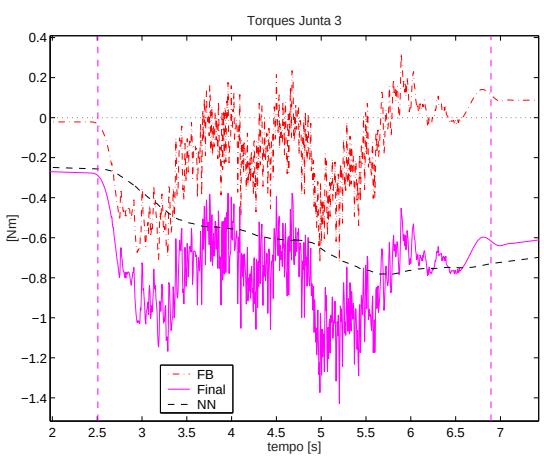


(e) PD + RBF(5) ($\eta = 0.0001$, $\alpha = 0.5$).

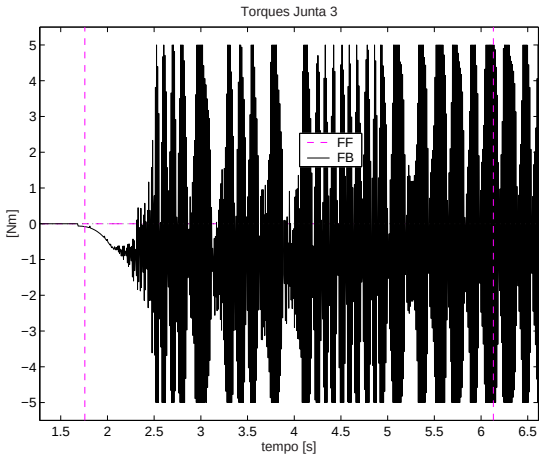


(f) PID + RBF(5) ($\eta = 0.005$ e $\alpha = 0.5$).

Figura 5.12: Torques enviados à junta 3 (Movimento ”A”).



(g) PD + RBF(7) ($\eta = 0.0001$ e $\alpha = 0.5$).



(h) Modos Deslizantes.

Figura 5.12: (Cont.) Torques enviados à junta 3 (Movimento "A").

Índices de Desempenho

R1) $\tilde{q}_f$: Erros de Posicionamento final: tabela 5.4. Unidades utilizadas: para as juntas 0, 1 e 3 (de revolução) os erros são expressos em radianos [rad], e para a junta 2 (prismática), em metros [m]; aos números são adicionados sufixos que representam a escala no sistema SI de unidades alcançada, onde: m (mili) = 10^{-3}; μ = (micro) = 10^{-6}; n (nano) = 10^{-9}; e $< Under >$ se refere a um valor abaixo de 1000p (pico) ou $< Under > < 999.99 \times 10^{-12}$ (abaixo da capacidade de resolução alcançada pelos encoders de posição utilizados no robô).

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	747.95μ	-1.49m	44.05μ	-7.25m
PID	1.35m	-5.84m	-127.79μ	-12.56m
PD + MLP2c (η=0.01, α=07)	-487.07μ	440.77μ	-39.25μ	922.46μ
PD + MLP2c (η=0.01, α=0.5)	23.80μ	290.32μ	4.28μ	1.93m
PD + RBF(5) (η=0.0001, α=0.5)	-867.55μ	1.37m	51.48μ	1.59m
PID + RBF(5) (η=0.005, α=0.5)	-69.54μ	151.42μ	3.11μ	176.22μ
PD + RBF(7) (η=0.0001, α=0.5)	-871.73μ	1.04m	24.05μ	1.42m
Modos Deslizantes	4.23m	-2.79m	-2.65m	-14.07m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.4: Erros de posicionamento final, $\tilde{q}_b$, encontrados para os diferentes controladores.

R2) $MAX\{\tilde{q}\}$: Erros Máximos de Seguimento de Trajetória: tabela 5.5.

R3) $IAE\{\tilde{q}\}$: Integral dos Erros Máximos de Seguimento de trajetória: tabela 5.6.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	2.97m	-4.39m	-314.63μ	-13.71m
PID	4.07m	-9.77m	-490.82μ	-23.14m
PD + MLP2c (η=0.01, α=0.7)	1.28m	-4.14m	181.20μ	-12.09m
PD + MLP2c (η=0.01, α=0.5)	506.01μ	-1.90m	84.58μ	-4.28m
PD + RBF(5) (η=0.0001, α=0.5)	1.55m	1.60m	307.08μ	-7.73m
PID + RBF(5) (η=0.005, α=0.5)	242.74μ	-670.50μ	74.70μ	-2.17m
PD + RBF(7) (η=0.0001, α=0.5)	1.31m	-3.37m	293.74μ	-7.64m
Modos Deslizantes	14.01m	-7.10m	-12.85m	-31.15m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.5: Erros máximos de seguimento de trajetória, $MAX\{\tilde{q}\}$, encontrados para os diferentes controladores.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	1.46	2.26	74.72m	6.63
PID	2.13	5.90	157.78m	11.86
PD + MLP2c (η=0.01, α=0.7)	451.62m	1.26	61.38m	1.93
PD + MLP2c (η=0.01, α=0.5)	131.75m	261.58m	14.83m	949.03m
PD + RBF(5) (η=0.0001, α=0.5)	589.98m	555.67m	73.50m	1.73
PID + RBF(5) (η=0.005, α=0.5)	37.64m	108.00m	4.48m	324.65m
PD + RBF(7) (η=0.0001, α=0.5)	488.74m	1.18	52.18m	1.72
Modos Deslizantes	6.88	3.72	3.22	13.72
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.6: Soma absoluta dos erros de seguimento de trajetória, $IAE\{\tilde{q}\}$, encontrados para os diferentes controladores.

R4) $MED\{\tilde{q}\}$**: Erros Médios de Seguimento de trajetória**: tabela 5.7.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	1.98m	-3.02m	-35.22μ	-7.62m
PID	2.90m	-8.03m	-215.55μ	-16.20m
PD + MLP2c (η=0.01, α=0.7)	508.72μ	-1.53m	-74.01μ	-2.22m
PD + MLP2c (η=0.01, α=0.5)	148.12μ	-13.07μ	-14.10μ	459.51μ
PD + RBF(5) (η=0.0001, α=0.5)	418.79μ	-164.49μ	9.44μ	-1.79m
PID + RBF(5) (η=0.005, α=0.5)	1.37μ	-1.59μ	-365.85n	18.70μ
PD + RBF(7) (η=0.0001, α=0.5)	300.95μ	-1.33m	-14.76μ	-1.90m
Modos Deslizantes	9.35m	-5.03m	-3.99m	-18.80m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.7: Erros médios de seguimento de trajetória, $MED\{\tilde{q}\}$, encontrados para os diferentes controladores.

R5) $MSE\{\tilde{q}\}$**: Erros Quadráticos Médios de Seguimento de trajetória**: tabela 5.8.

R6) $VAR\{\tilde{q}\}$**: Variância dos Erros de Seguimento de trajetória**: tabela 5.9.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	4.53μ	10.56μ	17.87n	86.64μ
PID	9.35μ	69.23μ	65.42n	284.19μ
PD + MLP2c (η=0.01, α=0.7)	524.44n	4.59μ	8.63n	15.24μ
PD + MLP2c (η=0.01, α=0.5)	44.67n	272.06n	$<Under>$	2.49μ
PD + RBF(5) (η=0.0001, α=0.5)	828.24n	704.26n	14.42n	9.42μ
PID + RBF(5) (η=0.005, α=0.5)	4.24n	38.86n	$<Under>$	341.72n
PD + RBF(7) (η=0.0001, α=0.5	578.09n	3.31μ	10.68n	9.52μ
Modos Deslizantes	102.90μ	28.75μ	26.31μ	396.04μ
Unidades	[rad^2]	[rad^2]	[m^2]	[rad^2]

Tabela 5.8: Erros médios quadráticos de seguimento de trajetória, $MSE\{\tilde{q}\}$, encontrados para os diferentes controladores.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	607.20n	1.45μ	16.65n	28.60μ
PID	929.29n	4.77μ	18.99n	21.86μ
PD + MLP2c (η=0.01, α=0.7)	266.01n	2.24μ	3.16n	10.31μ
PD + MLP2c (η=0.01, α=0.5)	22.77n	272.27n	$<Under>$	2.29μ
PD + RBF(5) (η=0.0001, α=0.5)	653.75n	678.13n	14.35n	6.22μ
PID + RBF(5) (η=0.005, α=0.5)	4.25n	38.91n	$<Under>$	341.84n
PD + RBF(7) (η=0.0001, α=0.5)	488.19n	1.55μ	10.47n	5.94μ
Modos Deslizantes	15.49μ	3.47μ	10.37μ	42.62μ
Unidades	[rad^2]	[rad^2]	[m^2]	[rad^2]

Tabela 5.9: Variância dos erros de seguimento de trajetória, $VAR\{\tilde{q}\}$, encontrados para os diferentes controladores.

R7) $MAX\{\tau\}$**: Maior valor da ação de controle**: tabela 5.10.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	28.42	-11.58	-132.83	-1.28
PID	26.93	19.66	-147.54	-1.43
PD + MLP2c (η=0.01, α=0.7)	29.02	-12.21	-151.18	-1.28
PD + MLP2c (η=0.01, α=0.5)	25.75	-10.85	-140.39	-1.40
PD + RBF(5) (η=0.0001, α=0.5)	31.48	-11.67	-135.53	-1.34
PID + RBF(5) (η=0.005, α=0.5)	26.71	-11.09	-140.05	-1.45
PD + RBF(7) (η=0.0001, α=0.5)	29.56	-11.61	-126.43	-1.43
Modos Deslizantes	27.17	-14.93	-134.97	5.00
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.10: Maiores ações de controle, $MAX\{\tau\}$, encontrados para os diferentes controladores.

R8) $MED\{\tau\}$**: Valor Médio da ação de controle**: tabela 5.11.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	16.63	-7.36	-14.34	-654.95m
PID	15.42	-6.46	-16.55	-676.81m
PD + MLP2c (η=0.01, α=0.7)	17.46	-8.31	-38.86	-692.04m
PD + MLP2c (η=0.01, α=0.5)	15.87	-7.17	-35.63	-743.13m
PD + RBF(5) (η=0.0001, α=0.5)	17.03	-7.91	-11.17	-762.47m
PID + RBF(5) (η=0.005, α=0.5)	15.74	-6.97	-37.26	-710.94m
PD + RBF(7) (η=0.0001, α=0.5)	15.97	-7.31	-30.97	-766.44m
Modos Deslizantes	15.83	-6.86	-41.98	-541.79m
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.11: Valores médios das ações de controle, $MED\{\tau\}$, encontrados para os diferentes controladores.

R9) $VARi\{\tau\}$**: Variação Média da ação de controle**: tabela 5.12.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	1.60	619.91m	1.96	65.96m
PID	1.29	692.47m	6.32	96.60m
PD + MLP2c (η=0.01, α=0.7)	1.17	430.99m	2.38	68.24m
PD + MLP2c (η=0.01, α=0.5)	1.00	612.45m	6.07	120.79m
PD + RBF(5) (η=0.0001, α=0.5)	2.14	505.76m	2.14	70.34m
PID + RBF(5) (η=0.005, α=0.5)	1.30	828.03m	6.01	126.65m
PD + RBF(7) (η=0.0001, α=0.5)	2.31	670.99m	2.11	78.82m
Modos Deslizantes	1.50	3.08	642.43m	5.70
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.12: Variações médias das ações de controle, $VARi\{\tau\}$, encontrados para os diferentes controladores.

Conclusões preliminares

As seguintes conclusões podem ser extraídas dos testes realizados anteriormente:

- O controlador que apresentou os melhores resultados foi o integrado PID + RBF(5), com $\eta = 0.005$ e $\alpha = 0.5$ – ver figuras 5.5(f) à 5.8(f).

- É nítido o efeito gerado pela ação integral do controlador convencional que trabalha em paralelo com uma rede neural. A rede neural responde mais prontamente a algum comando de mudança de configuração do robô além de executar a trajetória com menor erro de seguimento de trajetória.

- O controlador baseado em modos deslizantes é o que apresenta as ações de controle mais suaves – nitidamente a figura 5.11(h) (notar o índice de desempenho $VARi\{\tau\}$ obtido para a junta 2, na tabela 5.12 – junta melhor sintonizada), mas somente quando bem sintonizado. Os maiores erros de seguimento de trajetória se devem à dificuldade para sintonizar este controlador por não haver um método determinístico para ajuste de seus parâmetros. Na figura 5.12(h) se percebe claramente o efeito de *chattering* devido à parâmetros com valores que se revelaram inadequados (este controlador havia sido sintonizado para o robô numa outra configuração); neste caso, seria necessário incrementar o parâmetro ϵ, que aumentaria o tamanho da camada limite para a superfície de deslizamento para a junta 3 e provavelmente rebaixar também o parâmetro K. Note que o ajuste deste controlador depende muito do método por tentativa e erro. Entretanto, se bem sintonizado este controlador pode demonstrar um desempenho superior a um PID como ocorreu para a junta 1 – menor variação da ação de controle (tabela 5.12) além de menores erros de posicionamento da trajetória, tanto levando em conta o erro de posição no ponto final da trajetória (tabela 5.5) quanto o acúmulo de erros de seguimento de trajetória (tabela 5.6). Entretanto mesmo para esta junta, o controlador PID + RBF(5) permitiu alcançar um desempenho muito superior.

- Verificando as figuras 5.8(c)(PD+MLP2c, $\eta = 0.01$ e $\alpha = 0.7$) e 5.8(d)(PD+MLP2c, $\eta = 0.01$ e $\alpha = 0.5$) se percebe que mesmo que seja mantida a mesma taxa de aprendizado mas um incremento seja dado ao termo *momentum* (α) a rede neural se torna mais lenta. O termo *momentum*, $\alpha = 0.5$ se revelou o que leva a rede ao melhor compromisso: velocidade de resposta da rede $\times$ erros de seguimento de trajetória. Este efeito também pode ser percebido na ação de controle gerado pelas redes com diferentes momentos, ver figura 5.9(c) e 5.9(d), que trata dos torques enviados para a junta 1 por estas duas redes. A rede com termo *momentum*, $\alpha = 0.5$ permite o melhor resultado, com a ação predominante de controle sendo gerada pela rede neural – figura 5.9(d).

- Também através das figuras 5.9(e) e figura 5.9(f) percebe-se que a taxa de aprendizado, $\eta = 0.005$ se mostra a mais adequada para a rede RBF (ao menos quando utilizada com com período de amostragem igual à $T_s = 1[ms]$).

- Usar uma rede RBF com 7 funções gaussianas na sua camada intermediária não acar-

reta em uma melhora muito significativa se comparado com uma rede RBF com apenas 5 funções gaussinas, principalmente se for mantida uma taxa de aprendizado como $\eta = 0.0001$ que ser revelou baixa. A taxa de aprendizado que se revelou mais adequada para uso com as redes RBF foi de $\eta = 0.005$. Notar que uma rede RBF com 7 funções gaussianas acarreta numa sobrecarga de processamento para a CPU do robô em aproximadamente 17.27% sem equivalente melhora no desempenho. O aumento no número de funções gaussianas se traduz num ligeiro menor esforço de controle (ver figura 5.11(g)) com baixo impacto no erro de seguimento de trajetória.

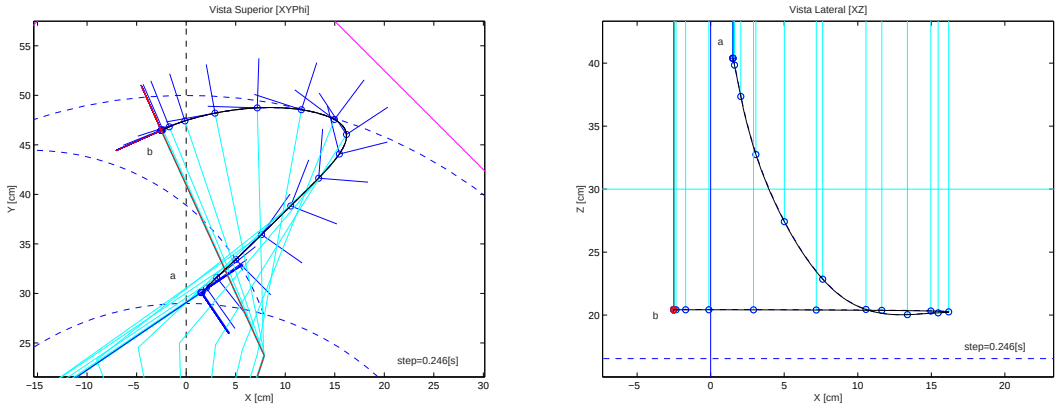
- Comparando-se o desempenho obtido entre o controlador PD + MLP2c (com $\eta = 0.01$ e $\alpha = 0.5$) e o controlador PD + RBF(5) (com $\eta = 0.005$ e $\alpha = 0.5$) referente à máxima ação de controle realizada (tabela 5.10) se percebe que a rede MLP exige um pouco menos dos atuadores de potência do robô mas não consegue erros de seguimento de trajetória tão baixos quanto a rede RBF.

- A respeito dos índices de desempenho utilizados, percebeu-se uma certa redundância entre os índices $IAE\{\tilde{q}\}$, $MED\{\tilde{q}\}$ e $MSE\{\tilde{q}\}$. Teria sido mais interessante permanecer com o índice $MSE\{\tilde{q}\}$ que ressalta melhor os erros acumulados de seguimento da trajetória.

5.5.2 Testes com a Trajetória "B"

Dados da Trajetória "B"

O perfil da trajetória "B" gerado no espaço operacional aparece na figura 5.13. Os comandos utilizados para gerar esta trajetória foram especificados no espaço de juntas conforme mostrado na tabela 5.13. Os perfis de velocidades e acelerações no espaço de juntas gerados para esta trajetória aparecem na figura 5.14.



(a) Vista XYθ (b) Vista XZ

Figura 5.13: Trajetória executada para o Movimento ”B”.

Dados	Junta 0	Junta 1	Junta 2	Junta 3
q_a	2.4445 [rad]	-1.8500 [rad]	-0.2547 [m]	-1.5708 [rad]
	140.06^o	-106.00^o	-0.2547 [m]	-90.00^o
q_b	1.2494 [rad]	0.7512 [rad]	-0.4606 [m]	0.0000 [rad]
	71.59^o	43.04^o	-0.4606 [m]	0.00^o
$\dot{q}_{máx}$	-0.7123 [rad/s]	1.1900 [rad/s]	-0.2209 [m/s^2]	0.8538 [rad/s]
	-40.81 [^{o}/s]	68.18 [^{o}/s]	-0.2209 [m/s^2]	48.92 [^{o}/s]
$\ddot{q}_{máx}$	0.8500 [rad/s^2]	1.0000 [rad/s^2]	0.4740 [m/s^2]	-0.9300 [rad/s^2]
	48.70 [^{o}/s^2]	57.30 [^{o}/s^2]	0.4740 [m/s^2]	-53.29 [^{o}/s^2]

Tabela 5.13: Dados da trajetória executada para Movimento ”B”.

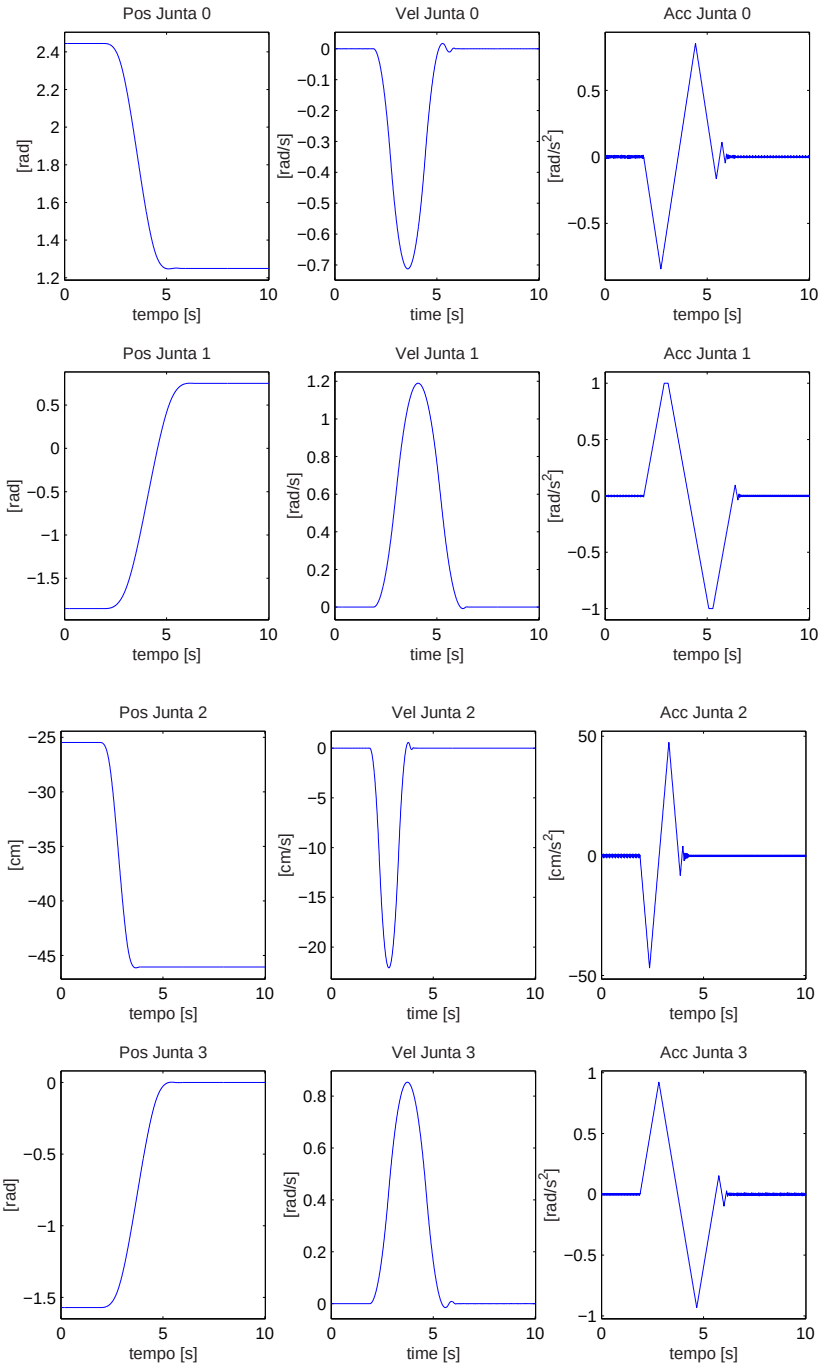


Figura 5.14: Perfis da trajetória do Movimento ″B″.

Tabela resumo dos testes realizados

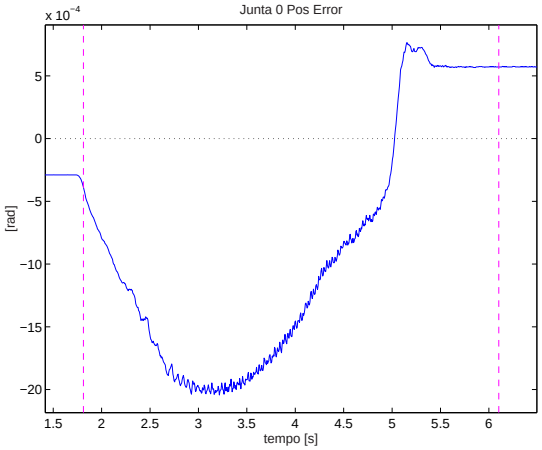
A tabela 5.14 (pág. 169) resume os dados dos ensaios que serão apresentados.

	Ensaio (Arquivo)	Controlador	Parâmetros
a)	(21040049)	PD de "fábrica"	$K_p = [$ 4900 12100 90000 14400] $K_i = [$ 140 220 600 240]
b)	(21042051)	PID	$K_p = [$ 4900 12100 90000 14400] $K_d = [$ 120 300 2300 350] $K_i = [$ 478 1200 9200 1410]
c)	(21041945)	PD + MLP1c	η=0.01, α=0.7 $12 \times 8 \times 4$
d)	(21040044)	PD + MLP2c	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
e)	(22040033)	PID + MLP2c	η=0.035, α=0.7 $12 \times 8 \times 5 \times 4$
f)	(21040054)	PD + RBF(5)	η=0.0001, α=0.5 5 funções gaussianas 61 PEs na camada intermediária
g)	(22040208)	PID + RBF(5)	η=0.005, α=0.5 5 funções gaussianas 61 PEs na camada intermediária
h)	(21040106)	PD + RBF(7)	η=0.0001, α=0.5 7 funções gaussianas 85 PEs na camada intemediária
i)	(22040227)	Modos Deslizantes	$c = [$ 10 10 50 10] $\epsilon = [$ 0.3 0.3 1.2 1.2] $K = [$ 50 40 250 5]

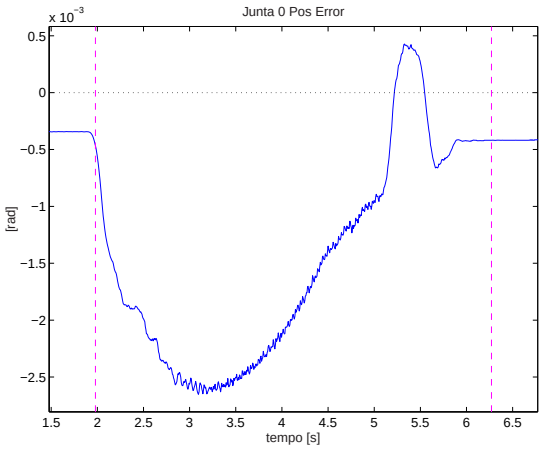
Tabela 5.14: Principais ensaios realizados para o Movimento "B".

Gráficos dos Erros de Posicionamento

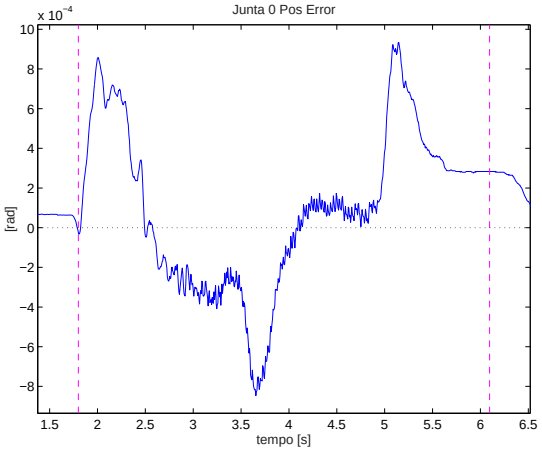
A figura 5.15, a partir da página 170 mostra os erros de seguimento de trajetória para primeira junta (Junta 0) e as figuras 5.16, 5.17 e 5.18, para as juntas 1, 2 e 3 respectivamente.



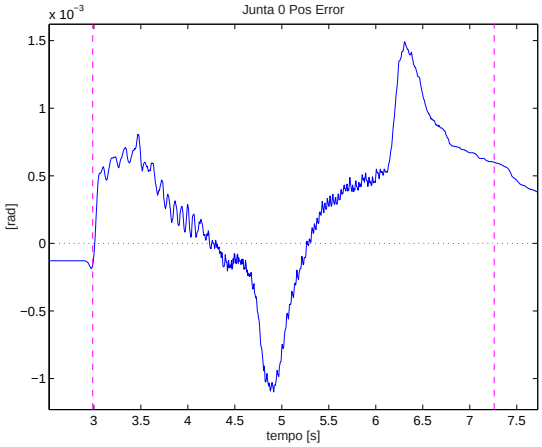
(a) PD de ”fábrica”.



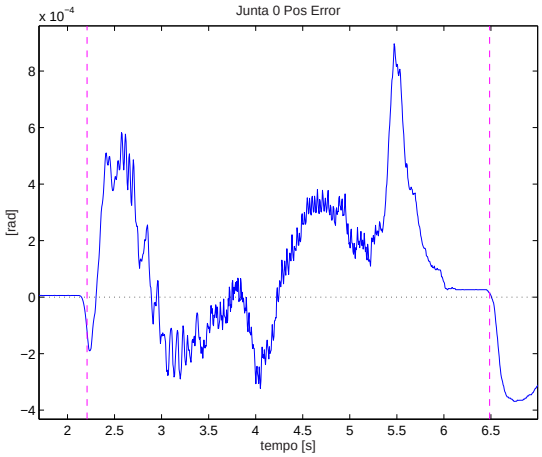
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

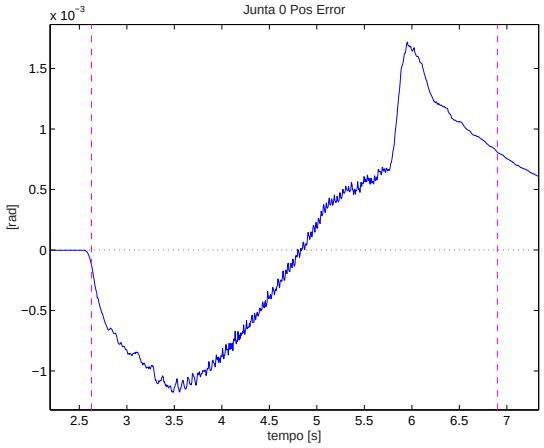


(d) PD + MLP2c (η=0.01, α=0.7).

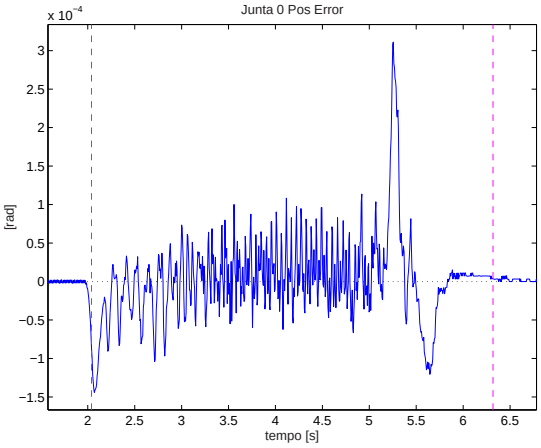


(e) PID + MLP2c (η=0.035, α=0.7).

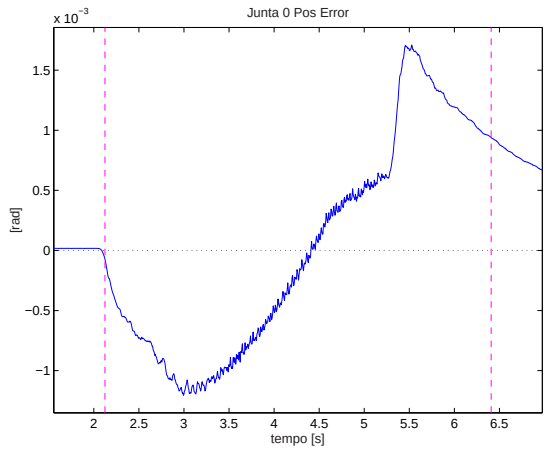
Figura 5.15: Erros de seguimento de trajetória para a Junta 0 (Movimento ”B”).



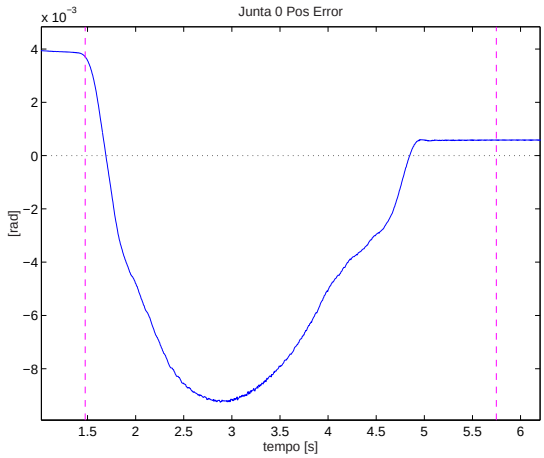
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

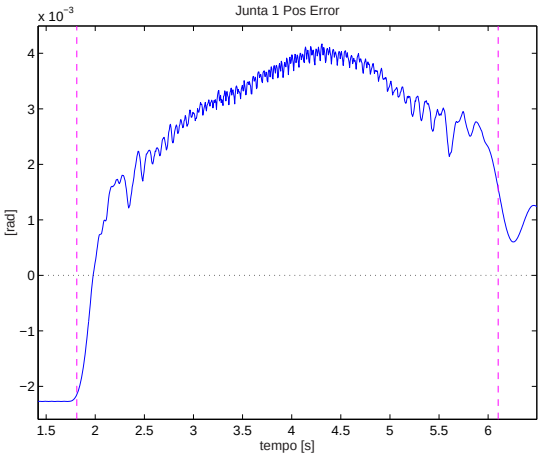


(h) PD + RBF(7) (η=0.0001, α=0.5).

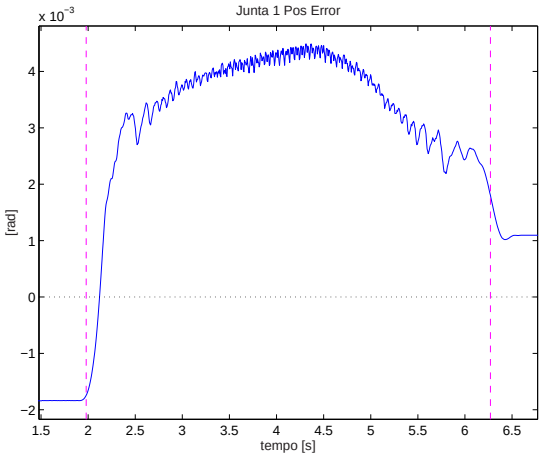


(i) Modos Deslizantes.

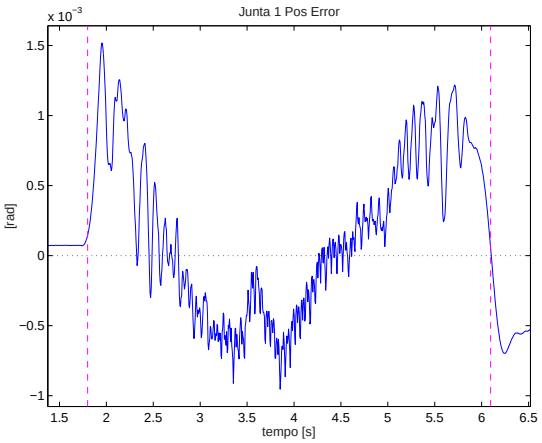
Figura 5.15: (Cont.) Erros de seguimento de trajetória para a Junta 0 (Movimento ″B″).



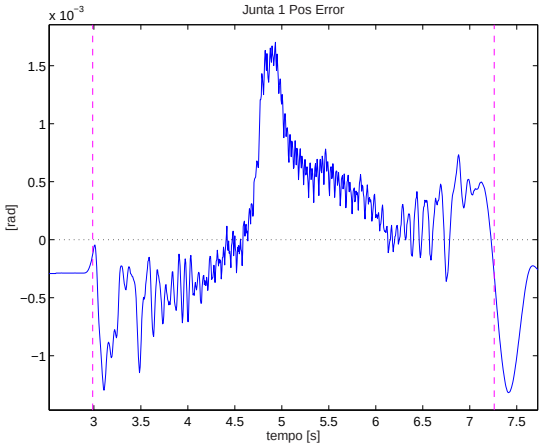
(a) PD de ”fábrica”.



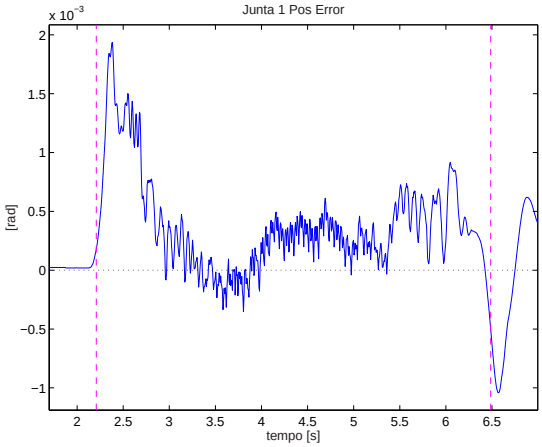
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

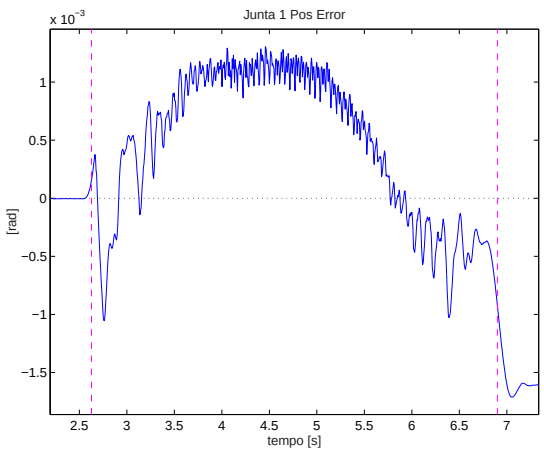


(d) PD + MLP2c (η=0.01, α=0.7).

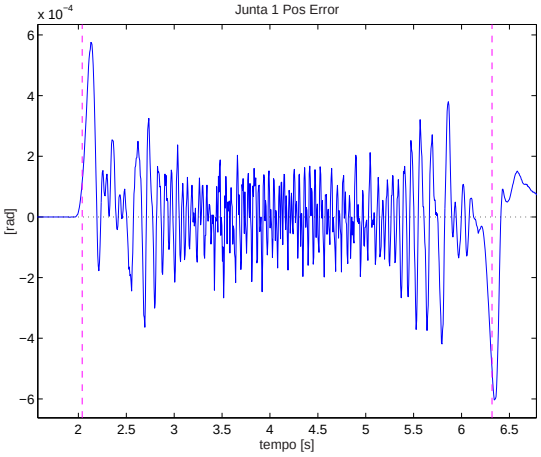


(e) PID + MLP2c (η=0.035, α=0.7).

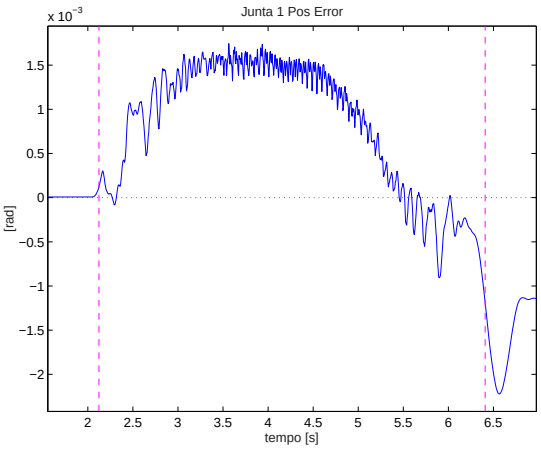
Figura 5.16: Erros de seguimento de trajetória para a Junta 1 (Movimento ”B”).



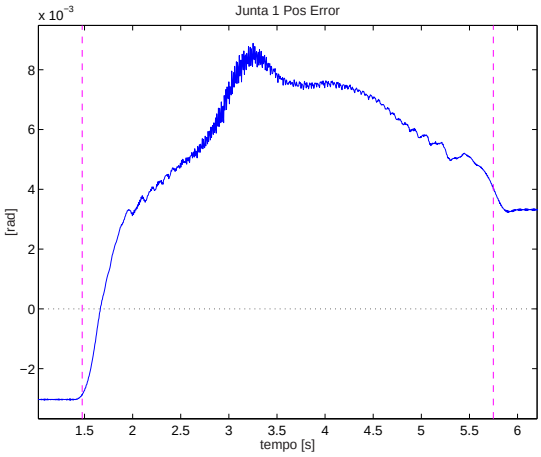
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

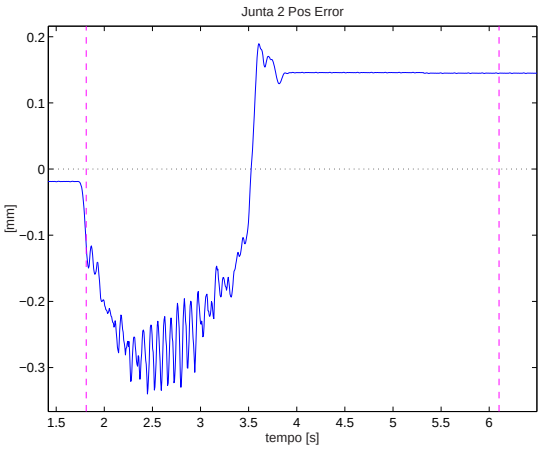


(h) PD + RBF(7) (η=0.0001, α=0.5).

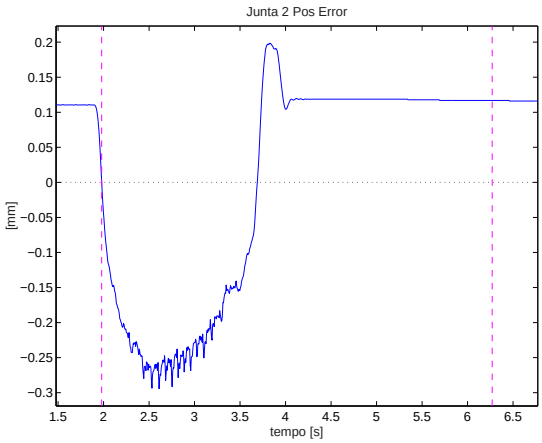


(i) Modos Deslizantes.

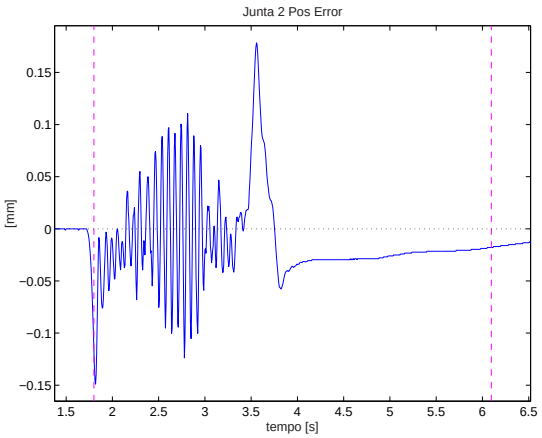
Figura 5.16: (Cont.) Erros de seguimento de trajetória para a Junta 1 (Movimento ”B”).



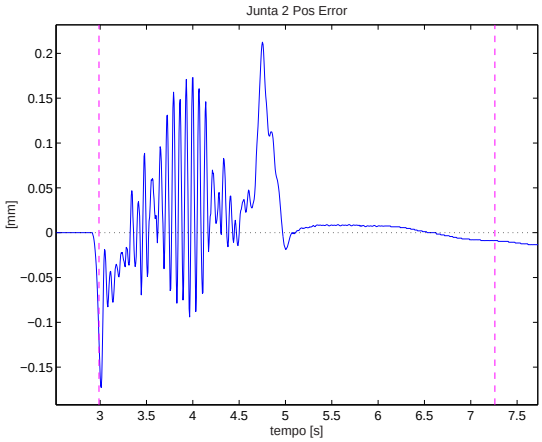
(a) PD de ”fábrica”.



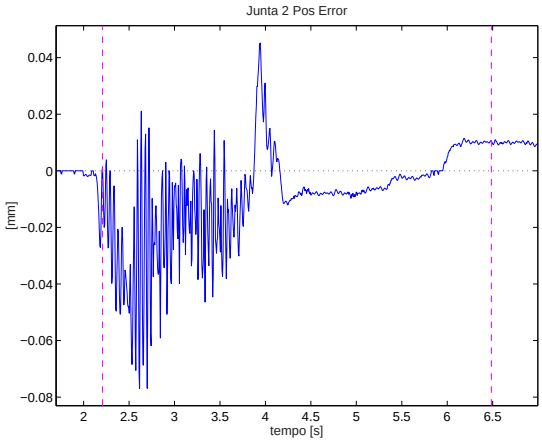
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

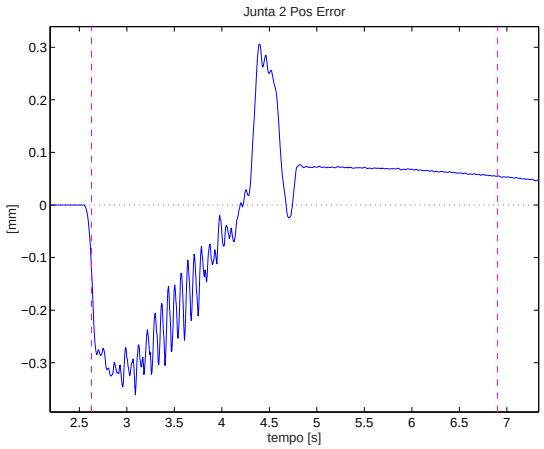


(d) PD + MLP2c (η=0.01, α=0.7).

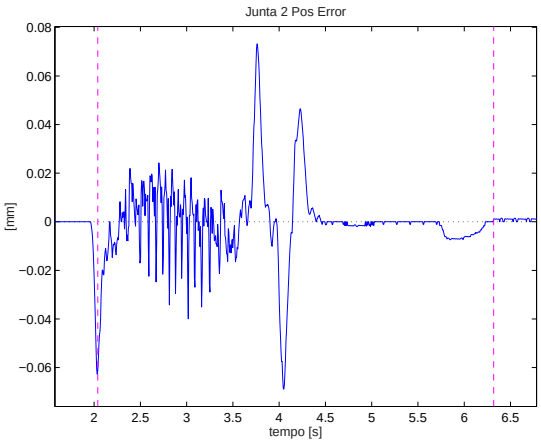


(e) PID + MLP2c (η=0.035, α=0.7).

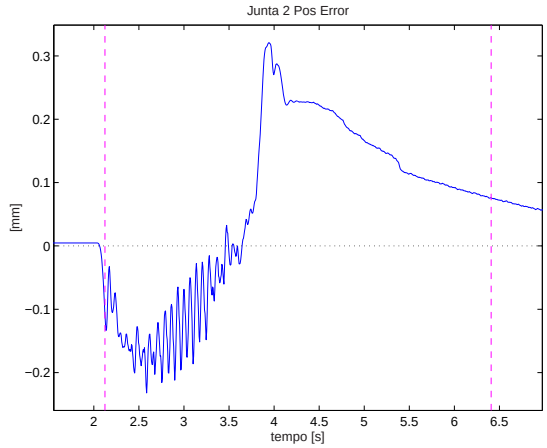
Figura 5.17: Erros de seguimento de trajetória para a Junta 2 (Movimento ”B”).



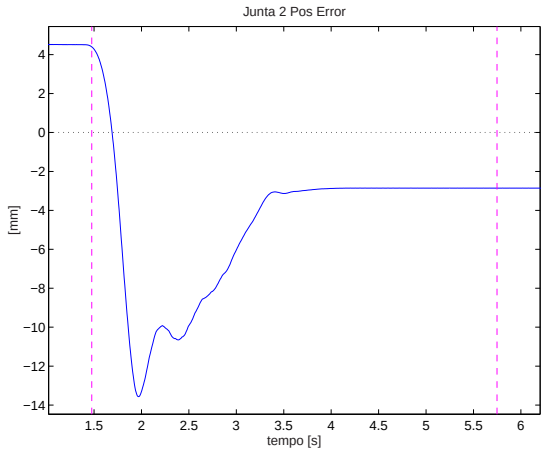
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

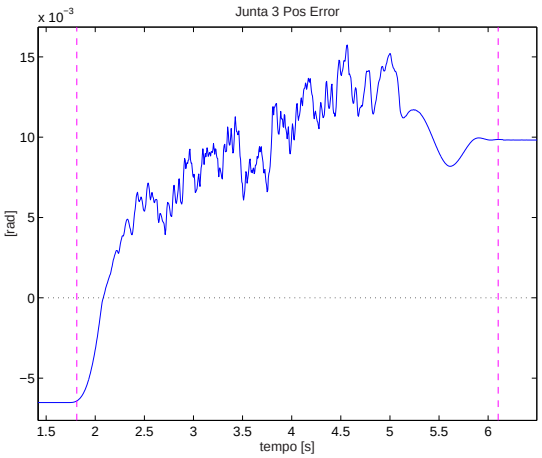


(h) PD + RBF(7) (η=0.0001, α=0.5).

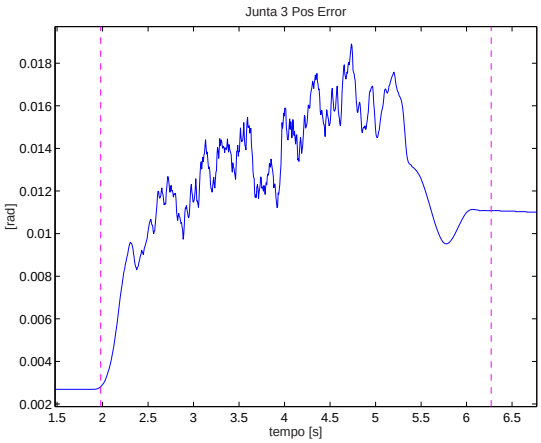


(i) Modos Deslizantes.

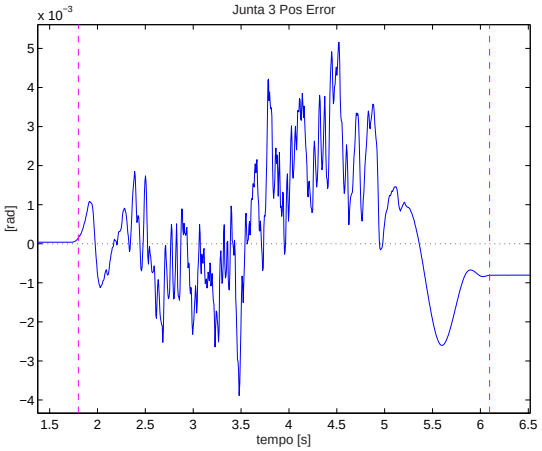
Figura 5.17: (Cont.) Erros de seguimento de trajetória para a Junta 2 (Movimento ”B”).



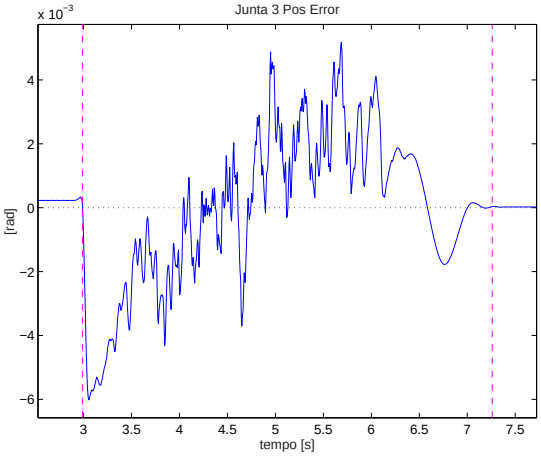
(a) PD de ”fábrica”.



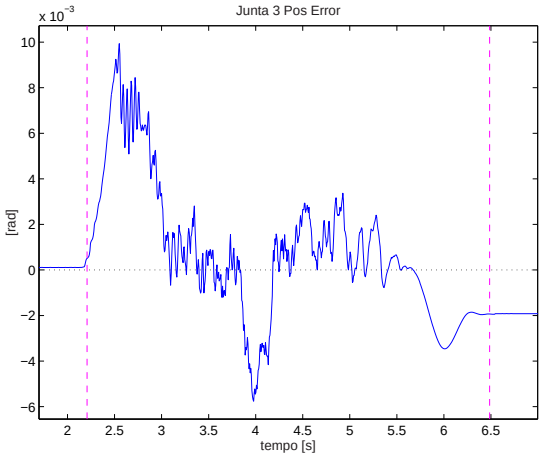
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

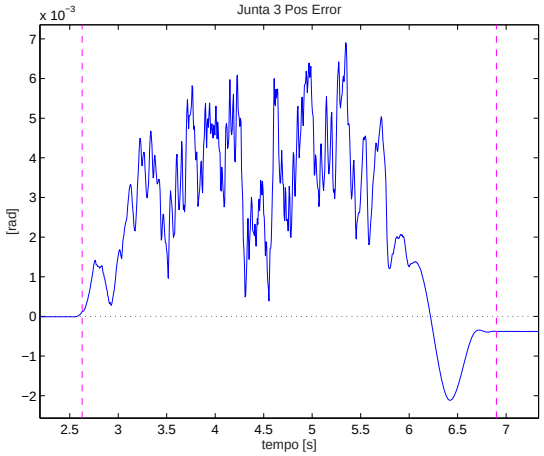


(d) PD + MLP2c (η=0.01, α=0.7).

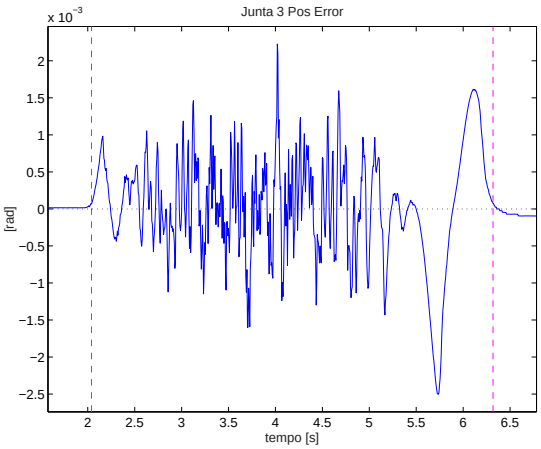


(e) PID + MLP2c (η=0.035, α=0.7).

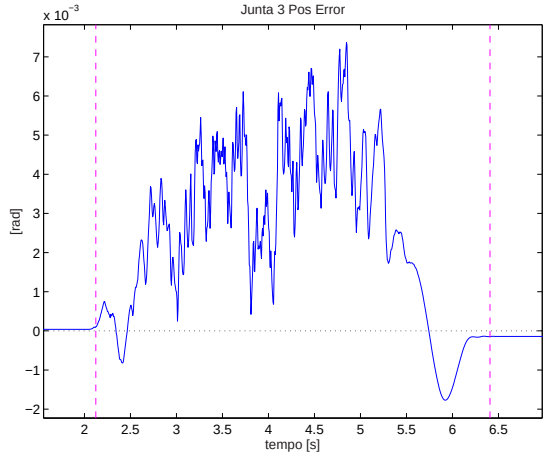
Figura 5.18: Erros de seguimento de trajetória para a Junta 3 (Movimento ”B”).



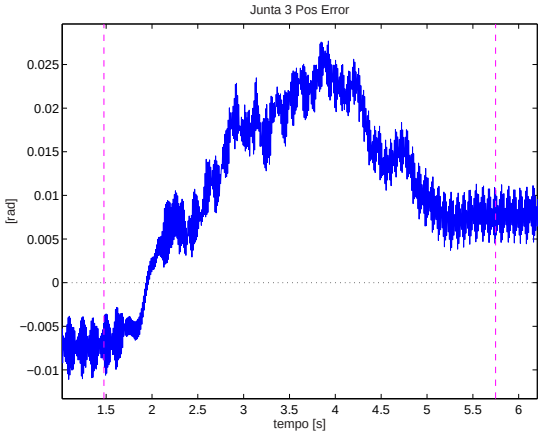
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).



(h) PD + RBF(7) (η=0.0001, α=0.5).

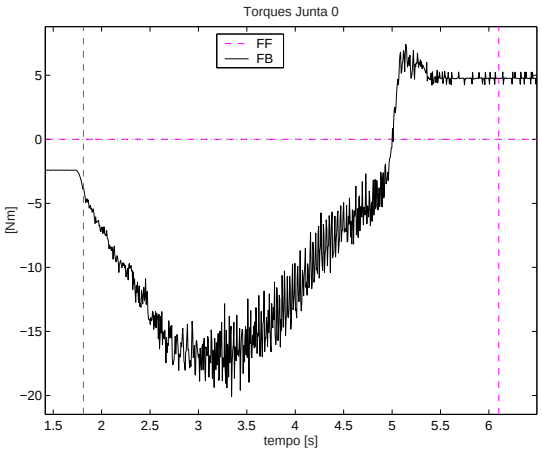


(i) Modos Deslizantes.

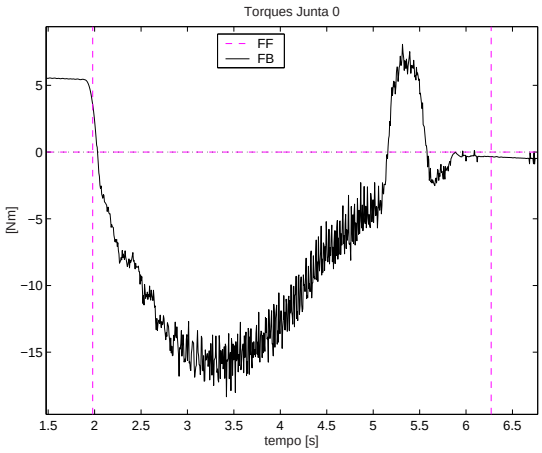
Figura 5.18: (Cont.) Erros de seguimento de trajetória para a Junta 3 (Movimento ”B”).

Gráficos dos Torques Desenvolvidos

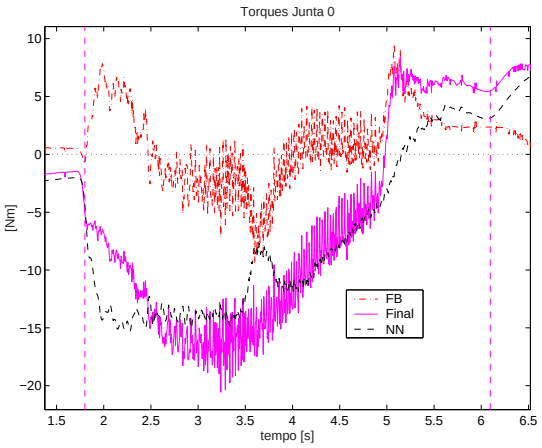
A figura 5.19 mostra os erros de seguimento de trajetória para primeira junta (Junta 0) e as figuras 5.20, 5.21 e 5.22, para as juntas 1, 2 e 3 respectivamente.



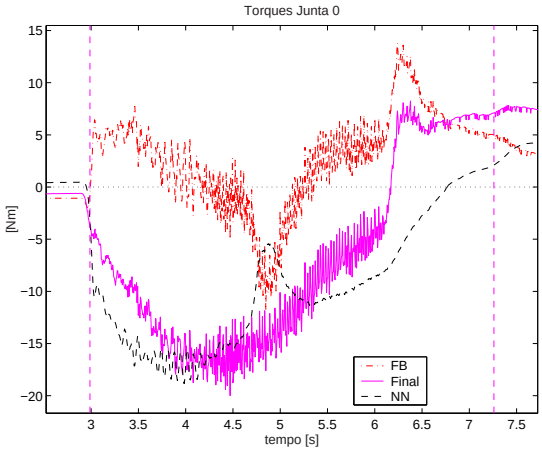
(a) PD de ”fábrica”.



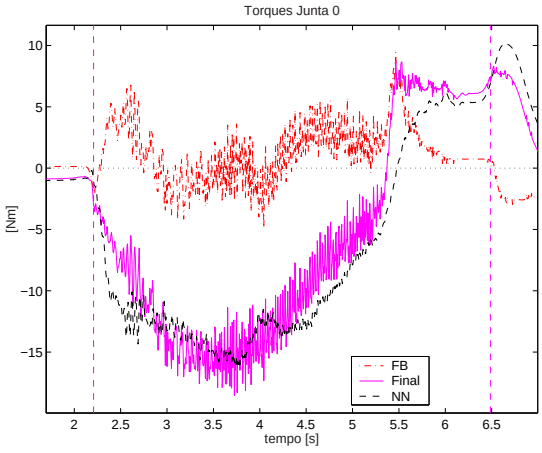
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

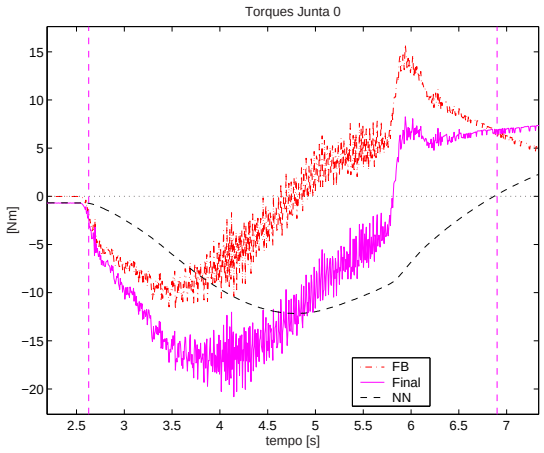


(d) PD + MLP2c (η=0.01, α=0.7).

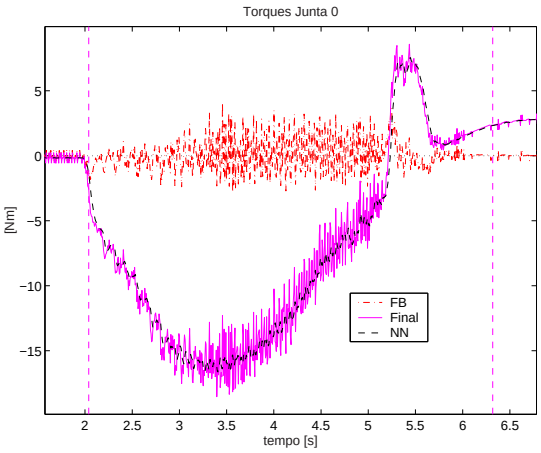


(e) PID + MLP2c (η=0.035, α=0.7).

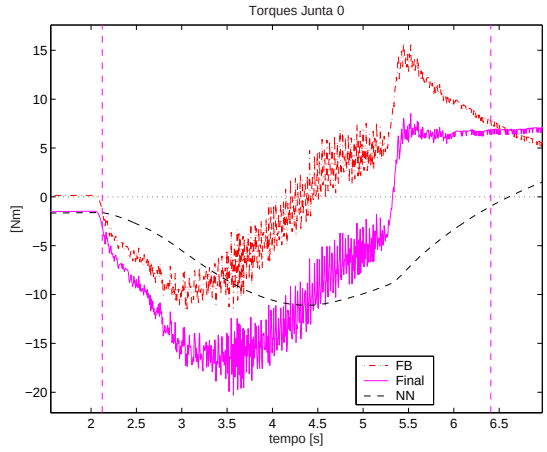
Figura 5.19: Torques enviados à junta 0 (Movimento ”B”).



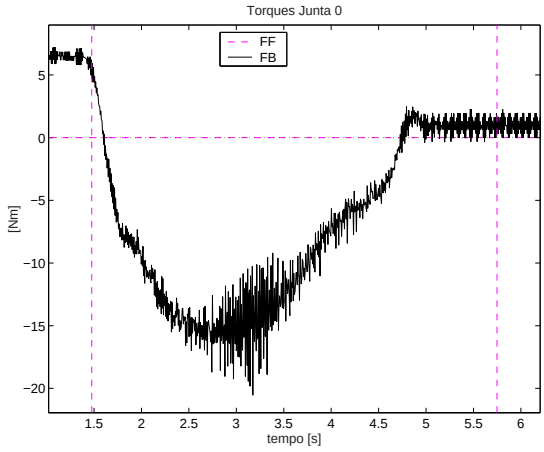
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

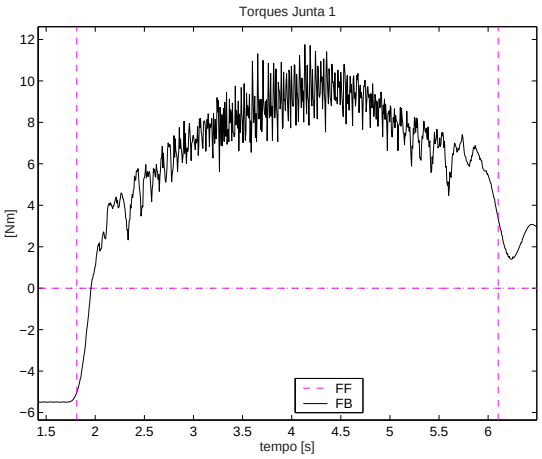


(h) PD + RBF(7) (η=0.0001, α=0.5).

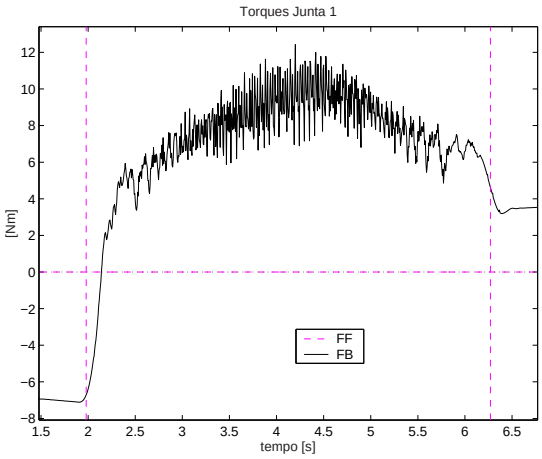


(i) Modos Deslizantes.

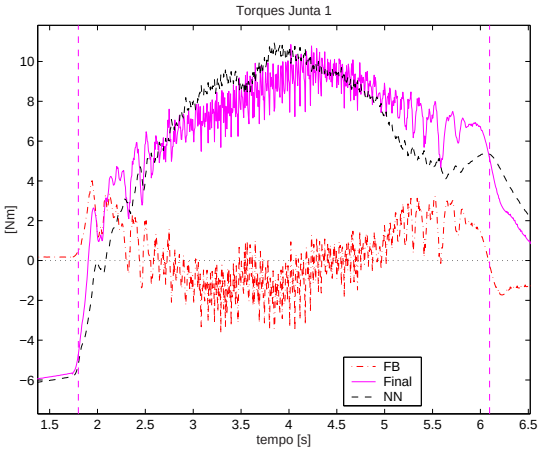
Figura 5.19: (Cont.) Torques enviados à junta 0 (Movimento ”B”).



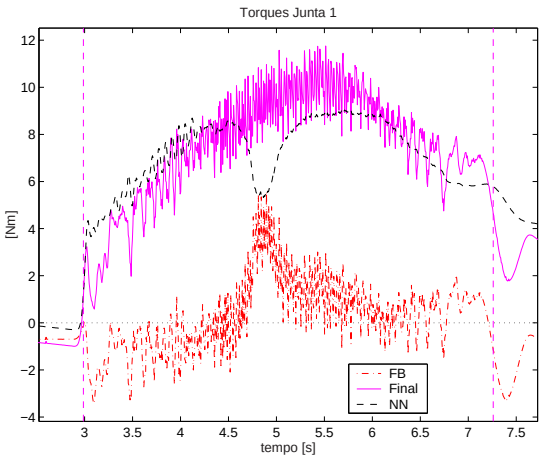
(a) PD de ″fábrica″.



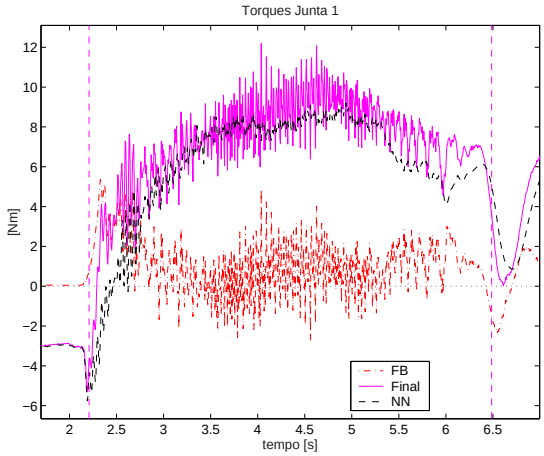
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

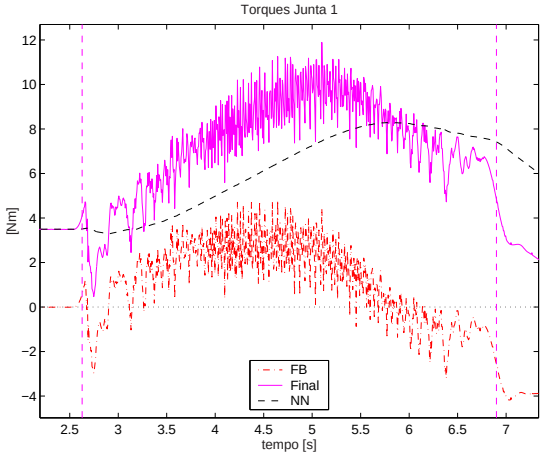


(d) PD + MLP2c (η=0.01, α=0.7).

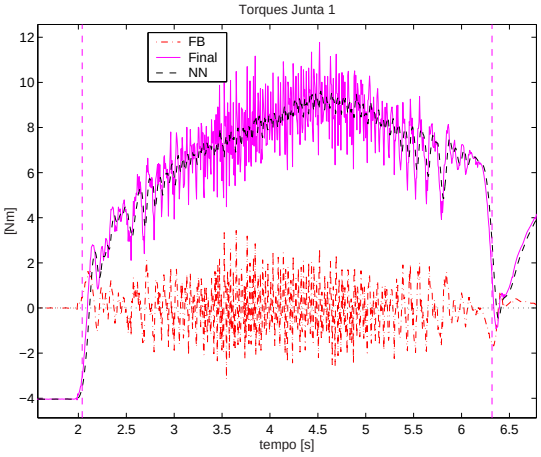


(e) PID + MLP2c (η=0.035, α=0.7).

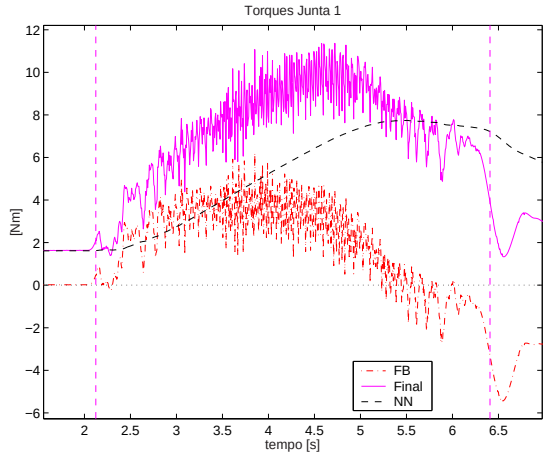
Figura 5.20: Torques enviados à junta 1 (Movimento ″B″).



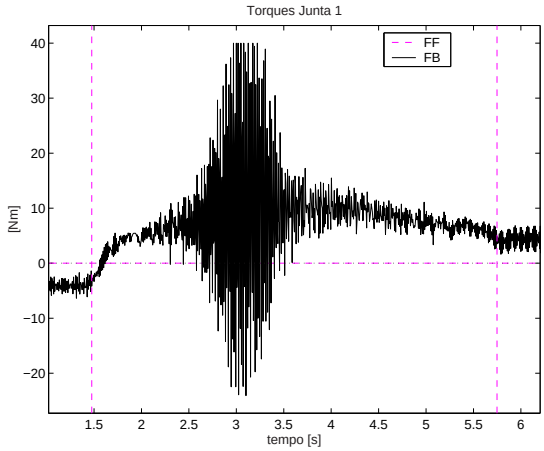
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

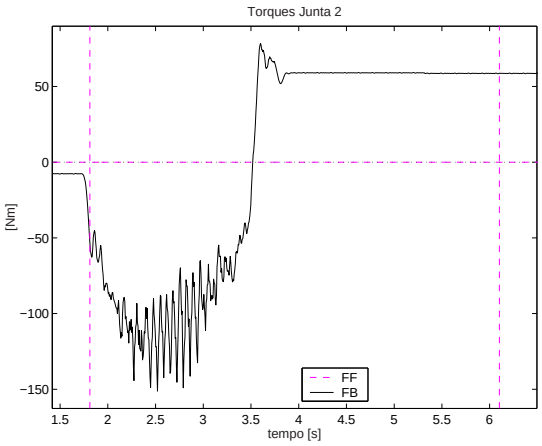


(h) PD + RBF(7) (η=0.0001, α=0.5).

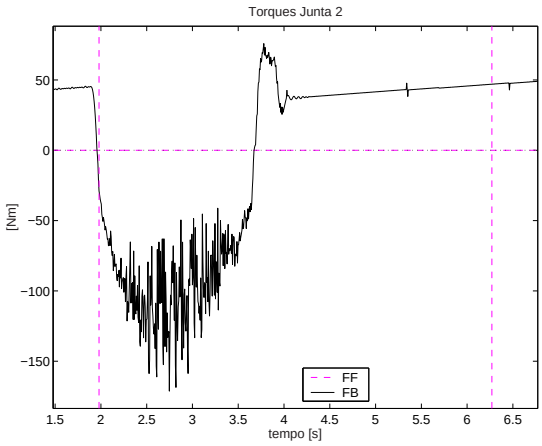


(i) Modos Deslizantes.

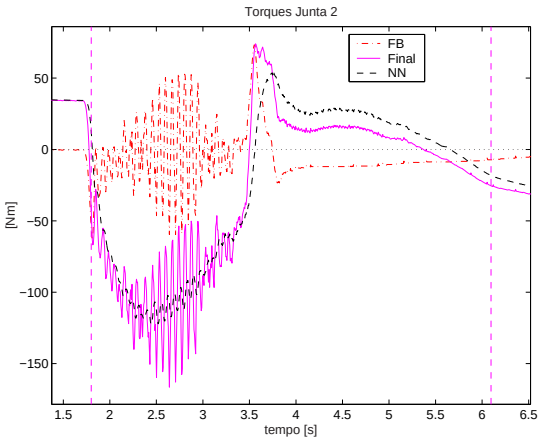
Figura 5.20: (Cont.) Torques enviados à junta 1 (Movimento ”B”).



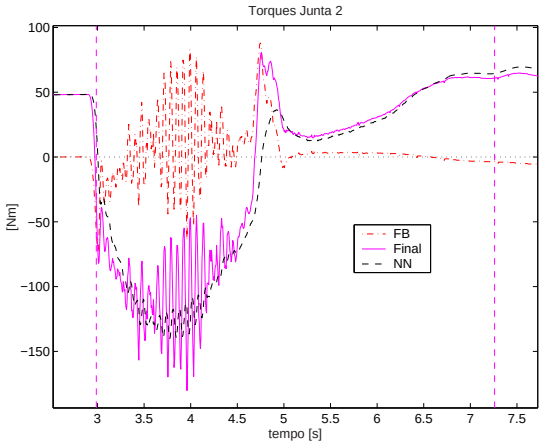
(a) PD de ″fábrica″.



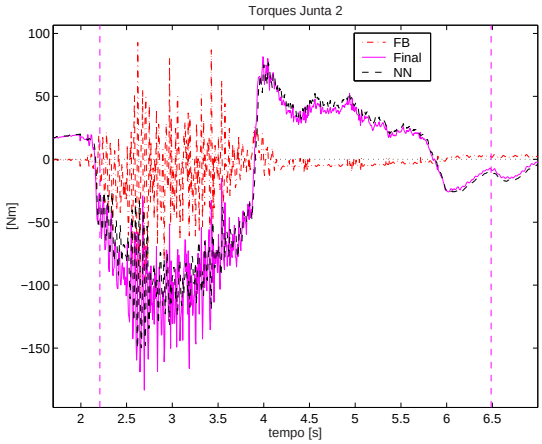
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

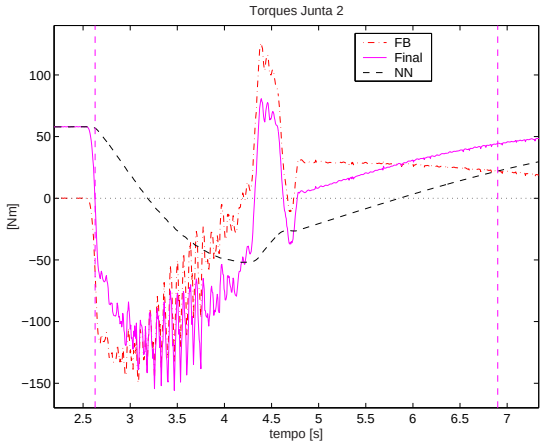


(d) PD + MLP2c (η=0.01, α=0.7).

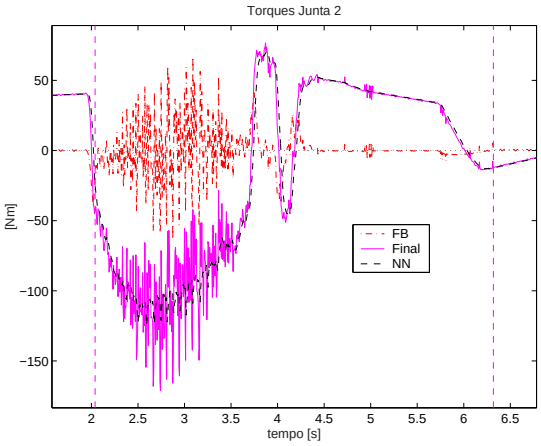


(e) PID + MLP2c (η=0.035, α=0.7).

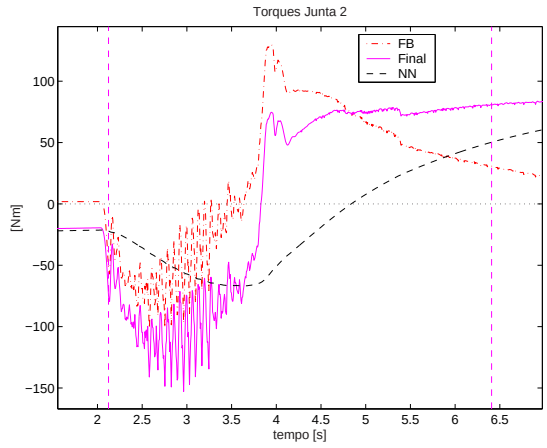
Figura 5.21: Torques enviados à junta 2 (Movimento ″B″).



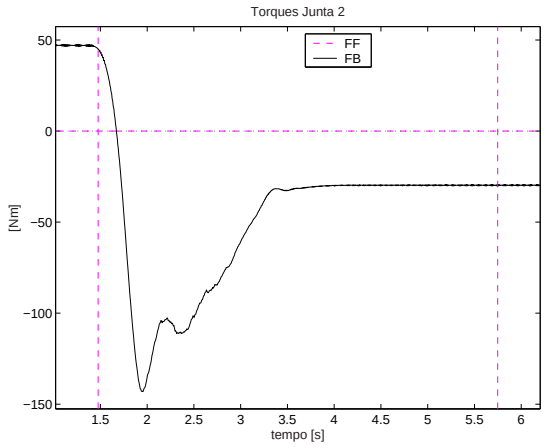
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).

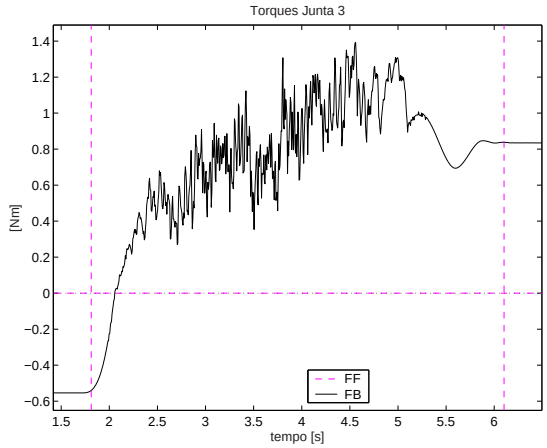


(h) PD + RBF(7) (η=0.0001, α=0.5).

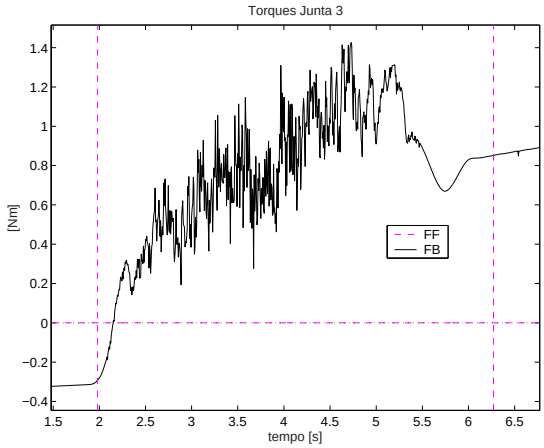


(i) Modos Deslizantes.

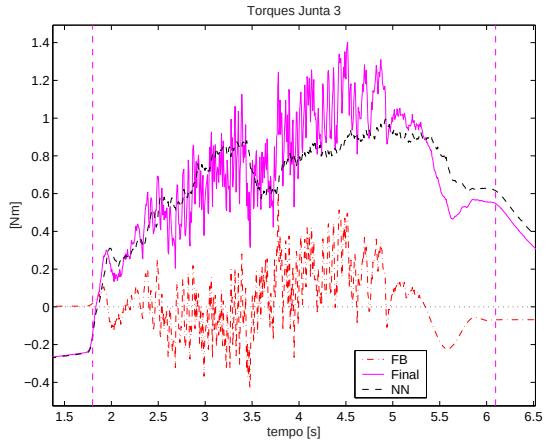
Figura 5.21: (Cont.) Torques enviados à junta 2 (Movimento ”B”).



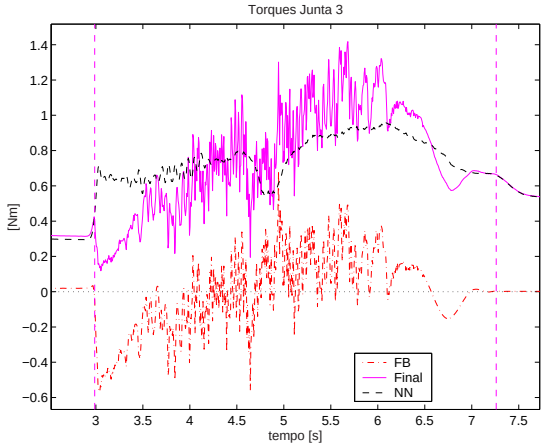
(a) PD de ”fábrica”.



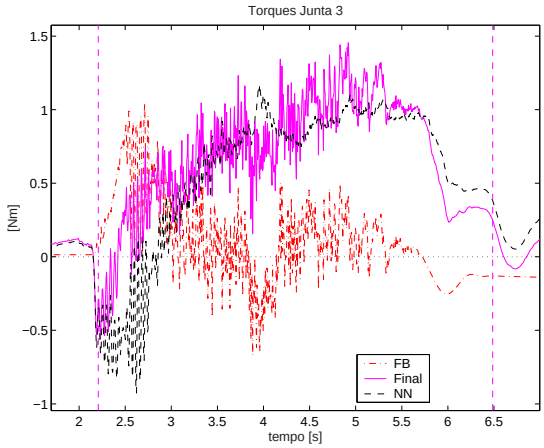
(b) PID.



(c) PD + MLP1c (η=0.01, α=0.7).

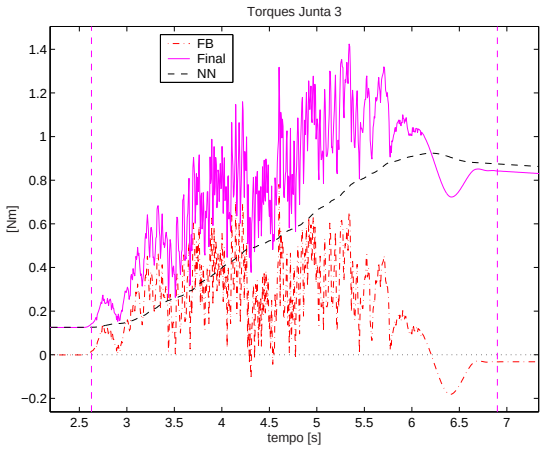


(d) PD + MLP2c (η=0.01, α=0.7).

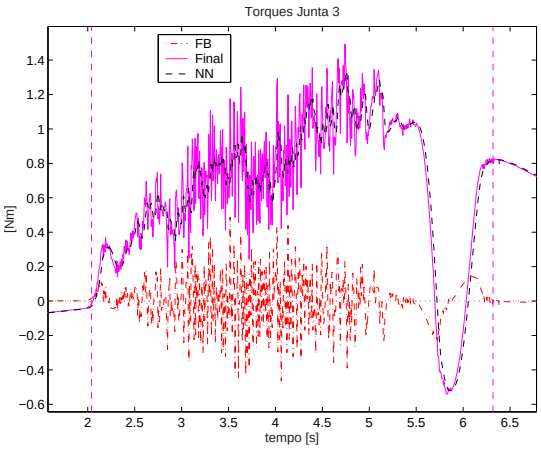


(e) PID + MLP2c (η=0.035, α=0.7).

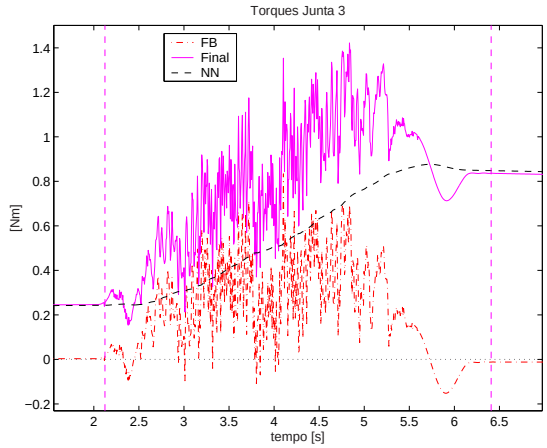
Figura 5.22: Torques enviados à junta 3 (Movimento ”B”).



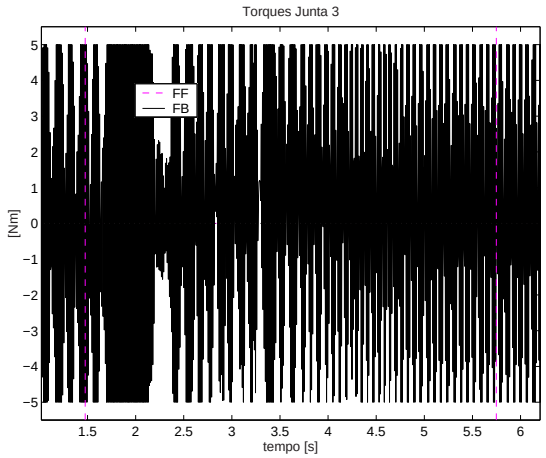
(f) PD + RBF(5) (η=0.0001, α=0.5).



(g) PID + RBF(5) (η=0.005, α=0.5).



(h) PD + RBF(7) (η=0.0001, α=0.5).



(i) Modos Deslizantes.

Figura 5.22: (Cont.) Torques enviados à junta 3 (Movimento "B").

Índices de Desempenho

R1) $\tilde{q}_f$**: Erros de Posicionamento final**: tabela 5.15.

R2) $MAX\{\tilde{q}\}$**: Erros Máximos de Seguimento de Trajetória**: tabela 5.16.

R3) $IAE\{\tilde{q}\}$**: Integral dos Erros Máximos de Seguimento de trajetória**: tabela 5.17.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	571.02μ	1.57m	144.76μ	9.86m
PID	-418.55μ	1.80m	116.83μ	11.08m
PD + MLP1c (η=0.01, α=0.7)	-487.07μ	440.77μ	-39.25μ	922.46μ
PD + MLP2c (η=0.01, α=0.7)	601.84μ	-300.61μ	-8.84μ	33.42μ
PID + MLP2c (η=0.035, α=0.7)	15.17μ	-501.50μ	10.55μ	-1.93m
PD + RBF(5) (η=0.0001, α=0.5)	808.06μ	-917.54μ	54.80μ	-378.65μ
PID + RBF(5) (η=0.005, α=0.5)	3.60μ	-499.28μ	1.16μ	75.06μ
PD + RBF(7) (η=0.0001, α=0.5)	939.31μ	-1.18m	76.15μ	-144.90μ
Modos Deslizantes	574.33μ	4.01m	-2.86m	10.61m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.15: Erros de posicionamento final, $\tilde{q}_b$, encontrados para os diferentes controladores

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	-2.04m	4.17m	-339.99μ	15.74m
PID	-2.65m	4.49m	-294.36μ	18.91m
PD + MLP1c (η=0.01, α=0.7)	933.73μ	1.52m	178.42μ	5.16m
PD + MLP2c (η=0.01, α=0.7)	1.49m	1.70m	212.52μ	-6.02m
PID + MLP2c (η=0.035, α=0.7)	896.58μ	1.94m	-76.92μ	9.93m
PD + RBF(5) (η=0.0001, α=0.5)	1.72m	1.31m	-360.37μ	6.90m
PID + RBF(5) (η=0.005, α=0.5)	311.12μ	575.58μ	73.24μ	-2.50m
PD + RBF(7) (η=0.0001, α=0.5)	1.71m	1.74m	320.97μ	7.36m
Modos Deslizantes	-9.26m	8.88m	-13.57m	27.69m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.16: Erros máximos de seguimento de trajetória, $MAX\{\tilde{q}\}$, encontrados para os diferentes controladores

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	835.67m	2.09	125.07m	6.50
PID	1.09	2.45	109.51m	9.05
PD + MLP1c (η=0.01, α=0.7)	239.92m	362.30m	24.90m	985.76m
PD + MLP2c (η=0.01, α=0.7)	373.78m	330.13m	21.82m	1.33
PID + MLP2c (η=0.035, α=0.7)	149.44m	292.39m	9.58m	1.60
PD + RBF(5) (η=0.0001, α=0.5)	558.54m	498.28m	89.97m	1.98
PID + RBF(5) (η=0.005, α=0.5)	26.09m	80.45m	6.44m	395.41m
PD + RBF(7) (η=0.0001, α=0.5)	576.39m	678.83m	99.51m	1.95
Modos Deslizantes	3.36	4.08	3.52	9.22
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.17: Soma absoluta dos erros de seguimento de trajetória, $IAE\{\tilde{q}\}$, encontrados para os diferentes controladores

R4) $MED\{\tilde{q}\}$**: Erros Médios de Seguimento de trajetória**: tabela 5.18.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	-871.34μ	2.81m	-296.16n	8.55m
PID	-1.47m	3.34m	-6.20μ	12.63m
PD + MLP1c (η=0.01, α=0.7)	99.88μ	182.59μ	-13.05μ	373.34μ
PD + MLP2c (η=0.01, α=0.7)	330.76μ	125.56μ	13.67μ	-60.25μ
PID + MLP2c (η=0.035, α=0.7)	125.37μ	374.25μ	-8.59μ	764.81μ
PD + RBF(5) (η=0.0001, α=0.5)	24.74μ	453.49μ	-24.09μ	2.45m
PID + RBF(5) (η=0.005, α=0.5)	2.76μ	11.13μ	-14.40n	39.45μ
PD + RBF(7) (η=0.0001, α=0.5)	22.58μ	806.15μ	61.94μ	2.47m
Modos Deslizantes	-4.24m	5.57m	-4.64m	11.75m
Unidades	[rad]	[rad]	[m]	[rad]

Tabela 5.18: Erros médios de seguimento de trajetória, $MED\{\tilde{q}\}$, encontrados para os diferentes controladores

R5) $MSE\{\tilde{q}\}$**: Erros Quadráticos Médios de Seguimento de trajetória**: tabela 5.19.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	1.69μ	9.29μ	33.55n	93.57μ
PID	2.99μ	12.44μ	26.73n	169.60μ
PD + MLP1c (η=0.01, α=0.7)	168.19n	366.73n	1.97n	3.00μ
PD + MLP2c (η=0.01, α=0.7)	394.56n	337.03n	2.70n	5.57μ
PID + MLP2c (η=0.035, α=0.7)	73.26n	308.24n	$<Under>$	9.57μ
PD + RBF(5) (η=0.0001, α=0.5)	762.90n	624.42n	25.19n	10.50μ
PID + RBF(5) (η=0.005, α=0.5)	3.20n	23.16n	$<Under>$	546.80n
PD + RBF(7) (η=0.0001, α=0.5)	821.84n	1.21μ	24.27n	10.88μ
Modos Deslizantes	32.70μ	36.61μ	34.64μ	215.60μ
Unidades	[rad^2]	[rad^2]	[m^2]	[rad^2]

Tabela 5.19: Erros médios quadráticos de seguimento de trajetória, $MSE\{\tilde{q}\}$, encontrados para os diferentes controladores

R6) $VAR\{\tilde{q}\}$**: Variância dos Erros de Seguimento de trajetória**: tabela 5.20.

R7) $MAX\{\tau\}$**: Maior valor da ação de controle**: tabela 5.21.

R8) $MED\{\tau\}$**: Valor Médio da ação de controle**: tabela 5.22.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	930.87n	1.39μ	33.60n	20.49μ
PID	833.30n	1.29μ	26.73n	9.98μ
PD + MLP1c (η=0.01, α=0.7)	158.44n	333.86n	1.80n	2.87μ
PD + MLP2c (η=0.01, α=0.7)	285.56n	321.72n	2.52n	5.58μ
PID + MLP2c (η=0.035, α=0.7)	57.62n	168.41n	$<Under>$	9.00μ
PD + RBF(5) (η=0.0001, α=0.5)	763.36n	419.35n	24.64n	4.48μ
PID + RBF(5) (η=0.005, α=0.5)	3.20n	23.07n	$<Under>$	546.01n
PD + RBF(7) (η=0.0001, α=0.5)	822.48n	557.79n	20.47n	4.76μ
Modos Deslizantes	14.71μ	5.61μ	13.14μ	77.60μ
Unidades	[rad^2]	[rad^2]	[m^2]	[rad^2]

Tabela 5.20: Variância dos erros de seguimento de trajetória, $VAR\{\tilde{q}\}$, encontrados para os diferentes controladores

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	-20.10	11.52	-154.71	1.40
PID	-17.46	11.87	-167.14	1.43
PD + MLP1c (η=0.01, α=0.7)	-20.57	10.85	-166.54	1.40
PD + MLP2c (η=0.01, α=0.7)	-19.99	11.75	-180.28	1.42
PID + MLP2c (η=0.035, α=0.7)	-18.53	12.20	-183.39	1.46
PD + RBF(5) (η=0.0001, α=0.5)	-20.80	11.89	-155.80	1.42
PID + RBF(5) (η=0.005, α=0.5)	-18.54	11.77	-171.05	1.49
PD + RBF(7) (η=0.0001, α=0.5)	-20.30	11.36	-152.90	1.42
Modos Deslizantes	-20.54	40.00	-143.03	5.00
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.21: Maiores ações de controle, $MAX\{\tau\}$, encontrados para os diferentes controladores

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	-7.23	6.81	319.90m	732.20m
PID	-7.20	7.10	-8.86	750.09m
PD + MLP1c (η=0.01, α=0.7)	-6.82	6.90	-28.40	718.05m
PD + MLP2c (η=0.01, α=0.7)	-6.76	7.47	-11.44	752.03m
PID + MLP2c (η=0.035, α=0.7)	-6.11	7.16	-21.19	680.55m
PD + RBF(5) (η=0.0001, α=0.5)	-6.97	7.29	-19.42	781.84m
PID + RBF(5) (η=0.005, α=0.5)	-7.27	6.79	-15.73	660.74m
PD + RBF(7) (η=0.0001, α=0.5)	-6.64	7.24	8.53	783.65m
Modos Deslizantes	-7.20	7.62	-48.69	290.12m
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.22: Valores médios das ações de controle, $MED\{\tau\}$, encontrados para os diferentes controladores

R9) $VARi\{\tau\}$**: Variação Média da ação de controle**: tabela 5.23.

Controlador	Junta 0	Junta 1	Junta 2	Junta 3
PD de fábrica	1.07	609.76m	2.94	55.53m
PID	917.88m	814.14m	7.96	75.42m
PD + MLP1c (η=0.01, α=0.7)	1.26	508.81m	4.20	61.26m
PD + MLP2c (η=0.01, α=0.7)	1.18	624.14m	4.19	59.96m
PID + MLP2c (η=0.035, α=0.7)	1.20	864.92m	9.02	104.45m
PD + RBF(5) (η=0.0001, α=0.5)	1.26	665.03m	3.50	62.32m
PID + RBF(5) (η=0.005, α=0.5)	1.03	891.05m	8.80	101.78m
PD + RBF(7) (η=0.0001, α=0.5)	1.29	645.49m	3.53	64.73m
Modos Deslizantes	1.58	7.97	686.09m	7.33
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 5.23: Variações médias das ações de controle, $VARi\{\tau\}$, encontrados para os diferentes controladores

Conclusões preliminares

As seguintes conclusões podem ser extraídas dos testes realizados anteriormente:

- Novamente, como demonstrando no ensaio anterior a este (envolvendo uma trajetória operando sobre uma região diferente do espaço de trabalho do robô), o controlador que obteve os melhores resultados foi o integrado PID + RBF(5), com $\eta = 0.005$ e $\alpha = 0.5$ – ver figuras 5.15(g) à 5.18(g).

- Apesar do controlador PID + MLP2c não conseguir atingir resultados tão bons quanto os alcançados pelo controlador PID + RBF(5), figura 5.17(e), é perfeitamente nítido o impacto gerado pela ação integral do controlador convencional quando operando em paralelo com uma rede neural. O controlador convencional que melhor se adapta para uso com redes neurais na arquitetura proposta de aprendizagem por realimentação dos erros é nitidamente o PID devido à sua ação integral.

- Ainda, comparando-se as figuras 5.15(e) (referente a um PID + MLP2c) e a figura 5.15(g) (referente ao PID + RBF(5)) se percebe que o controlador com a rede RBF consegue erros de seguimento da trajetória 50% menores e é nítido que rede RBF trabalha melhor com o controlador PID porque consegue manter durante a maior parte do tempo, os erros de seguimento de trajetória baixos e oscilando em torno de zero.

- No que se refere à taxas de aprendizado usados para as redes, as figura 5.18(e) (PID + MLP2c com $\eta = 0.035$ e $\alpha = 0.7$) e figura 5.18(d) (PID + MLP2c com $\eta = 0.01$ e $\alpha = 0.7$), é nítido que uma taxa de aprendizado maior deixa a rede mais rápida, mas também mais oscilatória, gerando picos maiores de erros de seguimento de trajetória, tabela 5.16. Este comportamento também é nítido nas ações de controle geradas para a junta 3 por exemplo, figuras 5.22(d) e (e). Note que a última junta do robô, a 3, responsável por definir a orientação do efetuador acoplado ao robô é também a mais rápida do sistema, com as menores constantes de tempo.

- De forma semelhante, se percebe que uma taxa de aprendizado como $\eta = 0.0001$ é inadequada para a rede RBF, figura 5.21(f), se comparada com $\eta = 0.005$, figura 5.21(g). Na figura 5.21(f) se percebe que o valor predominante do torque gerado para a junta 2 ficou à cargo do controlador PD e que a rede RBF reagiu muito aquém do esperado, o que é o oposto do que se verifica na figura 5.21(g)($\eta = 0.005$).

- Fica mais evidente que o PID ainda está com ganhos elevados para a junta 0, conforme pode ser visualizado na figura 5.19(b), cujos erros de seguimento de trajetória ficaram acima dos registrados para o PD de "fábrica" – figura 5.15(a). Notar a dificuldade em sintonizar os controladores do robô. Os ganhos variam conforme a região de trabalho do robô.

- Comparando o desempenho do controlador PD + MLP1c (rede MLP com apenas uma camada invisível) e do controlador PD + MLP2c (rede com 2 camadas invisíveis) se percebe um comportamento algo semelhante, mas a rede com apenas 1 camada invisível reage mais rápido à mudanças de configuração do robô e ainda é capaz de garantir erros de seguimento de trajetória menores do que os verificados com a rede com 2 camadas invisíveis – perceptível pelas figuras 5.15(c) e (d) e figuras 5.16(c) e (d). O que chega a ser surpreendente. Parece não ser de forma alguma vantajosa trabalhar com 2 camadas invisíveis pois além de tornar a rede mais "lenta" ainda exige maior poder de processamento da CPU do robô. Pela tabela 5.15 se percebe que a rede MLP com 2 camadas pode até garantir erros menores no instante final da execução da trajetória, mas não garante os menores erros durante o seguimento da trajetória, tabelas: 5.16($MAX\{\tilde{q}\}$), 5.19($MSE\{\tilde{q}\}$). O que sugere ser mais conveniente do ponto de vista

da relação custo $\times$ benefício, trabalhar com uma rede MLP contendo apenas uma cada invisível que exige quase 30% à menos de poder de processamento da CPU do robô, conforme pode ser deduzido à partir da tabela 5.24.

Rede	PEs	Modo "*forward*"	Modo "*backward*"	Total
MLP1c	$12 \times 8 \times 4$	152	704	856
MLP2c	$12 \times 8 \times 5 \times 4$	190	920	1110
RBF(5)	5 funções gaussinas	244	1008	1252
RBF(7)	7 funções gaussianas	340	1392	1732

Tabela 5.24: Operações em ponto flutuante necessárias para processar as redes [Flops].

- Comparando-se novamente o desempenho do controlador PD com uma rede RBF com 5 ou 7 funções gaussinas (figuras 5.16(f) e (h) respectivamente) se percebe um comportamento muito semelhante. Apenas para o caso da junta 2 (a prismática), figuras 5.17(f) e (h) se percebe que o PD com a rede RBF contendo 7 funções gaussianas na sua camada intermediária, garante erros de seguimento de trajetória ligeiramente menores mas com uma ação de controle ligeiramente mais oscilatória (figuras 5.21(f) e (h) e tabela 5.23 contendo os índices de desempenho relacionados com $VAR\{\tau\}$). Mais uma vez se percebe que a relação custo $\times$ benefício não compensa o acréscimo de mais funções gaussianas no controlador integrado baseado na rede RBF.

- Com respeito ao controlador por modos deslizantes, se confirma seu melhor ajuste para a juntas 2 (a prismática), figura 5.17(i) relativa aos erros de seguimento de trajetória e figura 5.21(i) referente ao torque enviado a esta junta. Mais uma vez, foi o controlador que apresentou a ação de controle menos oscilatória, tabela 5.23. Para esta junta, a 2, este controlador permitiu alcançar resultados superiores aos encontrados para o PD de "fábrica", mas inferior ao PID com relação a valor médio acumulado de erro quadrático de seguimento de trajetória ($MSE\{\tilde{q}\}$), tabela 5.19. De qualquer forma, fica evidenciado a dificuldade relacionada com o ajuste dos parâmetros deste tipo de controlador.

- O controlador baseado em modos deslizantes é o que permite erro de seguimento menos oscilatório, porém oscilando entre faixas maiores de valor – figura 5.16(i) se comparado aos outros controladores.

Observação: Pode ser percebido em alguns gráficos relacionados com torques enviados às juntas 0 e 1 que mesmo que o gerador de trajetórias esteja numa condição de "repouso" (velocidades e acelerações desejadas nulas), ainda assim, o torque enviado àquela junta não é nulo. Isto se deve à própria construção física do robô. O cabo que alimenta os motores é montado através de um junta fixa rígida sobre a segunda junta do robô conforme mostra a figura 5.23. O efeito do cabo sobre as duas primeiras juntas é tratado então como uma perturbação pelos próprios controladores, o que justifica o fato de que para certas configurações finais do robô e com o robô em repouso, ainda estar sendo gerado um torque não nulo para as duas primeiras juntas.

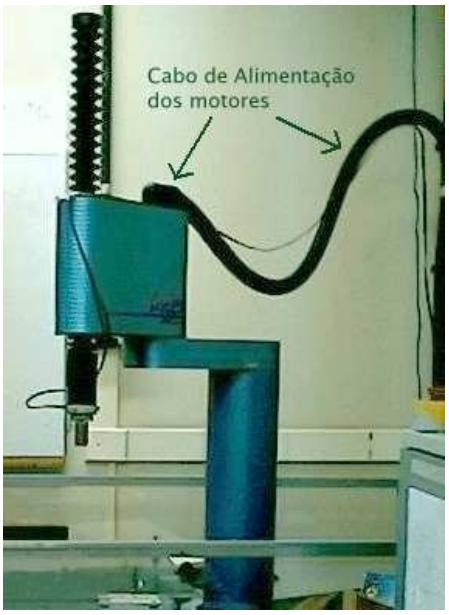


Figura 5.23: Imagem ressaltando fixação dos cabos de alimentação dos motores do robô Inter/Scara.

5.6 Ensaio de Rejeição à Perturbação

A figura 5.24 mostra a cinta elástica usada como elemento perturbados para os controladores de posição sendo testados.

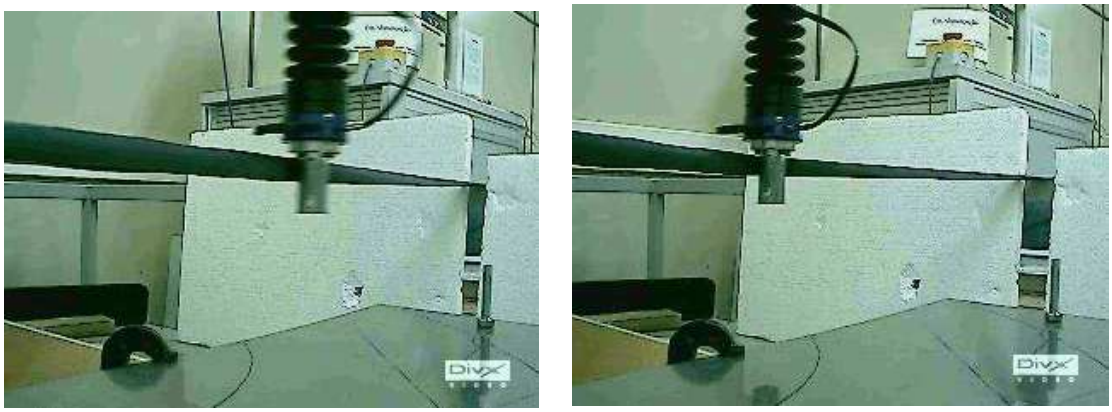
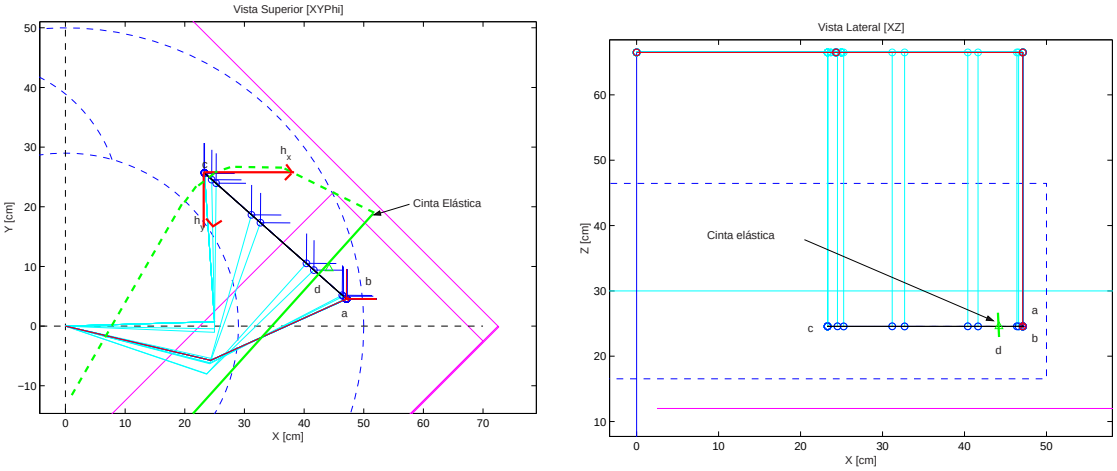


Figura 5.24: Trajetória executada com a perturbação presente.

Dados do Ensaio com Perturbação

O perfil da trajetória com a perturbação presente é mostrada na figura 5.25. Os comandos utilizados para gerar esta trajetória foram especificados no espaço operacional com movimentos indo dos pontos de: (a) → (c) → (b) com uma pequena pausa no ponto (c) (novamente foi utilizado o gerador de trajetórias original do robô). Os dados com respeito a movimentação realizada e forças e momentos gerados sobre o efetuador final do robô por ocasião da cinta elástica são mostrados na tabela 5.6. Os perfis de velocidades e acelerações gerados no espaço operacional podem ser vistos na figura 5.26, o que resultou no perfil no espaço de juntas mostrado na figura 5.27 (estes são os dados usados pelas redes neurais).



(a) Vista XYθ (b) Vista XZ

Figura 5.25: Trajetória executada com a perturbação presente.

Dados	X	Y	Z	Orientação (θ)
$x_a = x_b$	47.12 [cm]	4.54 [cm]	0.2456 [cm]	0.00 [^{o}]
x_c	23.26 [cm]	25.71 [cm]	0.2456 [cm]	0.00 [^{o}]
$\dot{x}_{máx}$	22.15 [cm/s]	19.66 [cm/s]	0.02 [cm/s]	0.00 [$^o/s$]
$\ddot{x}_{máx}$	41.95 [cm/s^2]	37.01 [m/s^2]	2.17 [cm/s^2]	0.16 [^{o}/s^2]
x_d	44.21 [cm]	9.65 [cm]	24.56 [cm]	
$f_{Sens_{máx}}$	30.29 [N]	-18.55 [N]	-1.44 [N]	
$\mu_{Sens_{máx}}$	-0.73 [Nm]	-1.28 [Nm]	0.21 [Nm]	
$h_{máx}$	30.30 [N]	-18.52 [N]	1.44 [N]	

Tabela 5.25: Dados da trajetória executada com a Perturbação presente (no espaço operacional).

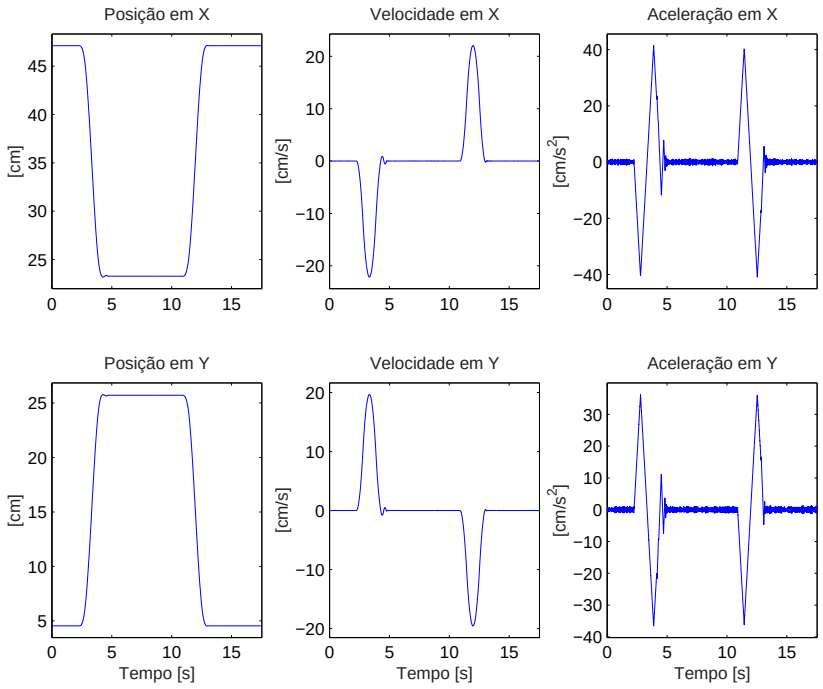


Figura 5.26: Perfis da trajetória com a perturbação presente (no espaço operacional).

Tabela resumo dos testes realizados

A tabela 5.26 (pág. 196) resume os dados relativos aos ensaios que serão apresentados.

Gráficos dos Erros de Posicionamento

A figura 5.28 (pág. 197) mostra os erros de seguimento de trajetória verificados em XY. É determinado da seguinte maneira: $\tilde{d} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$, onde $\tilde{x} = x - x_d$ e $\tilde{y} = y - y_d$.

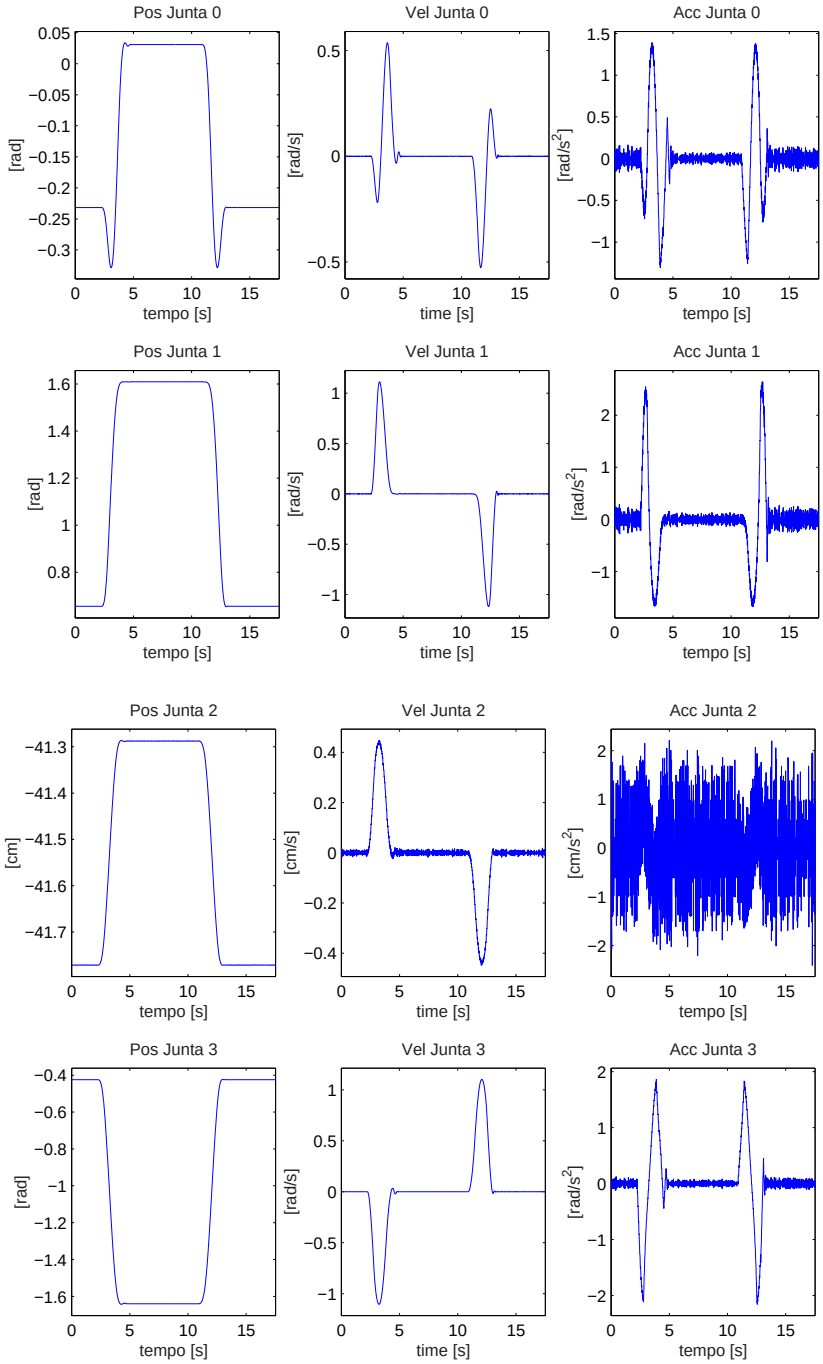
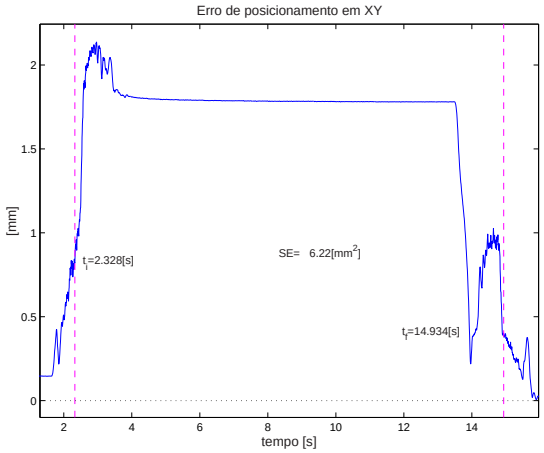


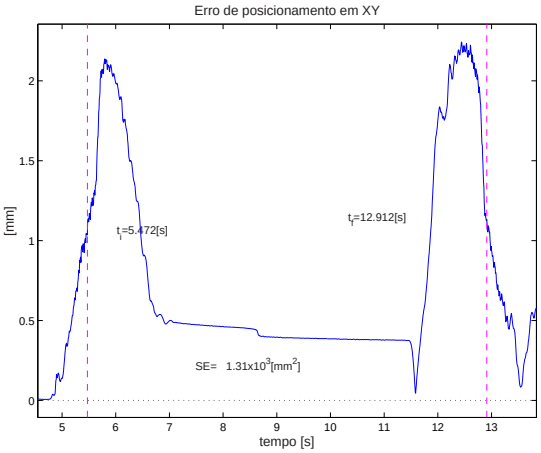
Figura 5.27: Perfis da trajetória com a perturbação presente (no espaço de juntas).

	Ensaio (Arquivo)	Controlador	Parâmetros
a)	(21041823)	PD de fábrica (com perturbação)	$K_p = [$ 4900 12100 90000 14400] $K_i = [$ 140 220 600 240]
b)	(21042207)	PID (com perturbação)	$K_p = [$ 4900 12100 90000 14400] $K_d = [$ 120 300 2300 350] $K_i = [$ 478 1200 9200 1410]
c)	(21041951)	PD + MLP1c (sem perturbação)	η=0.01, α=0.7 $12 \times 8 \times 4$
d)	(21041910)	PD + MLP1c (introduzida pertubação)	η=0.01, α=0.7 $12 \times 8 \times 4$
e)	(21041918)	PD + MLP1c ($10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 4$
f)	(21041924)	PD + MLP1c ($10 \times + 10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 4$
g)	(21041927)	PD + MLP1c (retirada a perturbação)	η=0.01, α=0.7 $12 \times 8 \times 4$
h)	(21041931)	PD + MLP1c ($10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 4$
i)	(21042014)	PID + MLP2c (sem perturbação)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
j)	(21042018)	PID + MLP2c (introduzida perturbação)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
k)	(21042025)	PID + MLP2c ($10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
l)	(21042029)	PID + MLP2c ($10 \times + 10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
m)	(21042032)	PID + MLP2c (retirada a perturbação)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
n)	(21042036)	PID + MLP2c ($10\times$ depois)	η=0.01, α=0.7 $12 \times 8 \times 5 \times 4$
o)	(22040215)	PID + RBF(5) (sem perturbação)	η=0.005, α=0.5 5 funções gaussianas
p)	(22050219)	PID + RBF(5) (com perturbação)	η=0.005, α=0.5 5 funções gaussianas
q)	(22040229)	Modos Deslizantes (sem perturbação)	$c = [$ 10 10 50 10] $\epsilon = [$ 0.3 0.3 1.2 1.2] $K = [$ 50 40 250 5]
r)	(22040231)	Modos Deslizantes (com perturbação)	$c = [$ 10 10 50 10] $\epsilon = [$ 0.3 0.3 1.2 1.2] $K = [$ 50 40 250 5]

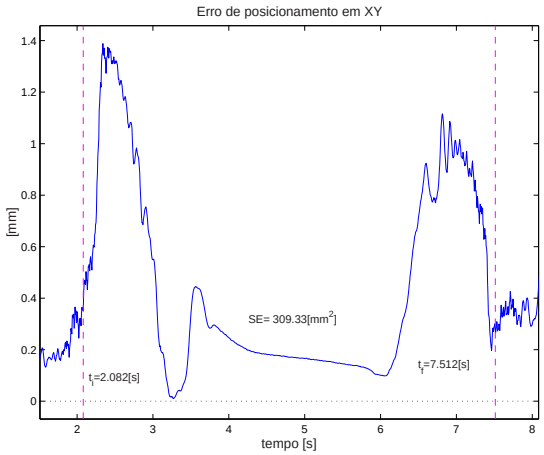
Tabela 5.26: Ensaios relativos à perturbação.



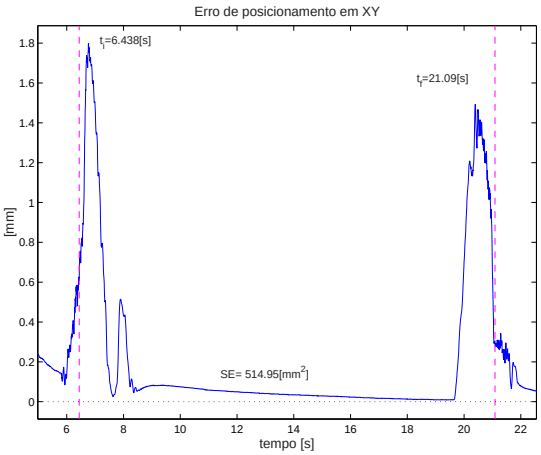
(a) PD de "fábrica" (com a perturbação).



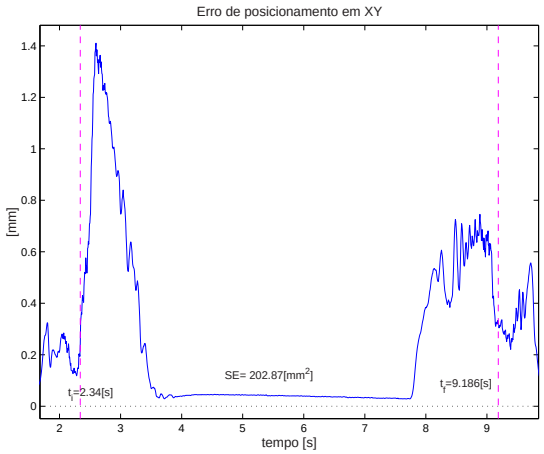
(b) PID (com perturbação).



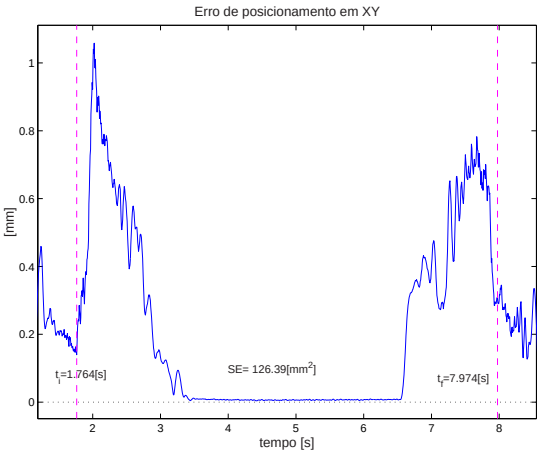
(c) PD + MLP1c (sem perturbação).



(d) PD + MLP1c (introduzida pertubação).

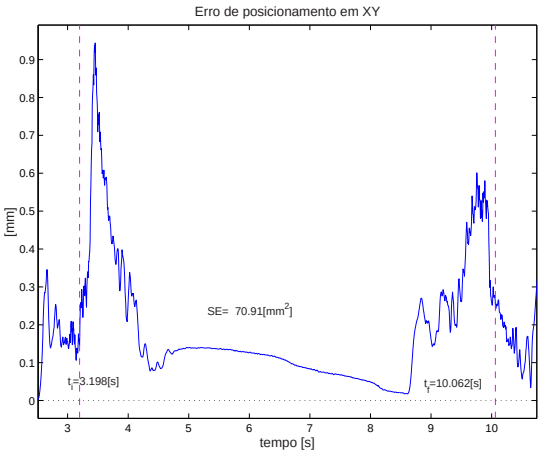


(e) PD + MLP1c (10× depois).

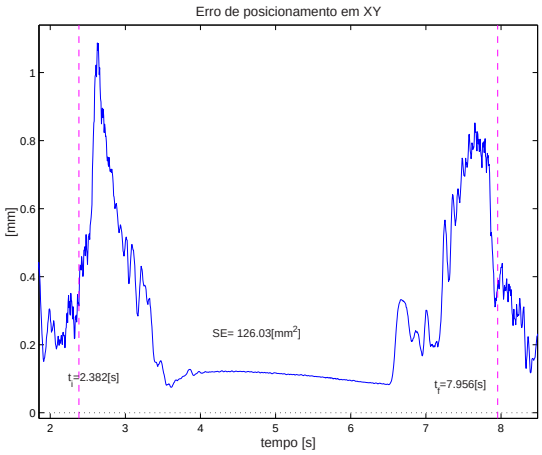


(f) PD + MLP1c (10 × +10× depois).

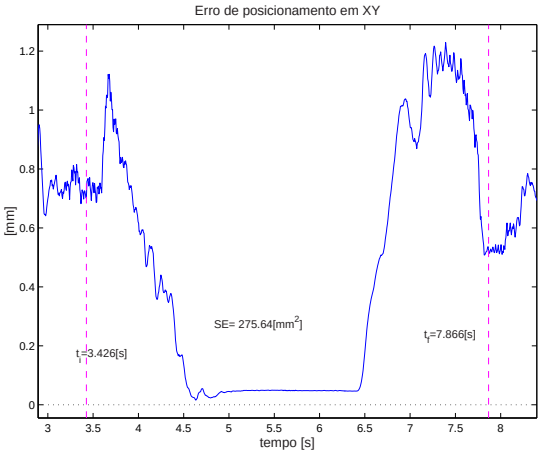
Figura 5.28: Erros de seguimento de trajetória em XY referentes ao teste com perturbação.



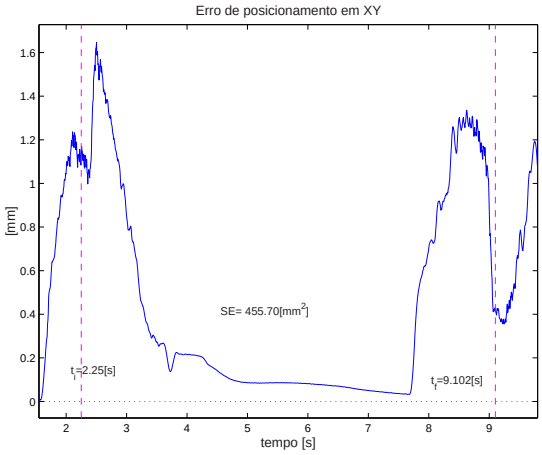
(g) PD + MLP1c (retirada a perturbação).



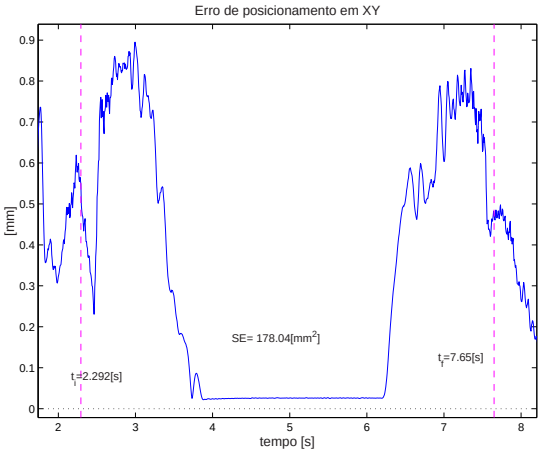
(h) PD + MLP1c (10× depois).



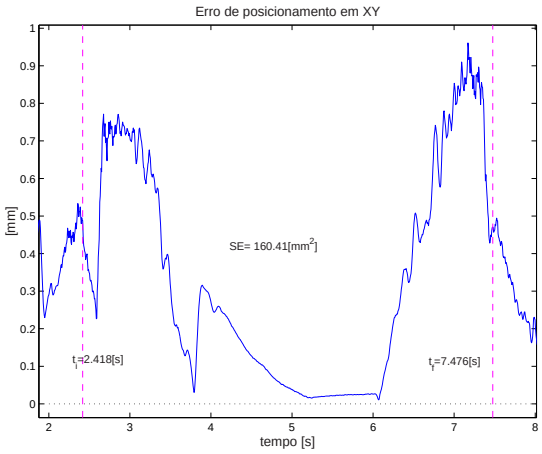
(i) PID + MLP2c (sem perturbação).



(j) PID + MLP2c (introduzida perturbação).

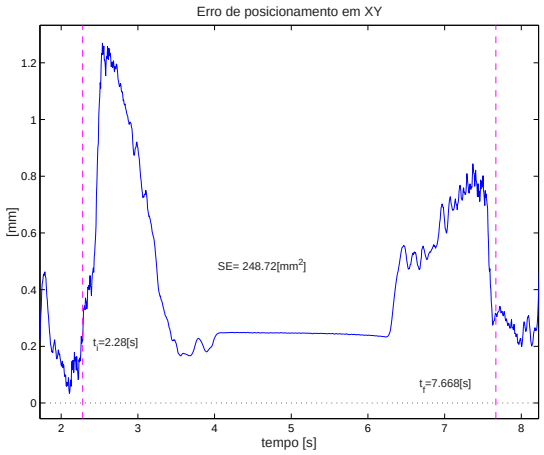


(k) PID + MLP2c (10× depois).

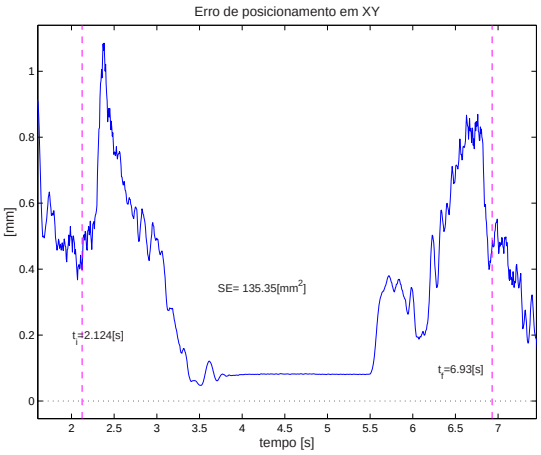


(l) PID + MLP2c (10 × +10× depois).

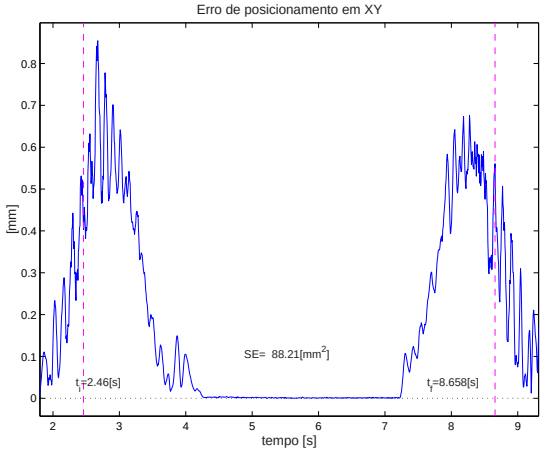
Figura 5.28: (Cont.) Erros de seguimento de trajetória em XY referentes ao teste com perturbação.



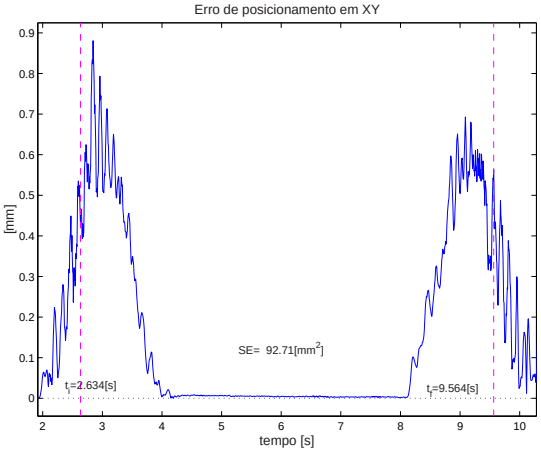
(m) PID + MLP2c (retirada a perturbação).



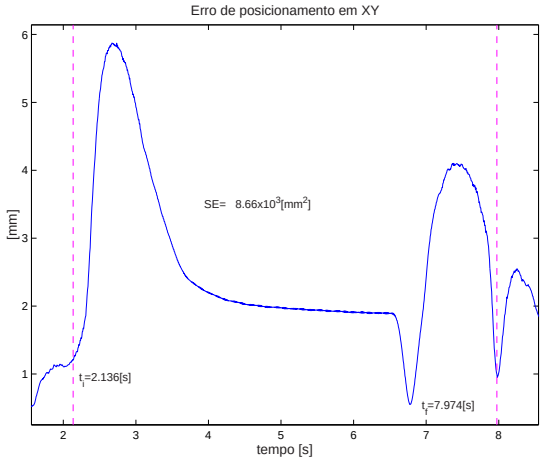
(n) PID + MLP2c (10× depois).



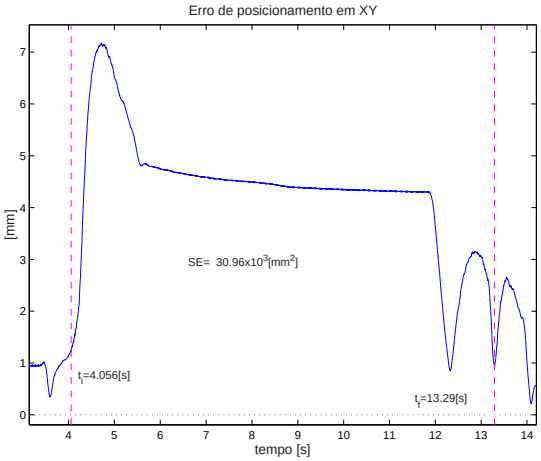
(o) PID + RBF(5) (sem perturbação).



(p) PID + RBF(5) (com perturbação).



(q) Modos Deslizantes (sem perturbação).

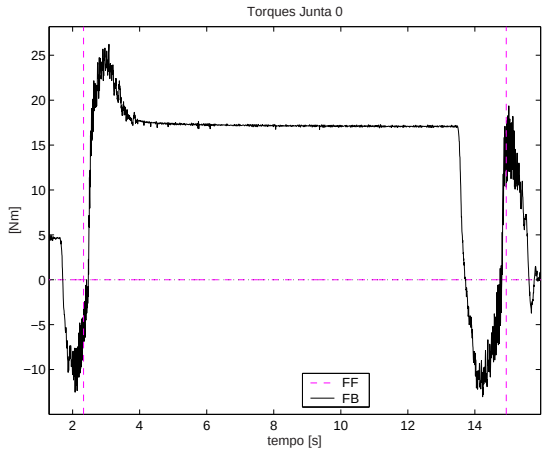


(r) Modos Deslizantes (com perturbação).

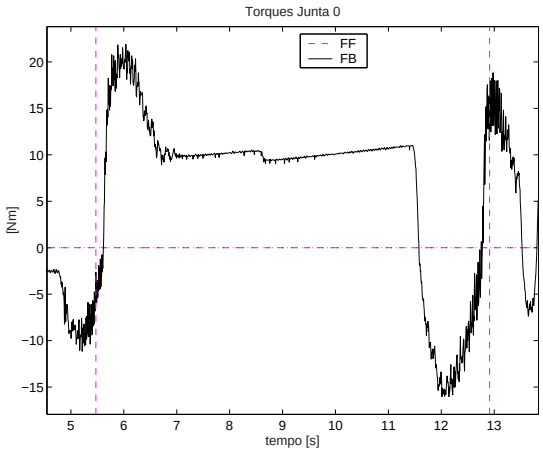
Figura 5.28: (Cont.) Erros de seguimento de trajetória em XY referentes ao teste com perturbação.

Gráficos dos Torques Desenvolvidos

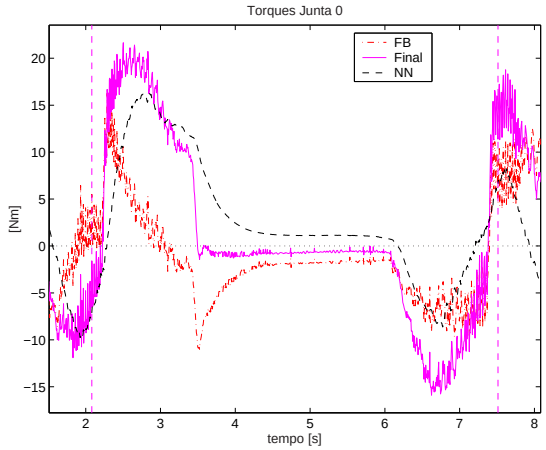
A figura 5.29 mostra os torques desenvolvidos pela junta 0 e a figura 5.30 para junta 1. São mostrados apenas as curvas dos torques gerados para as duas primeiras juntas do robô, já que praticamente não foi realizado movimento com as juntas restantes neste ensaio de rejeição à perturbação.



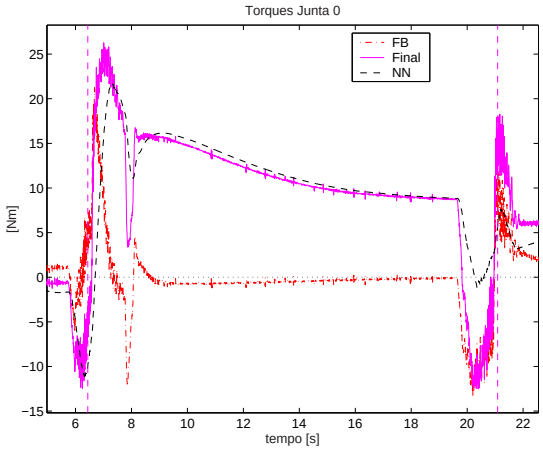
(a) PD de "fábrica" (com perturbação).



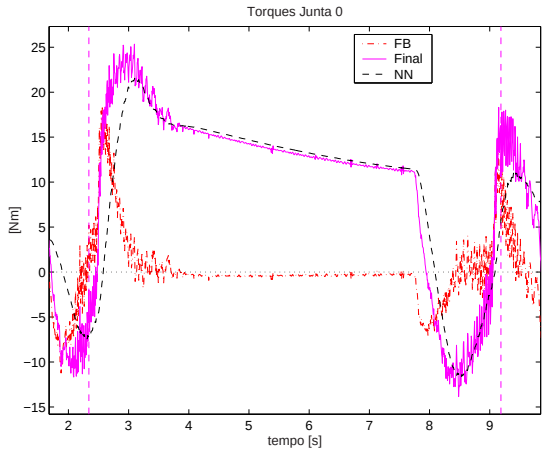
(b) PID (com perturbação).



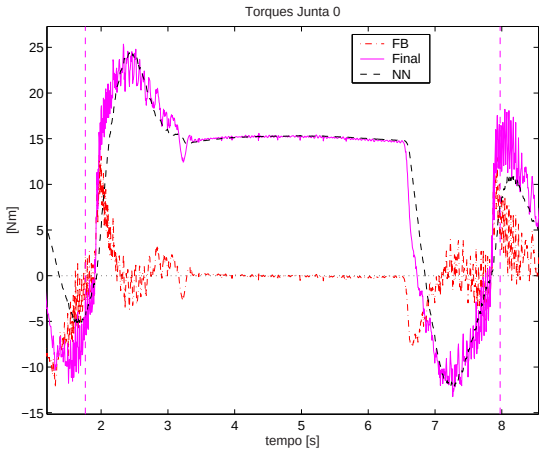
(c) PD + MLP1c (sem perturbação).



(d) PD + MLP1c (introduzida pertubação).

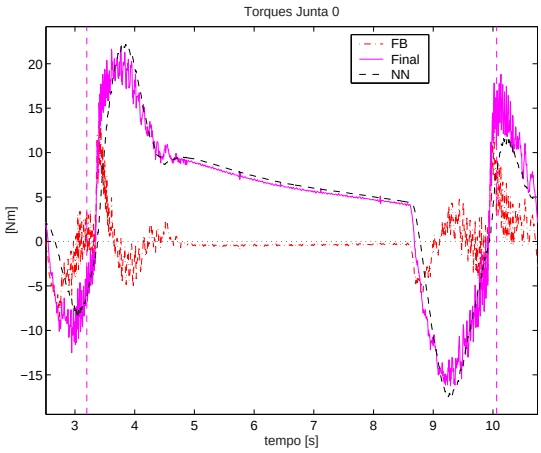


(e) PD + MLP1c (10× depois).

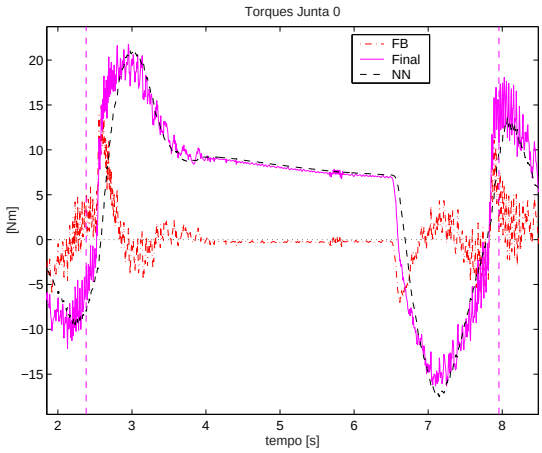


(f) PD + MLP1c (10 × +10× depois).

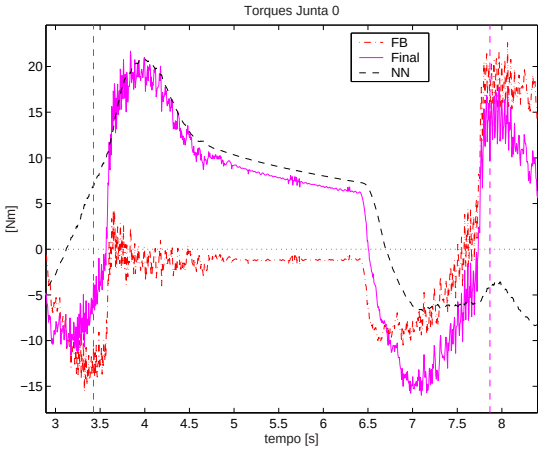
Figura 5.29: Torques enviados à junta 0 referentes ao teste com perturbação.



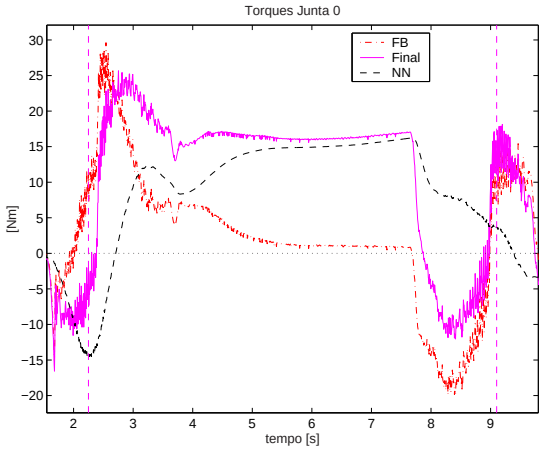
(g) PD + MLP1c (retirada a perturbação).



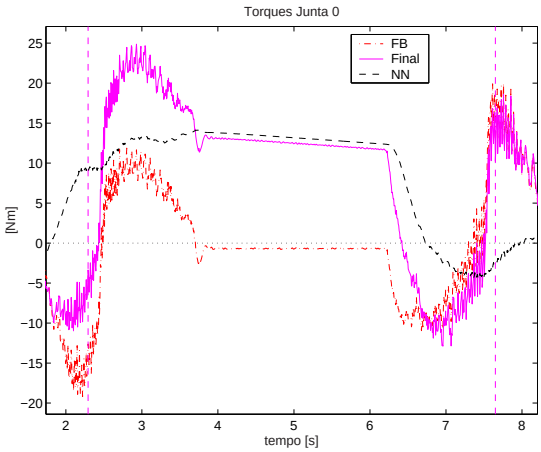
(h) PD + MLP1c (10× depois).



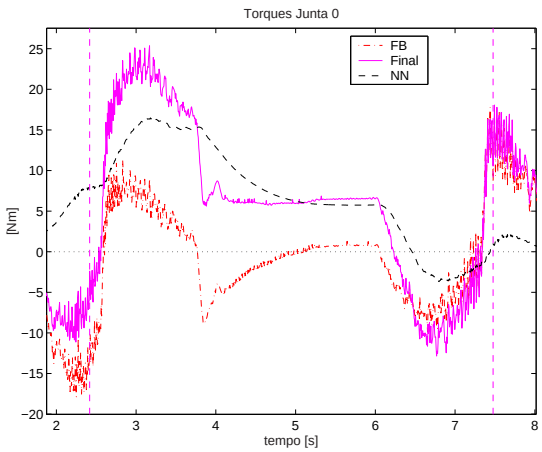
(i) PID + MLP2c (sem perturbação).



(j) PID + MLP2c (introduzida perturbação).

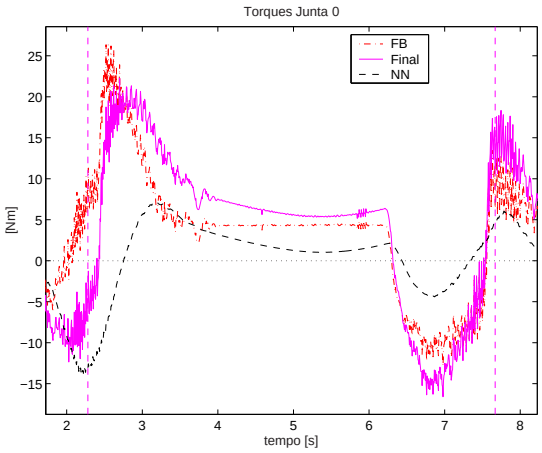


(k) PID + MLP2c (10× depois).

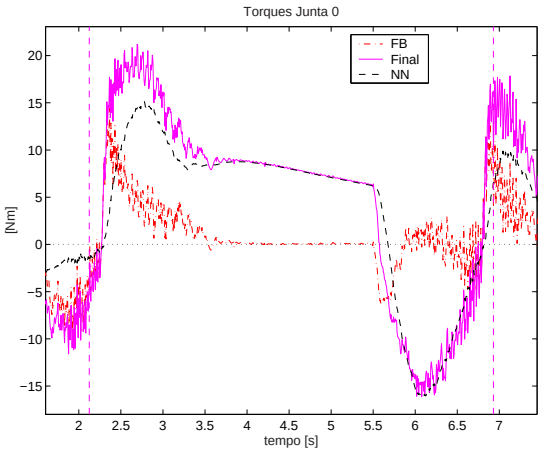


(l) PID + MLP2c (10 × +10× depois).

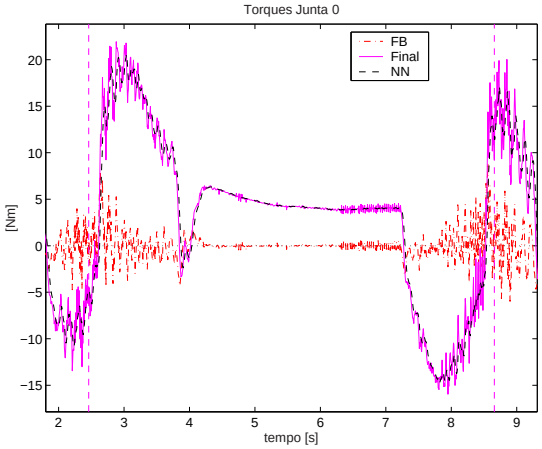
Figura 5.29: (Cont.) Torques enviados à junta 0 referentes ao teste com perturbação.



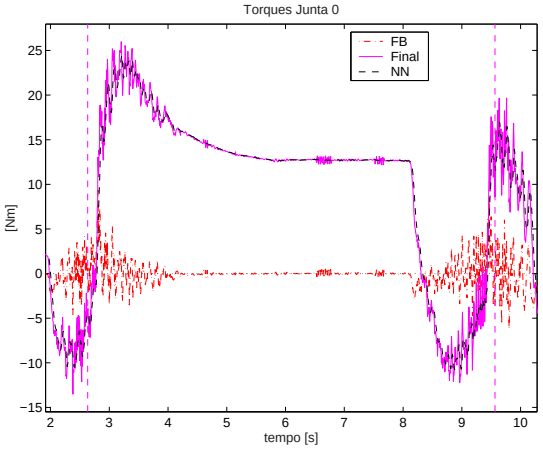
(m) PID + MLP2c (retirada a perturbação).



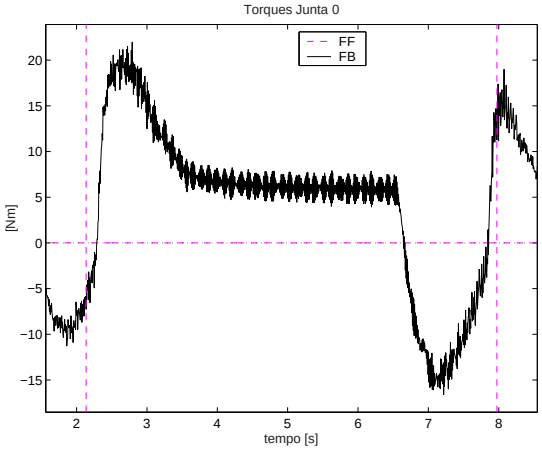
(n) PID + MLP2c (10× depois).



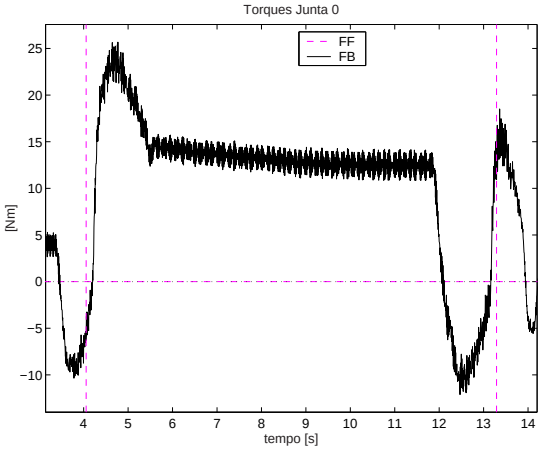
(o) PID + RBF(5) (sem perturbação).



(p) PID + RBF(5) (com perturbação).

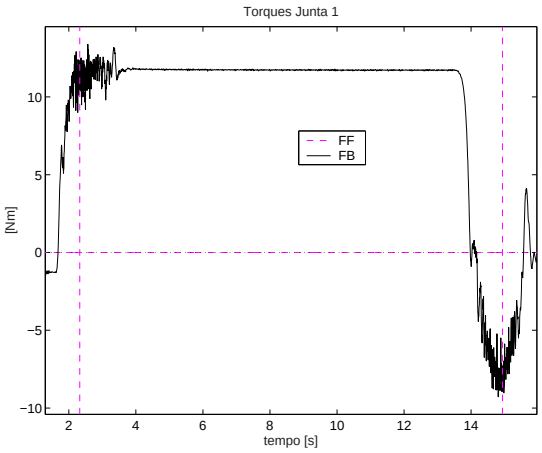


(q) Modos Deslizantes (sem perturbação).

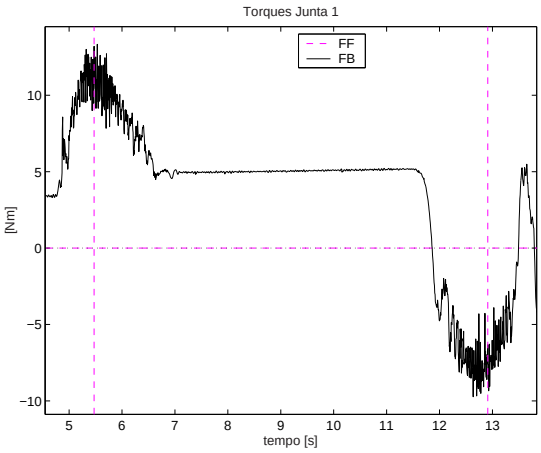


(r) Modos Deslizantes (com perturbação).

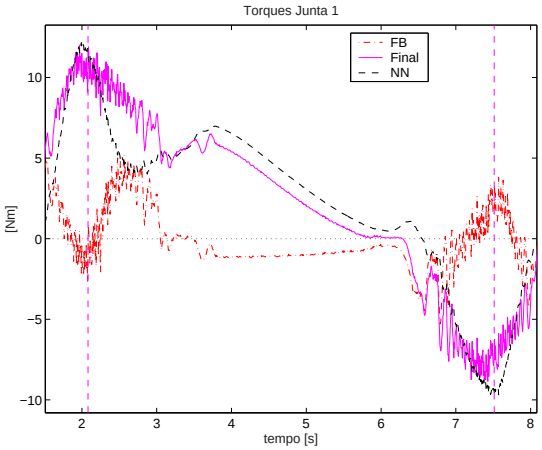
Figura 5.29: (Cont.) Torques enviados à junta 0 referentes ao teste com perturbação.



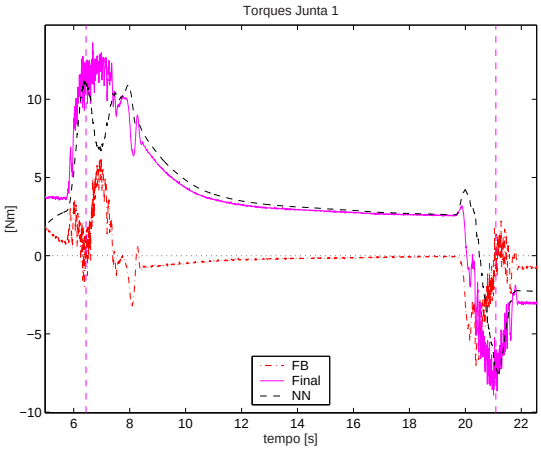
(a) PD de "fábrica" (com perturbação).



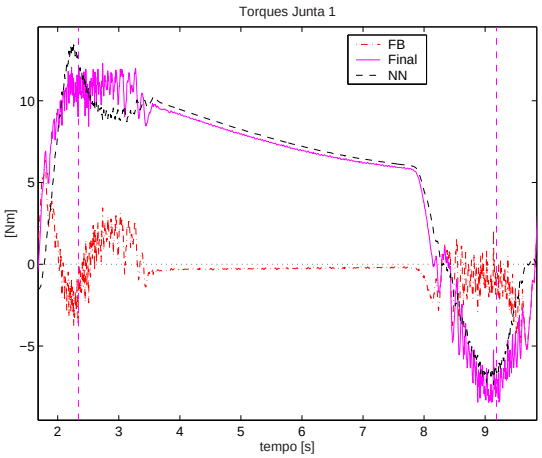
(b) PID (com perturbação).



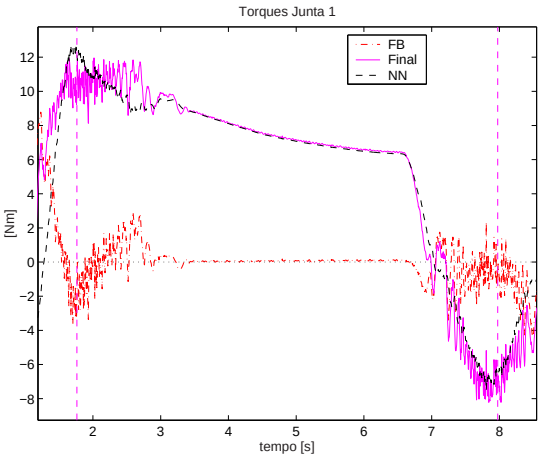
(c) PD + MLP1c (sem perturbação).



(d) PD + MLP1c (introduzida pertubação).

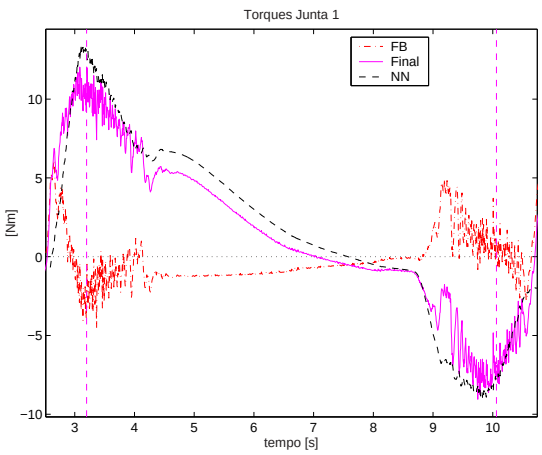


(e) PD + MLP1c (10× depois).

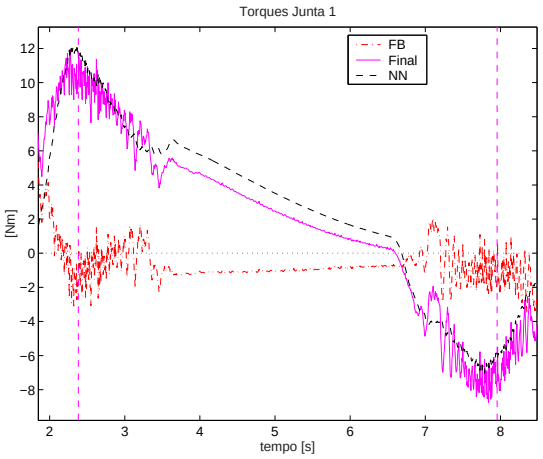


(f) PD + MLP1c (10 × +10× depois).

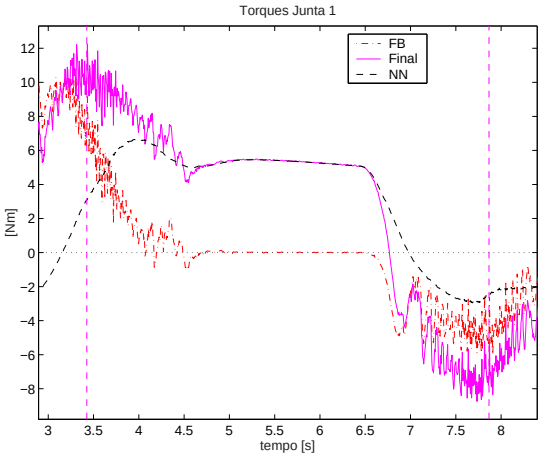
Figura 5.30: Torques enviados à junta 1 referentes ao teste com perturbação.



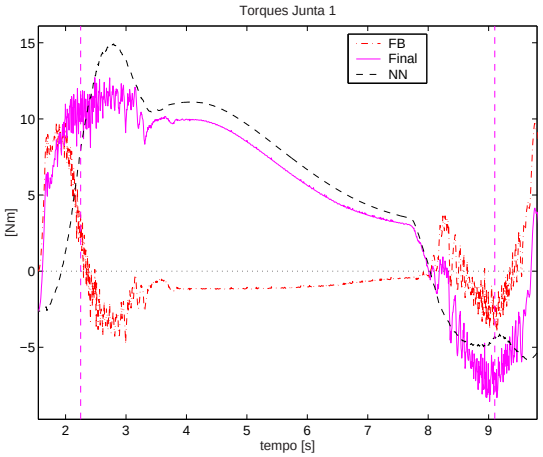
(g) PD + MLP1c (retirada a perturbação).



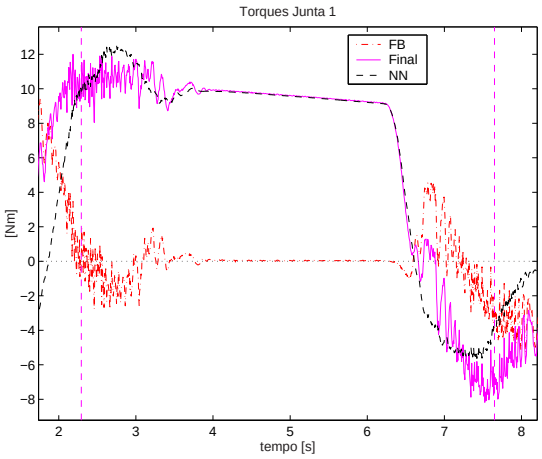
(h) PD + MLP1c (10× depois).



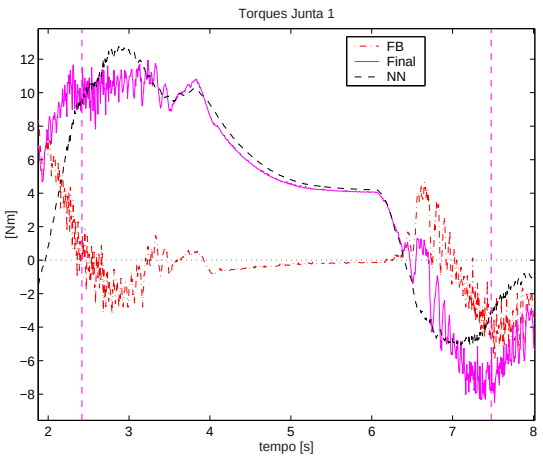
(i) PID + MLP2c (sem perturbação).



(j) PID + MLP2c (introduzida perturbação).

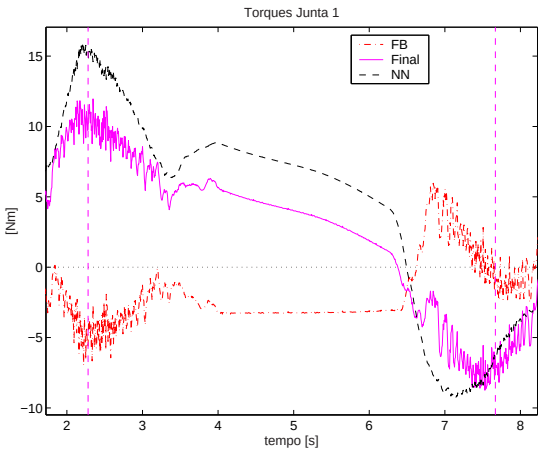


(k) PID + MLP2c (10× depois).

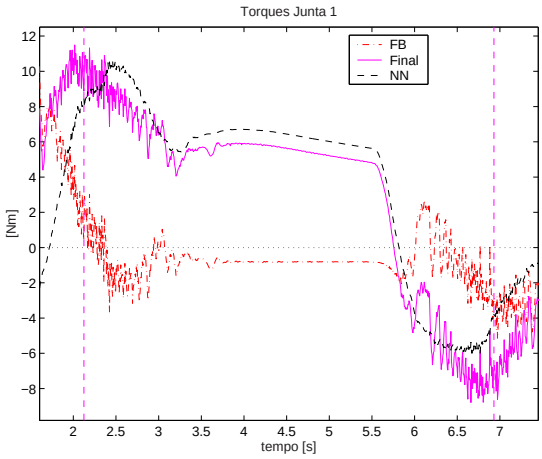


(l) PID + MLP2c (10 × +10× depois).

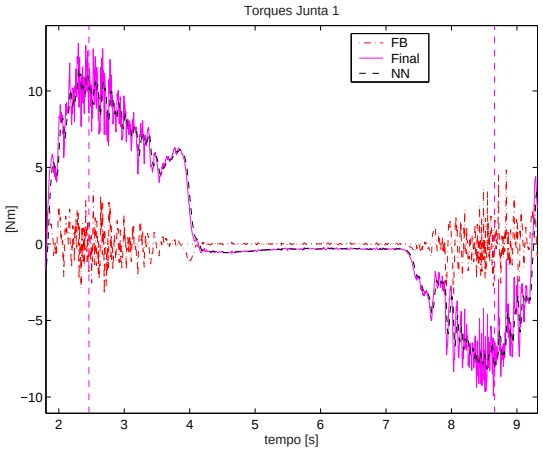
Figura 5.30: (Cont.) Torques enviados à junta 1 referentes ao teste com perturbação.



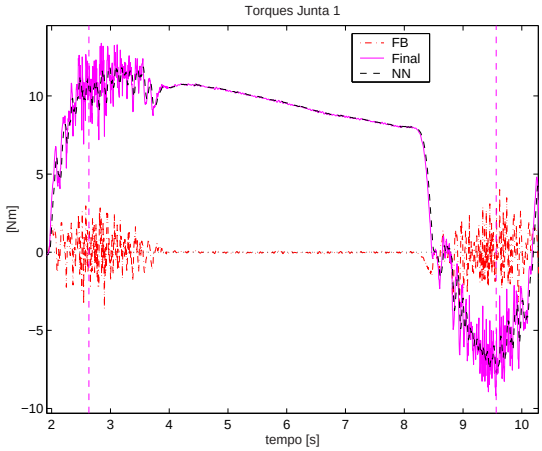
(m) PID + MLP2c (retirada a perturbação).



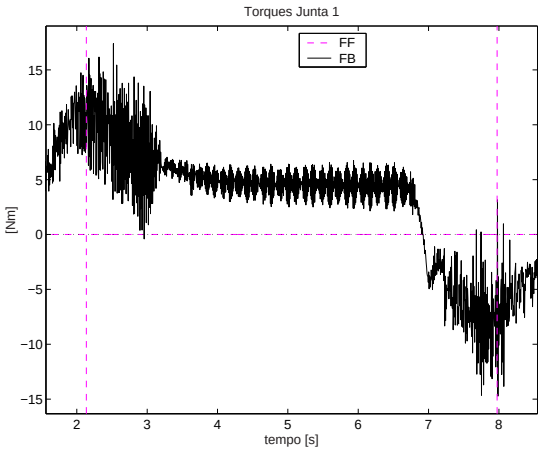
(n) PID + MLP2c (10× depois).



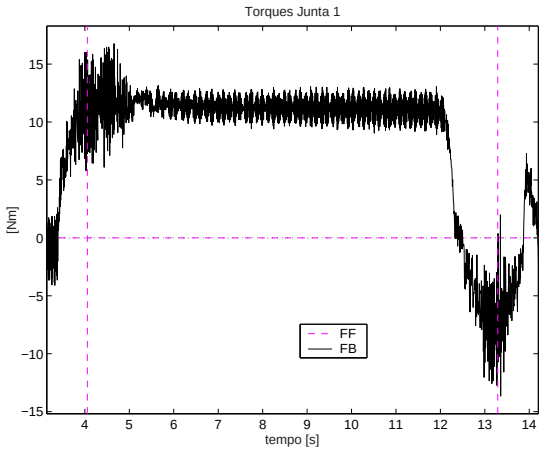
(o) PID + RBF(5) (sem perturbação).



(p) PID + RBF(5) (com perturbação).



(q) Modos Deslizantes (sem perturbação).



(r) Modos Deslizantes (com perturbação).

Figura 5.30: (Cont.) Torques enviados à junta 1 referentes ao teste com perturbação.

Índices de Desempenho

Segue a tabela 5.27 que mostra o erro médio quadrático acumulado ($MSE\{\tilde{d}\}$) dos erros de posicionamento verificados neste teste de rejeição à perturbação. Note que neste somatório apenas são levados em consideração os erros ocorridos dentro do intervalo de tempo no qual o efetuador do robô se encontra em contato com a cita elástica.

Controlador	$MSE\{\tilde{d}\}$ $[mm^2]$	$t_{ini}[s]$	$t_{fim}[s]$	$\Delta t[s]$
PD de fábrica (com a perturbação)	2.96	2.328	14.934	12.606
PID (com a perturbação)	1.06	5.472	12.912	7.440
PD + MLP1c (sem perturbação)	341.42m	2.082	7.512	5.430
PD + MLP1c (introduzida pertubação)	210.79m	6.438	21.090	14.652
PD + MLP1c ($10\times$ depois)	177.65m	2.340	9.186	6.846
PD + MLP1c ($10\times$ $+10\times$ depois)	122.00m	1.764	7.974	6.210
PD + MLP1c (retirada a perturbação)	61.93m	3.198	10.062	6.864
PD + MLP1c ($10\times$ depois)	135.52m	2.382	7.956	5.574
PID + MLP2c (sem perturbação)	371.98m	3.426	7.866	4.440
PID + MLP2c (introduzida perturbação)	398.69m	2.250	9.102	6.852
PID + MLP2c ($10\times$ depois)	199.15m	2.292	7.650	5.358
PID + MLP2c ($10\times$ $+10\times$ depois)	190.06m	2.418	7.476	5.058
PID + MLP2c (retirada a perturbação)	276.67m	2.280	7.668	5.388
PID + MLP2c ($10\times$ depois)	168.76m	2.142	6.930	4.806
PID + RBF(5) (sem perturbação)	85.31m	2.460	8.658	6.198
PID + RBF(5) (com perturbação)	80.20m	2.634	9.564	6.930
Modos Deslizantes (sem perturbação)	8.89	2.136	7.974	5.838
Modos Deslizantes (com perturbação)	20.10	4.056	13.290	9.234

Tabela 5.27: Erro médio quadrático ($MSE\{\tilde{d}\}$) encontrado nos testes de rejeição à perturbação.

Conclusões preliminares

As seguintes conclusões podem ser extraídas dos testes realizados anteriormente:

- O PID rejeitou melhor a perturbação que o PD conforme esperado devido à ação integral – figura 5.28(a) e (b).

- O controlador PD + MLP1c rejeita melhor as perturbação que isoladamente o PD (figuras 5.28(a) e (c)).

- Novamente, os controladores que apresentaram melhor desempenho foram o PID + RBF(5), com $\eta = 0.005$ e $\alpha = 0.5$.

- Comparando-se o comportamento do controlador PD + RBF(5) antes da perturbação e depois, com a perturbação presente (figuras 5.28(o) e (p)), percebe-se que este conjunto de controladores praticamente não percebeu a perturbação. Este controlador foi o que melhor rejeitou a perturbação introduzida no sistema.

- Já o controlador por modos deslizantes consegue rejeitar a perturbação mas não de forma tão eficiente quanto um PD ou PID, conforme pode ser percebido pelas figuras 5.28(q) e (r), e pela tabela 5.27 que compara o erro médio quadrático acumulado durante a execução da trajetória. E das figuras 5.29(o) e (p) referentes aos torques gerados para a junta 0 e igualmente pelas figuras 5.30(o) e (p), se percebe a necessidade de novo reajuste nos parâmetros deste controlador, já que ligeiro efeito de *chattering* pode ser percebido.

- A medida que o ensaio prossegue com a perturbação presente o controlador PD + MLP1c vai melhorando seu desempenho sem entretanto "viciar" seu treinamento devido a perturbação. Percebe-se pelas figuras 5.28(g) (retirada a perturbação) e figura 5.28(ensaio sem a perturbação) que o erro de seguimento de trajetória continua diminuindo. Dai pode-se concluir que o controlador usando a rede MLP contendo apenas 1 camada não apresenta nenhum efeito residual negativo de memorização. Ou seja, trajetórias repetitivas melhoram a forma como esta rede realiza a compensação dinâmica, sem entretanto "viciar" a rede. Tão logo uma nova configuração seja exigida do robô, a rede contendo apenas 1 camada se re-adapta rapidamente a esta nova situação.

- Analisando-se as figuras dos torques encaminhados para as juntas do robô, fica clara a necessidade do controlador convencional trabalhando em paralelo com a rede neural, principalmente no caso da rede MLP. Percebe-se pelas figuras 5.29(c) e (d) (torques enviados para a junta 0) e figuras 5.30(c) e (d) (torques enviados para a junta 1), que o fato do controlador convencional aumentar sua ação de controle, exige um aprendizado mais intenso da rede. No início das figuras anteriores, é nítido que a rede só começa a reagir de forma preponderante na ação de controle depois que o controlador convencional entra em ação. A ação do controlador convencional apenas diminui um

pouco sua ação se a mesma trajetória é repetida. Dai, outra conclusão: o controlador integrado baseado nas redes MLP necessita da ação de controle gerada pelo controlador convencional que trabalha em paralelo com ela, para que ela se torne "sensível" a necessidade de um reajuste mais intenso de seus pesos. De fato, conforme aumenta o valor do torque gerado pelo controlador convencional, aumenta o sinal de erro usado pela rede na sua etapa de reajuste dos pesos. Este comportamento é quase idêntico entre uma rede MLP contendo 1 camada invisível ou 2. O único porém, é que a rede MLP com apenas 1 camada apresenta respostas mais rápidas que a rede com 2 camadas, sem entretanto, apresentar desempenho inferior.

- Realizando-se esta mesma análise sobre os torques gerados pela rede RBF com o controlador PID, se percebe que esta rede é muito mais "sensível" a uma necessidade mais forte para reajuste de seus pesos de saída. Percebe-se pelas figuras 5.29(o) (sem a perturbação) e (p) (com a perturbação), que esta rede além de reagir mais rápido à perturbação introduzida ao sistema, praticamente não deixa o controlador convencional que trabalha em paralelo com ela, reagir. Mas da mesma forma que no item anterior, mesmo os baixos torques gerados pelo controlador convencional no caso de operação com a rede RBF, são necessários para acentuar a forma como esta rede reajusta seus pesos. Apenas que comparando o comportamento desta rede com a MLP, que são distintos, se percebe que esta rede é muito mais rápida que a rede MLP e mais precisa também – seus erros quadráticos médios de seguimento de trajetória são muito inferiores aos do controlador trabalhando com a rede MLP (tabela 5.27). E nem por isto, os torques gerados pelo controlador trabalhando com a rede RBF são mais oscilatórios.

- Já a rede MLP com 2 camadas invisíveis apresenta certo efeito de memória residual. A rede MLP com apenas 1 camada além de reagir mais rápido à mudanças de configuração do robô, apresenta um caráter mais "elástico", flexível para reajuste de seus pesos sinápticos independente de trajetórias repetitivas (com ou sem perturbação) ou não. Nota-se pelas figura 5.28(i) (PID + MLP2c, sem a perturbação) e figura 5.28(h) (PID + MLP2c, logo após retirada a perturbação), um ligeiro efeito residual de memória. Se bem que este leve efeito residual não chega a comprometer o desempenho deste controlador integrado, uma vez que o $MSE\{\tilde{d}\}$ relativo à estes dois

instantes ainda baixam – tabela 5.27. Mas por esta mesma tabela se percebe que a rede com 2 camadas reage de maneira um pouco mais lenta à entrada da perturbação ao sistema. Dai poder-se concluir que a rede MLP com 2 camadas além de exigir maior poder de processamento da CPU do robô, possui uma menor capacidade de reagir à perturbações e mudanças de configuração do robô, em relação ao uso de outra rede MLP contendo apenas 1 camada invisível.

5.7 Discussão Final

Os testes realizados anteriormente (e outros tantos que foram realizados mas não foram aqui apresentados) se pode concluir:

- Os controladores integrados, baseado na arquitetura de realimentação do erro (*feedback error learning control*), que mesclam a ação de controle gerada por um controlador convencional com a ação de controle gerada por uma rede neural, propiciam um desempenho final superior ao verificado para controladores proporcionais simples ou mesmo robustos (como no caso do controlador por modos deslizantes utilizado nos testes).

- A arquitetura do controlador integrado utilizada reflete um comportamento característico para as ações de controle geradas tanto pela rede neural quanto pelo controlador convencional. É facilmente perceptível que uma mudança de configuração do robô, induz a que inicialmente o controlador convencional até gere uma ação de controle preponderante, mas estes mesmos valores não nulos de torques gerados pelo controlador convencional, força com que a rede reajuste seus pesos de forma mais dramática, fazendo com que a mesma rapidamente assuma a ação de controle preponderante sendo gerada (isto ocorre ainda antes do terço inicial da execução de uma nova trajetória). O novo reajuste dos pesos da rede, faz com que a mesma praticamente se responsabilize por todo o torque sendo enviado para cada junta do robô (perceptível principalmente na composição do torque final enviado às juntas no caso do robô em regime permanente, repouso). Isto indica que a rede neural nesta arquitetura de controle é capaz de realizar a compensação dinâmica necessária de forma até a suprimir a ação de controle que antes era gerada pelo controlador convencional. Entretanto, o

controlador convencional não pode ser suprimido do sistema uma vez que ele é fundamental para disparar a necessidade de reajuste dos pesos da rede.

- Mesmo o emprego de redes neurais com mais de uma camada invisível não chega a comprometer o desempenho final do sistema devido a algum efeito residual de memorização. Apenas que, uma rede neural com 2 camadas ocultas acabe apresentando um fator de "inércia" maior para reajuste de seus pesos. O que implica em que uma rede com 2 camadas resulte numa velocidade de resposta (e adaptação à uma nova situação) mais lenta que uma rede contendo apenas 1 camada invisível. Além disto, a rede contendo apenas 1 camada invisível inclusive apresenta menores erros de seguimento de trajetória que uma rede com 2 camadas ocultas. Dai se conclui que do ponto de vista: custo (poder de processamento exigido) $\times$ benefício (menores erros de seguimento de trajetória), a rede contendo apenas 1 camada oculta é mais adequada para aplicações práticas principalmente no caso de restrições relacionadas com tempos de processamento para a tarefa em tempo-real relacionada com a lei de controle.

- O objetivo das redes utilizadas em paralelo com os controladores convencionais era o de realizar a compensação dinâmica necessária de forma a anular os erros de seguimento de trajetória. Tanto a rede MLP quanto a rede RBF permitem que este objetivo seja plenamente atingido, desde que entretanto não se empreguem taxas de aprendizado demasiadamente excessivas para as redes neurais. Taxas de aprendizado elevadas, além de induzir comportamentos oscilatórios, podem, num pior caso, levar a saturação dos pesos da rede. Neste pior caso, a saída da rede satura num dos extremos da ação de controle possível de ser encaminha às juntas do robô. Se for adotada uma estratégia de *software* que não limite os valores de torque gerados pela rede neural ou pelo controlador convencional e que apenas limite o torque final resultante do somatório destes torques, ainda assim, é possível se garantir a estabilidade do controlador, pois o controlador convencional tende a corrigir o comportamento errôneo apresentado pela rede. Esta foi a estratégia empregada na implementação dos torques de saída à serem enviados à cada junta do robô. Por *software* foi limitado apenas o torque final máximo que poderia ser enviado à cada junta do robô, o que no casos em que isto ocorria, incrementava as mensagens de advertência do novo sistema de controle implementado, informando ao usuário, que rede saturou, qual atuador e para

que junta. Da forma como foi implementado o novo controlador de posição é possível até mesmo se descobrir qual o neurônio que saturou sua saída. Naturalmente casos como este ocorreram na prática, principalmente durante a fase de "tentativa-e-erro" para ajuste de parâmetros do controlador e e das redes.

- A taxa de aprendizado e momento mais adequadas para operação com as redes MLP foi de: $\eta = 0.035$ e $\alpha = 0.5$, respectivamente.

- Já para a rede RBF, a taxa de aprendizado que se mostrou a mais adequada foi de $\eta = 0.005$ e termo *momentum*, $\alpha = 0.5$.

- Foi percebido que o controlador convencional mais adequado para operação conjunta com as rede neurais empregadas para realizar a compensação dinâmica é o PID, nitidamente devido à sua ação integral. Tão logo o torque de saída gerado pelo controlador PID aumente devido à alguma mudança no sistema, a rede neural trabalhando em paralelo com este, reajusta seus pesos de maneira mais rápida e volta mais rapidamente a assumir a ação preponderante de controle.

- O controlador que apresentou melhor desempenho: menores erros de seguimento de trajetória, capacidade mais rápida de resposta e melhor rejeição à perturbações foi o **PID + RBF(5)**. Inclusive gostaria-se de ressaltar o baixíssimo nível de ruído desenvolvido pelo robô quando operado pelo PID + RBF(5) mesmo para os movimentos que exigem maiores deslocamentos e maiores velocidades de resposta. Em seguida figuram os controladores: PID + MLP1c, PID + MLP2c, PID e PD.

- Apesar do uso de um controlador convencional junto com uma rede neural, deve-se ficar atento ao poder de processamento extra exigido do processador para operação com tais redes neurais. A rede que apresentou o melhor desempenho também foi a que mais poder de processamento exigiu da CPU do robô, ver tabela 5.24. Isto pode ser um fator limitante principalmente nos casos em que existe baixo poder de processamento disponível ou os tempos para processamento são limitados. Foi constatado inclusive que uma rede RBF com 9 funções gaussianas exige um poder de processamento tal que não foi mais possível executar a lei de controle usando o período de amostragem original do robô que era de 1[ms]. Por conseguinte uma nova versão para o sistema controlador do robô foi desenvolvida adotando como período de amostragem o valor

de 2[ms], já levando em conta o fato de ser necessário executar mais uma rede neural distinta para controle de força (próximo capítulo). E mesmo assim, ainda se foi obrigado a instalar testes adicionais à nível de *software* neste novo sistema de controle a fim de verificar se realmente sobraram recursos no sistema XO/2 para instalação dinâmica das tarefas em tempo-real necessárias para operacionalizar o sistema. Houve mais de um caso em que foi detectada a falta de recursos no XO/2 para efetivamente instalar a própria tarefa em tempo-real relacionada com a lei de controle. Neste caso, o sistema XO/2 não se mostra um sistema tempo-real muito adequado pois no caso de verificada sobrecarga no tempo máximo de processamento especificado para certa tarefa (parâmetro *deadline*), seu gerenciador de tarefas simplesmente "suspende" a execução desta tarefa, sem haver alguma forma a nível de *software* para se poder avaliar continuamente se certas tarefas críticas continuam sendo executadas (ao menos não foi descoberta em nenhuma documentação disponível acerca do sistema XO/2 ou mesmo consultando-se a lista de discussão relacionado com este sistema[1]) . Mais de uma vez, a própria tarefa em tempo-real relacionada com a lei de controle (Módulo Robot.Mod, PROCEDURE (h: ControlFBHdl) Run();) foi suspensa. Neste caso, é visualmente perceptível alguma falha de execução no robô, uma vez que os últimos torques que haviam sido calculados antes da tarefa da lei de controle ter sido suspensa continuam sendo enviados para as juntas do robô até que a rotina de segurança entre em ação devido à excessivos erros de seguimento de trajetória ou alguma chave de fim de curso tenha sido atingida ou ainda tenha sido detectado algum impacto contra o efetuador final do robô (esta última característica de segurança foi acrescentada à rotina original de tempo real PROCEDURE (h: ControlSecurityHdl) Run(); que roda a cada 10[ms], de forma a tentar garantir a integridade do próprio sensor de força do robô já objetivando as futuras aplicações de controle de força). A tabela 5.28 mostra os tempos de processamento reais exigidos para operar os diferentes controladores.

- A figura 5.31 mostra no mesmo gráfico a relação custo benefício alcançada pelos diferentes controladores de posição testados. Através desta figura se percebe que se há baixo poder de processamento disponível o melhor é optar por um controlador convencional do tipo PID. Se ainda há tempo de processamento disponível então pode-se

[1] *Oberon@inf.ethz.ch* mailing list for ETH Oberon and related systems, ou http://www.lists.inf.ethz.ch/mailman/listinfo/oberon.

Tipo de Controlador	Tempos de Processamento	
	Min	Max
PD + MLP2c	0.194	0.385
PD + RBF(5)	0.333	0.579
PD + RBF(7)	0.425	0.679
PD + RBF(9)	0.104	0.809

Obs.: Tempos de Processamento em [ms].

Tabela 5.28: Tempos de processamento exigidos pelos diferentes controladores integrados de posição.

optar pelo PID + MLP1c (com apenas 1 camada oculta). E caso haja ”folga” no processamento, o controlador mais indicado seria o PID + RBF(5) (com 5 funções gaussinas) que permite levar ao melhor resultado dentro os apresentados anteriormente.

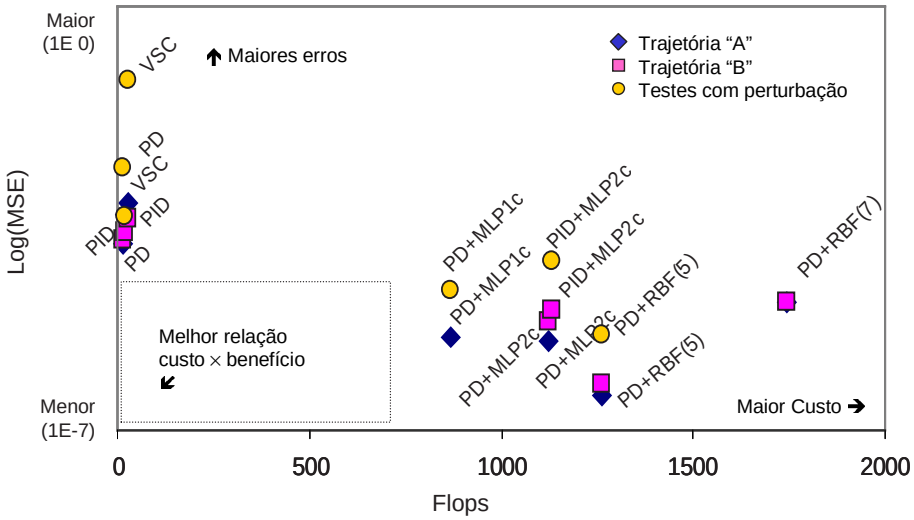


Figura 5.31: Relação custo × benefício dos controladores de posição

- Mesmo que o controlador PID + RBF(5) tenha apresentado os melhores resultados, acrescentar mais funções gaussianas à camada intermediária desta rede não reflete uma melhora significativa no desempenho deste rede em comparação com o poder de processamento extra que será exigido da CPU do robô.

- Foi verificada ainda uma dificuldade real para ajuste dos parâmetros dos controladores num robô manipulador real. Mesmo o uso de métodos como o de Ziegler-Nichols para ajuste de controladores PID não garante os melhores resultados para todas as configurações que o robô pode assumir dentro de seu espaço de trabalho.

Capítulo 6

Resultados para Controle de Força/Posição

6.1 Introdução

Neste capítulo serão apresentados os resultados obtidos com os controladores de posição/força implementados. Foi adotada a arquitetura híbrida para controle de posição/força onde duas malhas distintas de controle são projetadas: uma malha para controle de posição, realizado nas direções em que o movimento do robô não está restringido; e outra malha para o controle de força que é realizado nas direções onde o movimento do robô está restringido. Foram implementados controladores de força explícitos, isto é, fechando a malha de controle de força com base nas informações captadas à partir do sensor de força JR3 instalado no robô Inter/Scara (figura 4.2, pág. 103).

O capítulo inicia com resultados de testes de controladores de posição operando diretamente no espaço operacional e continua com os testes dos controladores convencionais de força (alguns controladores de posição no espaço operacional foram utilizados dentro da arquitetura de controle híbrido de posição/força na malha de controle de posição). Primeiramente foram avaliados os resultados obtidos com controle Proporcional de força e depois com controlador PI e seguem após os resultados dos testes obtidos com as 4 variações diferentes de rede neural para a malha de controle de força (RNf-1 à RNf-4). Cada conjunto de resultados inclui um item particular contendo suas conclusões preliminares e quase ao final é realizada uma discussão com relação aos resultados obtidos com os diferentes tipos

de controladores integrados de força testados. O capítulo encerra com uma discussão final acerca o tema controle de força confrontando os diferentes resultados obtidos.

6.2 Resultados dos Testes com Controladores de Posição no Espaço Operacional

Antes de prosseguir diretamente para o controle de força, foram implementados controladores de posição para atuar diretamente no espaço operacional uma vez que o controle de força/posição é normalmente realizado no espaço operacional (de coordenadas de base do robô) ou no espaço tarefa (um sistema cartesiano cujas origens não coincidem com os das coordenadas de base do robô, mas por facilitar a especificação de certas tarefas é por vezes adotado). Foram implementados e testados 3 tipos de controladores capazes de atuar diretamente à partir no espaço operacional:

1. Controlador por Jacobiana Inversa (J_A^{-1});
2. Controlador PD no Espaço de Juntas (PD-Op); e;
3. Controlador PID no Espaço de Juntas (PID-Op).

Estes controladores podem ser utilizados na malha de controle de posição do controlador híbrido de posição/força, sendo que as redes neurais para a malha de controle de posição estão preparadas para trabalhar tanto com os controladores convencionais operando sobre o espaço de juntas quanto sobre o espaço operacional. Foi implementado um *array* de objetos relacionados com os 2 tipos de redes neurais implementadas (MLP e RBF), desta forma, são associadas diferentes redes neurais (com diferentes estados de aprendizado e características próprias diferentes) para a malha de controle de posição e malha de controle de força.

A seguir são apresentados os testes efetuados para os controladores de posição implementados para operação sobre o espaço operacional.

6.2.1 Trajetória do Teste

Para o teste dos controladores operando no espaço operacional foi prevista a trajetória de teste mostrada na figura 6.1. Na realidade uma seqüencia de movimentos (comandos fornecidos pelo usuário no espaço operacional): do ponto (a) para o ponto (b), de (b) para

(c) e de (c) para (d) – conforme pode ser verificado em maiores detalhes na tabela 6.1. A velocidade máxima especificada para execução das trajetórias ponto a ponto foi de 0.5 [m/s].

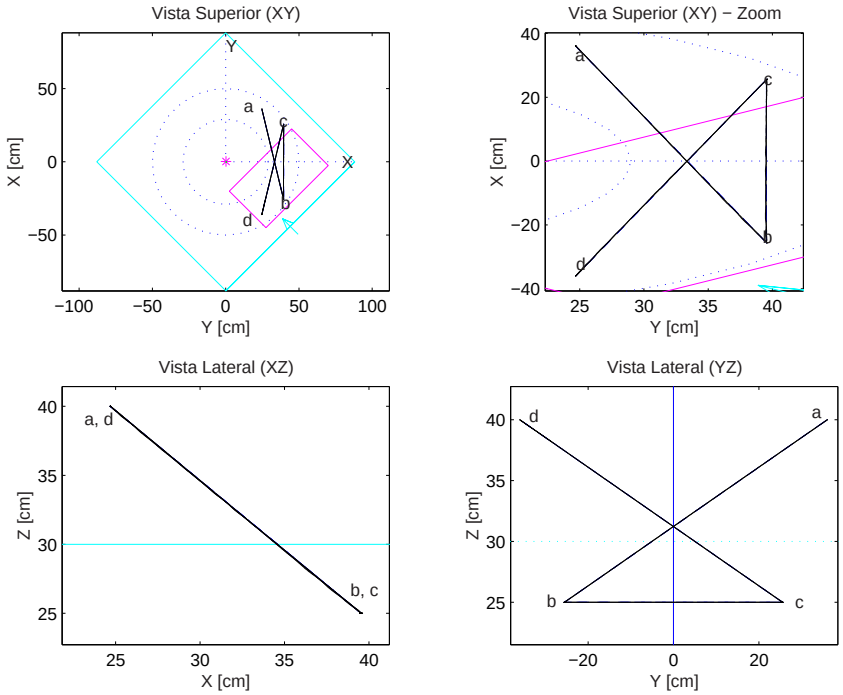


Figura 6.1: Trajetória de teste prevista para os controladores no espaço operacional.

A tabela 6.1 mostra os dados da trajetória utilizada para teste destes controladores.

Posição	X [m]	Y [m]	Z [m]	θ [rad]
x_a	0.2468	0.3600	0.40	1.5707
x_b	0.3950	-0.2560	0.25	1.5707
x_c	0.3950	0.2560	0.25	1.5707
x_d	0.2468	0.3600	0.40	1.5707
$\dot{x}_{máx}$	-0.1082	0.4481	0.1092	-1.3470
$\ddot{x}_{máx}$	0.1580	-0.6514	-0.1594	1.0000

Tabela 6.1: Dados da trajetória usada para o teste dos controladores operando sobre o espaço operacional.

6.2.2 Teste do Controlador por Jacobiana Inversa (J_A^{-1})

Os ganhos encontrados pelo método de tentativa e erro para este controlador (eq. 4.9, pág. 113) foram: $K_p = [\ 6000 \quad 6000 \quad 40000 \quad 120\ ]$, relativos respectivamente à direção X, Y e Z e orientação final do robô: θ. Os erros de seguimento de trajetória observados com

este teste aparecem na figura 6.2. Os índices de desempenho alcançados para este controlador são apresentados mais adiante na página 222, item 6.2.6 juntamente com os resultados encontrados para os outros controladores projetados diretamente para o espaço operacional.

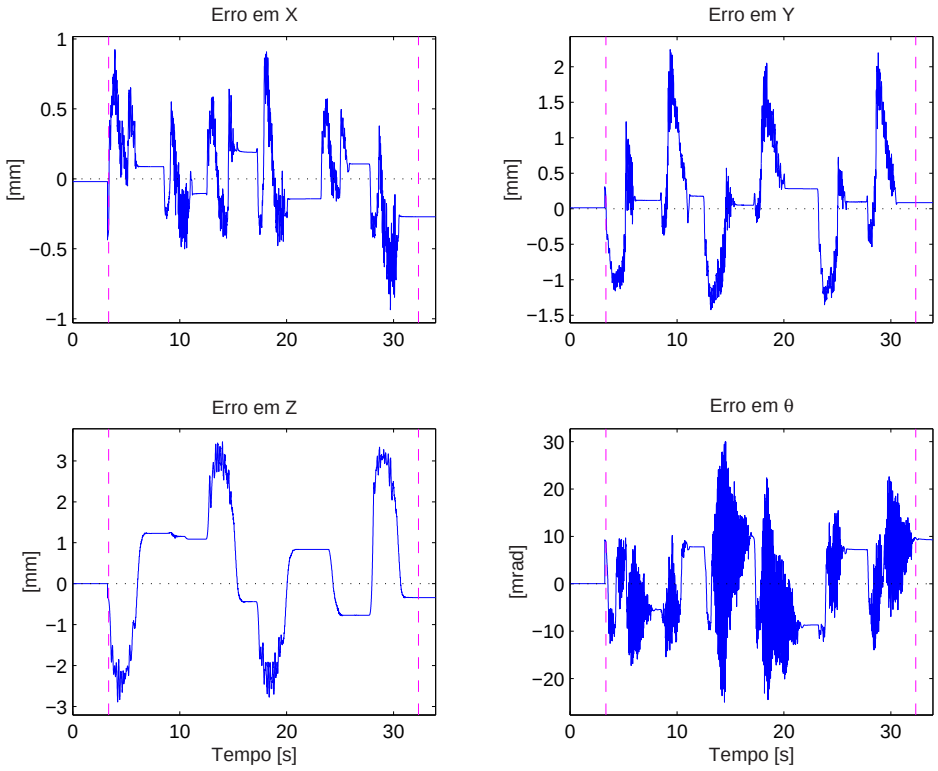


Figura 6.2: Erros do teste para o controlador por J_A^{-1}.

6.2.3 Testes do Controlador PD-Op

Este controlador foi sintonizado baseado nos parâmetros de Ziegler-Nichols. Os valores encontrados para os ganhos são mostrados na tabela 6.2. O traçado dos erros de seguimento de trajetória são mostrados na figura 6.3. Os índices de desempenho alcançados são apresentados mais adiante na página 222, item 6.2.6.

Parâmetro	Direção X	Direção Y	Direção Z	Orientação θ
K_p	102.000,0	102.000,0	1.740.000,0	240,0
K_d	425,0	425,0	10.875,0	1,2

Tabela 6.2: Ganhos encontrados para controlador PD no espaço operacional.

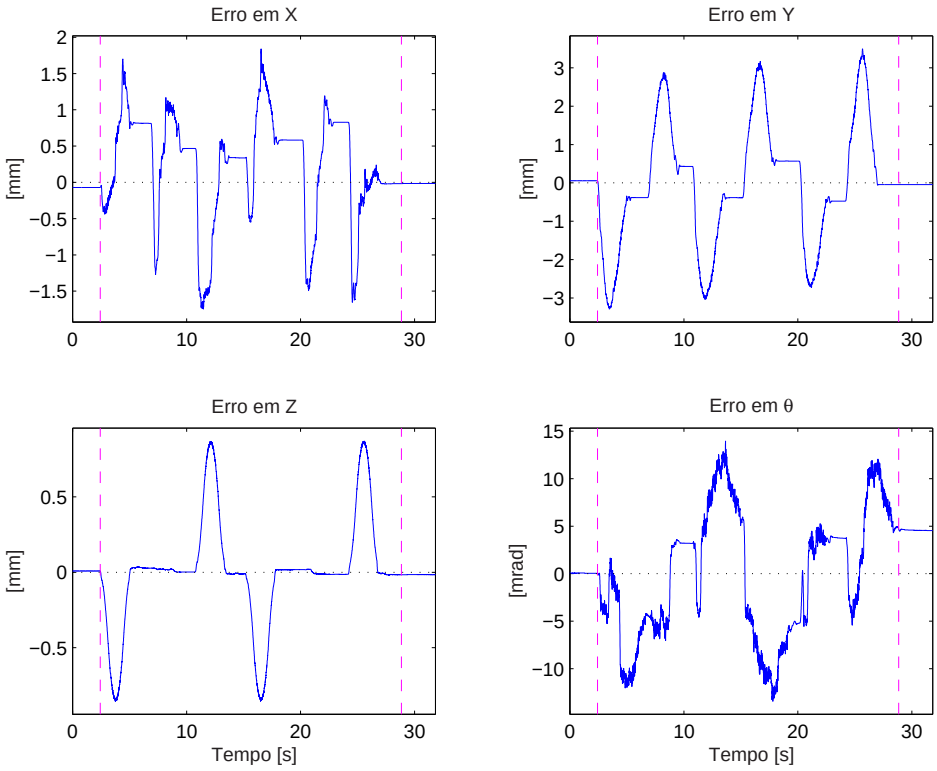


Figura 6.3: Gráficos dos erros de seguimento de trajetória para o PD-Op.

6.2.4 Testes do Controlador PID-Op

Os ganhos utilizados para este controlador são apresentados na tabela 6.3 – este controlador também foi inicialmente sintonizado com base nos parâmetros de Ziegler-Nichols. A figura 6.4 mostra os erros de seguimento de trajetória alcançados. Os índices de desempenho alcançados são apresentados mais adiante na página 222, item 6.2.6.

Parâmetro	Direção X	Direção Y	Direção Z	Orientação θ
K_p	102.000,0	102.000,0	1.740.000,0	240,0
K_d	425,0	425,0	10.875,0	1,2
K_i	425,0	425,0	10.875,0	1,2

Tabela 6.3: Parâmetros de sintonia do controlador de posição PID no espaço operacional.

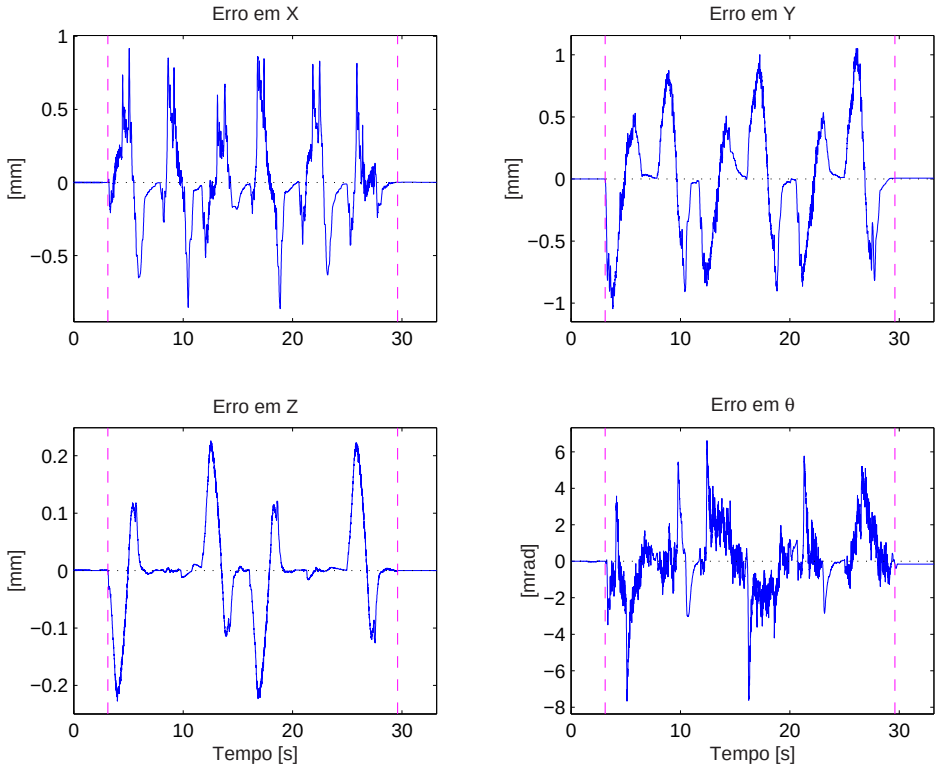
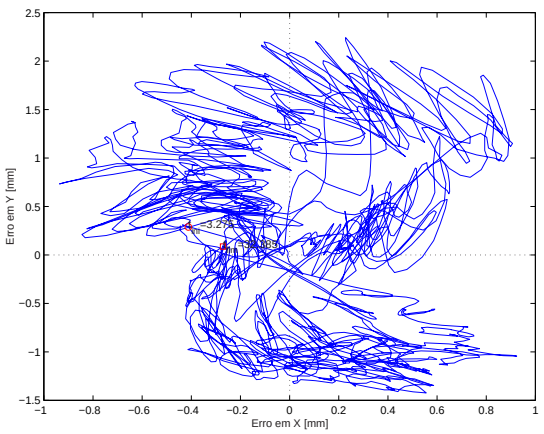


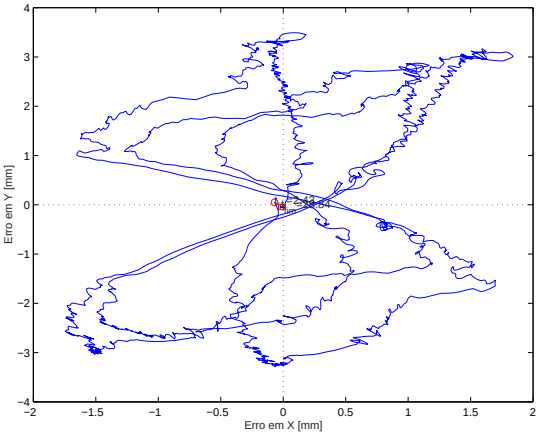
Figura 6.4: Erros de seguimento de trajetória para PID-Op.

6.2.5 Erros de Seguimento em XY dos controladores sobre o Espaço Operacional

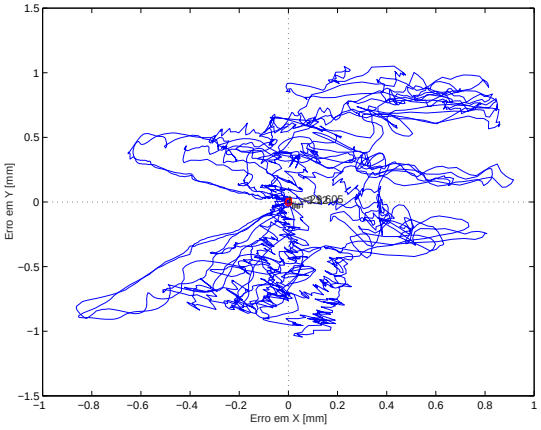
A figura 6.5 permite comparar simultaneamente os resultados obtidos pelos 3 controladores em relação aos erros de seguimento de trajetória cometidos em X e Y ao longo do tempo (gráfico de $\tilde{x}(t) \times \tilde{y}(t)$). Percebe-se que o controlador por Jacobiana Inversa desenvolve uma trajetória com menores erros de seguimento de trajetória no tempo e de forma menos ruidosa (vibrante) que o PD-Op. Entretanto, como esperado devido a ação integral, o controlador PID-Op foi o que desenvolveu menores erros de seguimento de trajetória durante a execução de toda a trajetória ($Máx(|\tilde{x}(t)|)$ e $Máx(|\tilde{y}(t)|) < 1[mm]$).



(a) Controlador por J_A^{-1}.



(b) Controlador PD-Op.



(c) Controlador PID-Op.

Figura 6.5: Gráficos de $\tilde{x} \times \tilde{y}$ ao longo do tempo.

6.2.6 Índices de Desempenho dos controladores sobre o Espaço Operacional

A tabela 6.4 apresenta os índices de desempenho alcançados para cada um dos controladores testados.

R1) $\tilde{q}_f$

Controlador.	X	Y	Z	θ
J_A^{-1}	-271.00μ	87.00μ	-340.00μ	9.53m
PD-Op	-17.00μ	-43.00μ	-16.00μ	4.71m
PID-Op	2.00μ	6.00μ	$<10^{-9}$	-176.00μ
Unidades	[m]	[m]	[m]	[rad]

R2) $MAX\{\tilde{x}\}$

Controlador	X	Y	Z	θ
J_A^{-1}	-936.00μ	2.24m	3.47m	29.98m
PD-Op	1.84m	3.49m	870.00μ	13.96m
PID-Op	916.00μ	1.05m	-227.00μ	-7.66m
Unidades	[m]	[m]	[m]	[rad]

R3) $IAE\{\tilde{x}\}$

Controlador	X	Y	Z	θ
J_A^{-1}	1.33	2.80	7.45	44.54
PD-Op	3.41	6.50	1.02	31.44
PID-Op	1.08	1.79	254.61m	6.78
Unidades	[m]	[m]	[m]	[rad]

R4) $MED\{\tilde{x}\}$

Controlador	X	Y	Z	θ
J_A^{-1}	5.00μ	116.57μ	346.17μ	414.63μ
PD-Op	238.04μ	-18.30μ	5.62μ	-105.57μ
PID	-1.05μ	-3.51μ	-30.96n	31.10μ
Unidades	[m]	[m]	[m]	[rad]

R5) $MSE\{\tilde{x}\}$

Controlador	X	Y	Z	θ
J_A^{-1}	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$
PD-Op	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$
PID-Op	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$	$<10^{-9}$
Unidades	[m^2]	[m^2]	[m^2]	[rad^2]

R6) $VAR\{\tilde{x}\}$

Controlador	X	Y	Z	θ
J_A^{-1}	78.27n	444.02n	2.19μ	76.38μ
PD-Op	558.17n	2.55μ	120.39n	45.67μ
PID-Op	85.33n	196.38n	6.62n	3.29μ
Unidades	[m^2]	[m^2]	[m^2]	[rad^2]

R7) $MAX\{\tau\}$

Controlador	0	1	2	3
J_A^{-1}	32.20	-22.59	136.35	3.86
PD-Op	31.55	13.62	172.48	1.71
PID-Op	31.89	-13.68	178.17	-1.57
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

R8) $MED\{\tau\}$

Controlador	0	1	2	3
J_A^{-1}	2.28	-60.98m	13.84	5.37m
PD-Op	2.71	-1.26	9.84	14.07m
PID-Op	-959.95m	1.93	10.73	-148.85m
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

R9) $VARi\{\tau\}$

Controlador	0	1	2	3
J_A^{-1}	286.43m	824.96m	467.66m	231.85m
PD-Op	671.15m	451.34m	8.69	95.92m
PID-Op	748.40m	491.50m	9.01	98.63m
Unidades	[Nm]	[Nm]	[Nm]	[Nm]

Tabela 6.4: Indices de desempenho atingidos para os controladores operado sobre o espaço operacional.

6.2.7 Discussão sobre Controladores de Posição no Espaço Operacional

a) O controlador PD, devido a sua ação derivativa, é o que apresenta maiores erros de seguimento de trajétoria, maiores até que o controlador por Jacobiana Inversa (J_A^{-1}).

b) A ação derivativa do controlador PD-Op implicou em maiores erros de seguimento de trajetória.

c) O PID-Op induziu um comportamento super amortecido ao sistema. Percebe-se que o

integrador demora um pouco mais para anular o erro em regime permanente (últimos segundos dos gráficos de erros e torques enviados às juntas), mas o o erro de seguimento de trajetória baixa consideravelmente em comparação com o PD-Op.

d) Controlares que operam diretamente sobre o espaço operacional exigem muito mais atenção do usuário, pois posições próximas do limite do espaço de trabalho do robô (o círculo mais externo tracejado mostrado na figura 6.1, pág. 217) podem paralisar o robô. Pôde ser comprovado através de testes práticos, que comandar o robô para um raio de abertura próximo de 49.8 [cm] (à cerca de 2 [mm] do raio máximo possível que é de 50 [cm] – isto implica em estender quase ao máximo as duas primeiras juntas do robô), os controladores começam a fazer o robô vibrar próximo desta posição limite. Neste caso, o robô se encontra muito próximo de uma **condição de singularidade**, na qual, o ângulo entre a junta 0 e 1 deste robô (na notação de Denavith-Hartenberg, equivale ao ângulo da junta 1), se aproxima de 0 (zero) graus, o que implica em: $sin(q_1) \cong 0$ (cálculo utilizado para determinar a matriz Jacobiana) e assim a matriz Jacobiana perde posto nesta condição. Na prática, ângulos entre a junta 0 e 1, menores que 10 graus implicaram problemas para este tipo de controladores. Nesta condição, $q_1 \cong 10^o$, o controlador não consegue mais comandar um torque não nulo para a junta 1, e a função de segurança entra em ação (erro limite de seguimento de trajetória atingido) pois o gerador de trajetória continua gerando referências para as juntas do robô, mas o robô permanece parado. O robô só consegue sair desta situação, com o usuário especificando novos comandos de movimentação diretamente no espaço de juntas, selecionando também um controlador que opere sobre o espaço de juntas, forçando um torque para a junta 1 de forma a afastar o robô da condição de singularidade.

6.3 Controladores Convencionais de Força

A seguir são apresentados os resultados obtidos com os testes efetuados com os controladores convenconais de força (malha de controle de força).

6.3.1 Testes Propostos

Foram organizados 3 tipos de testes para verificar o controlador híbrido de força/posição:

1) Deslocamento sobre superfície de borracha;

2) Deslocamento sobre isopor;

3) Deslocamento sobre um plano inclinado de madeira;

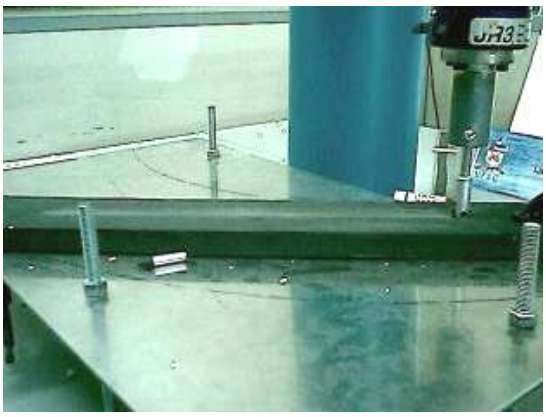
Todos estes tipos de teste, especificaram controle de força em Z e de posição nas direções X, Y e θ. Sendo que para controle de força é especificado uma força de contato que varia entre 3.0 e 7.5 [N]. Com limiar para contato especificado entre 0.5 e 1.0 [N]. Para a malha de controle de posição foi especificada uma velocidade máxima de deslocamento no plano XY de 10 [cm/s] e apenas uma correção na orientação inicial e final do robô (ângulo θ). A figura 6.6 dá uma idéia visual dos testes previstos.

Foram previstos testes usando para controlador de posição: PD-Op e PID-Op (de forma a verificar como estes controladores rejeitam resistências à movimentação em XY e θ) e controladores Proporcional e PI de força na direção Z. O último teste, de controle de força sobre um plano inclinado de madeira objetivava verificar se o controlador de força seria capaz de fazer o robô subir ou descer o plano inclinado conforme a posição final de movimentação em XY que houvesse sido especificada. Todos os controladores operaram com período de amostragem de $T_s = 2$ [ms] e filtro passa-baixa (FPB) de primeira ordem para o sinal captado pelo sensor de força com a freqüência de corte tendo sido fixada em: $f_c = 40$ [Hz].

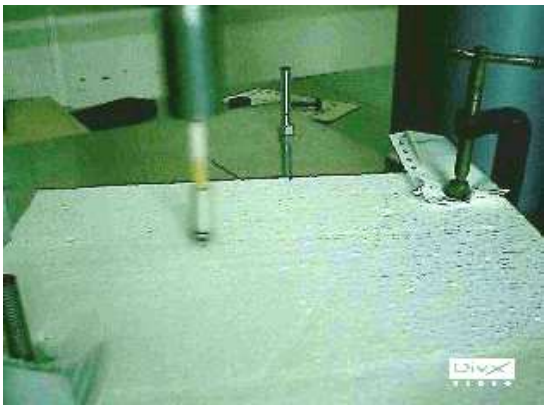
Como o controlador de força somente á ativado depois que certo limiar de contato é detectado pelo sensor de força, primeiramente foram realizados testes para comprovar a estratégia proposta para chavear entre controle de posição e controle de força (próximo item).

6.3.2 Teste da Estratégia de Contato

A figura 6.7 mostra o teste realizado para comprovar a eficácia da estratégia adotada para chavear o controle de força apenas enquanto está sendo detectado contato acima de certo limiar (no caso, $f_{THR} = 0.75$ [N] $\cong 75$ [gramas]). O robô iniciou a trajetória afastado do meio, isto é, sem contato inicial com o meio. Estava previsto, pela forma como foi especificado a



(a) Teste sobre borracha.



(b) Teste sobre isopor.

(c) Teste sobre plano inclinado de madeira.

Figura 6.6: Testes previsto para os controladores de força.

movimentação do robô (note o ligeiro plano inclinado em Z), que o contato seria realizado durante a execução da trajetória ($vel_x = 10$ [cm/s]). Como pode ser visto através da figura 6.8, a partir do instante em que $f_{Sens_z} > 0.75$ [N], o algoritmo de controle para a junta 2 foi chaveado do controlador de posição PD-Op que estava sendo adotado para o controlador de força Proporcional na direção Z (o que havia sido selecionado para controle de força neste instante), cuja referência foi fixada em (força de contato desejada): $h_{z_d} = 5.0$ [N].

Como pode ser percebido pela figura 6.8 esta simples estratégia para chavear o controlador de força se mostrou suficiente para a aplicação pretendida neste trabalho. O única restrição é que, se durante a execução de uma tarefa envolvendo força de contato, for perdido o contato com o meio, a estratégia proposta chaveia abruptamente para controle de posição e neste caso, o robô é forçado a ir para uma posição em Z definida pelo gerador de trajetória (linha pontilhada na figura 6.8(a)). Isto implica em que o usuário deve estar atento

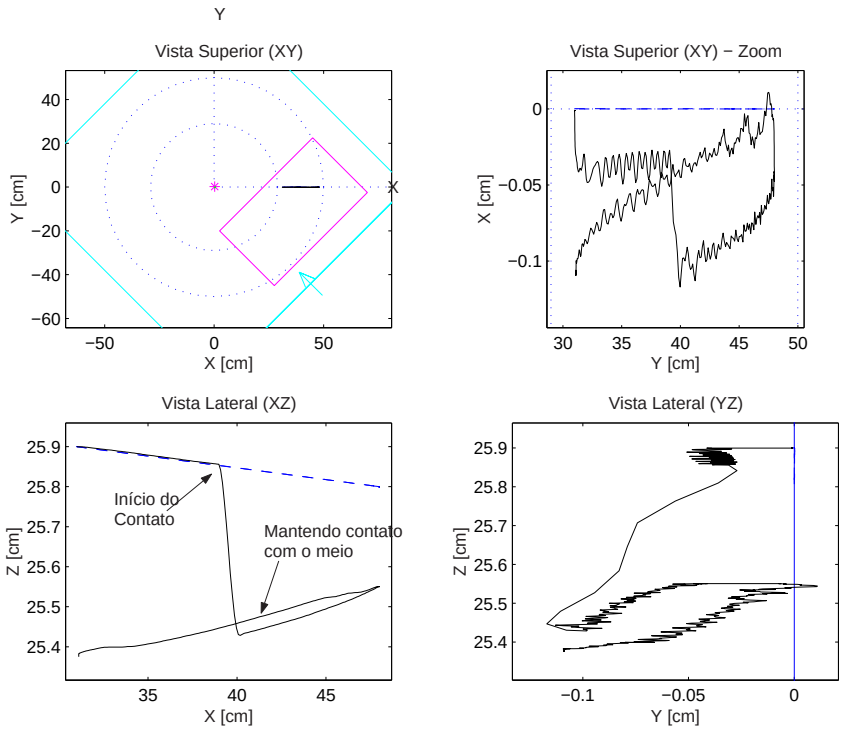
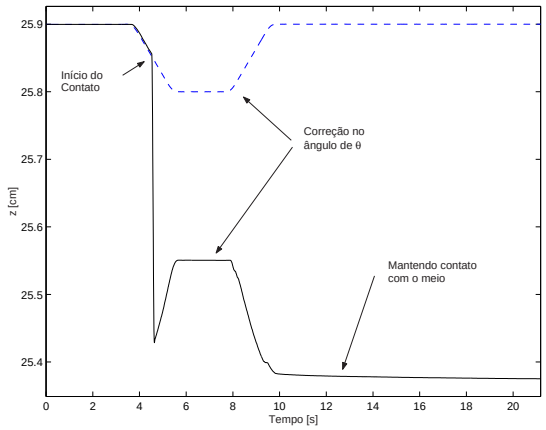


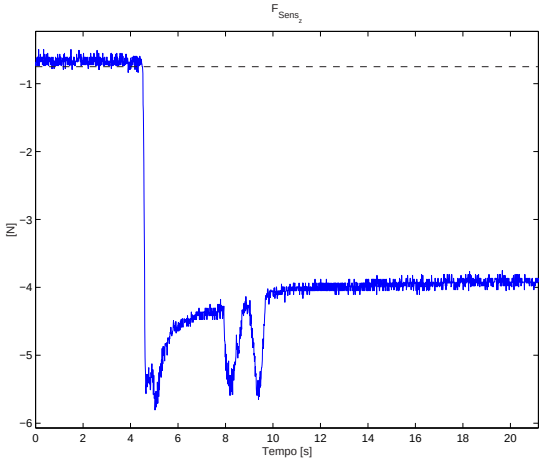
Figura 6.7: Trajetória do teste da Estratégia de Contato.

ao especificar os pontos iniciais e finais de uma trajetória no espaço operacional. Outro detalhe é que dependendo do controlador de posição selecionado, este chaveamento pode ser abrupto ao ponto de no instante seguinte de chaveamento de controle de força para controle de posição na direção Z, seja enviado o torque máximo para a junta 2, devido ao erro de seguimento de trajetória envolvido com o controlador de posição que trabalha com a posição alcançada em Z $\times$ altura desejada em Z (especificada pelo gerador de trajetórias). Eventualmente este torque abrupto faz com o que o robô recupere o contato com o meio.

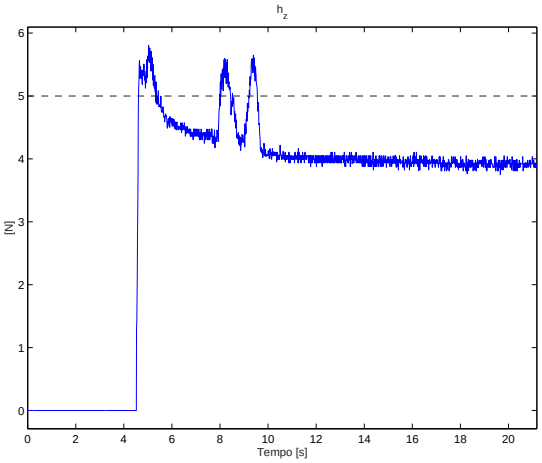
Teste similar foi feito "equilibrando-se" o robô sobre o dedo, figura 6.9. Neste caso, o gerador de trajetória permaneceu estático, tendo sido apenas especificada uma altura fixa conveniente em Z, limiar de contato igual à $f_{THR} = 1.0$ [N] $\cong 100$ [gramas] e força de contato desejada em $h_{d_z} = 5.0$ [N] $\cong 500$ [gramas]. Este teste consistiu simplesmente em verificar a estratégia adotada para chavear controle de força/posição na direção Z com base na pressão realizada pelo dedo contra o sensor de força (controle de força na direção Z apenas). Neste teste foi utilizado o controlador Proporcional simples de força na direção Z.



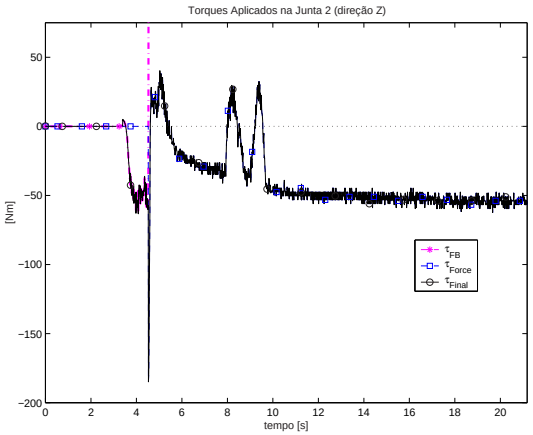
(a) Trajetória desenvolvida na direção Z.



(b) Forças de contato na direção Z do sensor.



(c) Contato desenvolvido na direção Z (espaço operacional).



(d) Torques enviados para a junta 2 (direção Z).

Figura 6.8: Resultados do teste da Estratégia de Contato.

Figura 6.9: ”Teste do dedinho”.

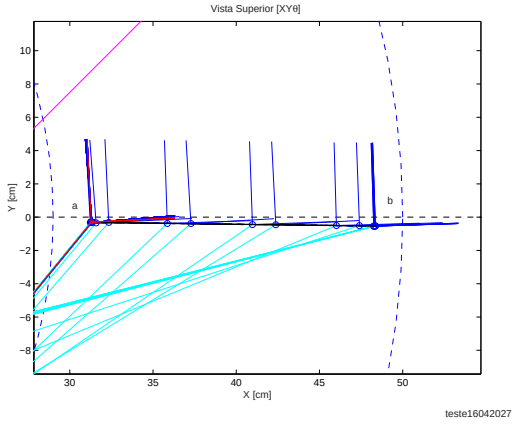
A seguir são apresentados os testes realizados com os controladores de força.

6.3.3 Teste sobre Borracha

A trajetória utilizada para este teste e seus dados pode ser vista na figura 6.10. O robô sai da posição (a), se desloca para a posição (b) e volta para a posição (a) desenvolvendo uma velocidade máxima de 10 [cm/s] na direção X. Foi especificado como força de contato a ser mantida, $h_{d_z} = 7.5$ [N] e limiar para força de contato: $f_{THR} = 1.0$ [N].

Dados	X	Y	Z	θ
x_a	0.3129	-0.0030	0.2313	0.0502
x_b	0.4827	-0.0054	0.2341	0.0297
	[m]	[m]	[m]	[rad]
$\dot{x}_{máx}$	0.1001	0.0014	-0.0017	-0.0475
	[m/s]	[m/s]	[m/s]	[rad/s]
$\ddot{x}_{máx}$	0.3153	-0.0205	-0.0196	-0.2225
	[m^2/s]	[m^2/s]	[m^2/s]	[rad^2/s]

(a) Dados da trajetória.

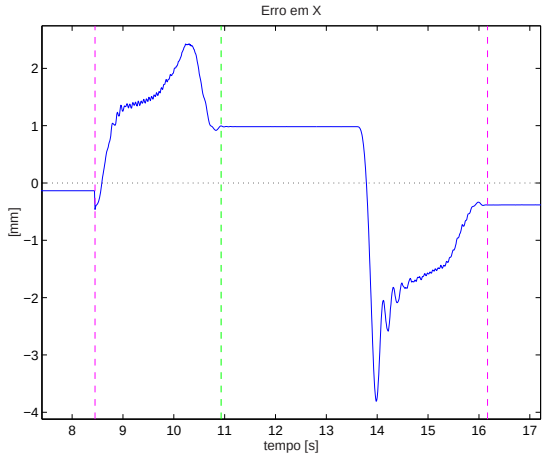


(b) Trajetória desenvolvida.

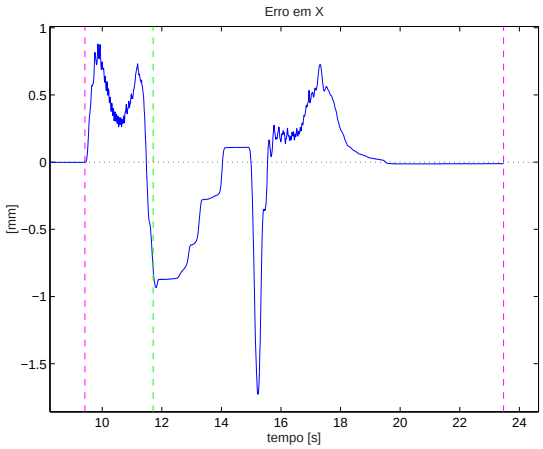
Figura 6.10: Dados do teste sobre a borracha.

A figura 6.11 mostra os erros de posicionamento desenvolvidos pelos controladores de posição PD-Op e PID-Op nas direções não restringidas para o movimento.

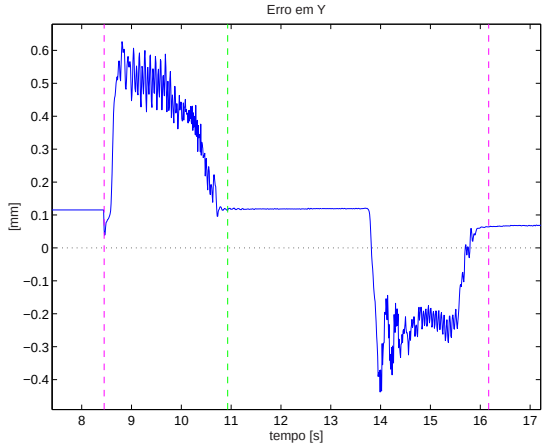
A figura 6.12 mostra aproximadamente a profundidade do contato realizado sobre a superfície de borracha. A altura inicial nula, referência estabelecida artificialmente, equivale a altura assumida pelo robô quando este inicia o seu movimento sobre a borracha. Esta referência (equação linear) foi estabelecida com base na ligeira diferença em altura (eixo Z), assumida pelo robô entre os pontos iniciais e finais do deslocamento realizados em XY, pontos nos quais a velocidade de deslocamento XY do robô se torna nula. Desta forma, tentou-se compensar o leve desnível verificado em Z (3.7 [mm]) e mostrar num gráfico, uma estimativa do tanto que o robô afunda na peça ou não. Na realidade esta peça e todas as demais testadas não permanecem perfeitamente niveladas na horizontal (ângulo nulo com relação ao plano XY de coordenadas de base do robô). Alturas "positivas" significam que o robô afundou mais na peça e alturas "negativas" significam que o robô levantou da peça (podendo ser resultado de alguma irregularidade na peça) mas não necessariamente que foi



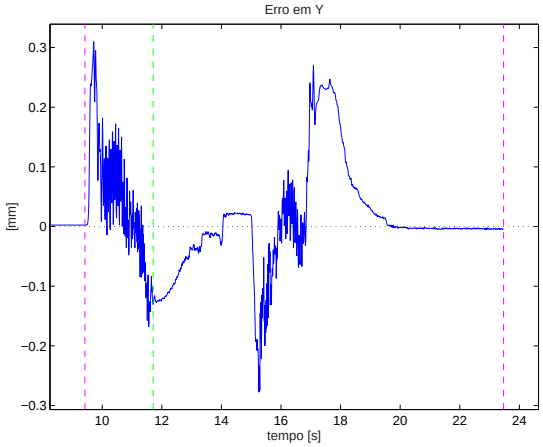
(a) Erros em X para PD-Op.



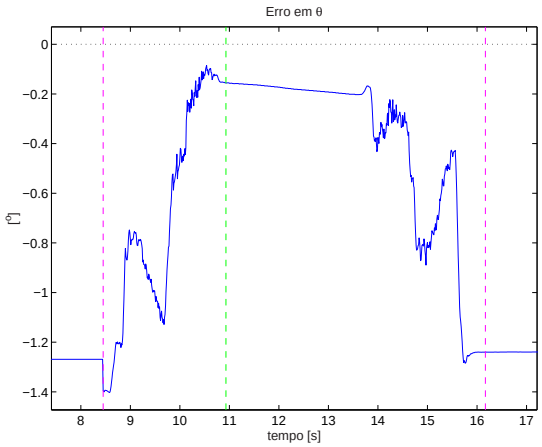
(b) Erros em X para PID-Op.



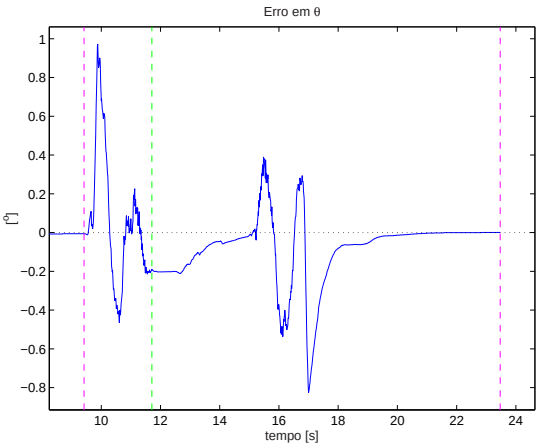
(c) Erros em Y para PD-Op.



(d) Erros em Y para PID-Op.



(e) Erros em θ para PD-Op.

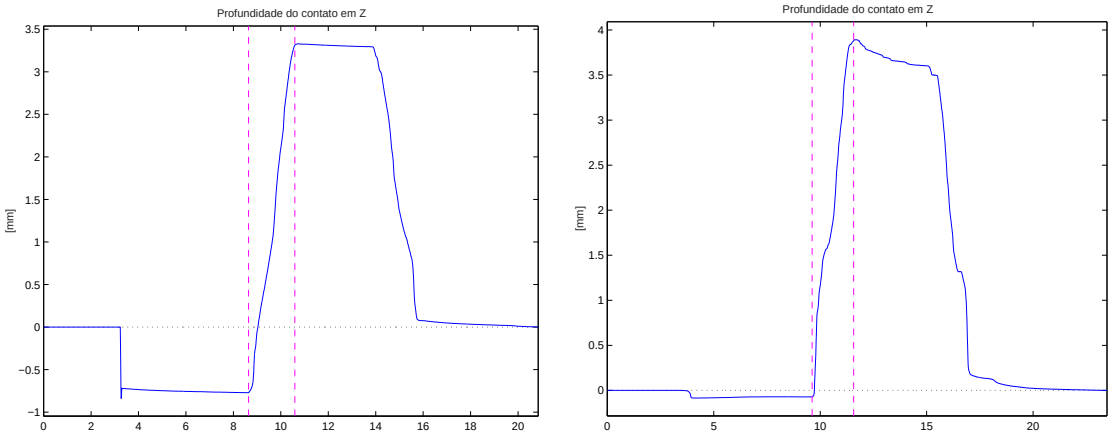


(f) Erros em θ para PID-Op.

Figura 6.11: Erros de Posição para o teste sobre a borracha.

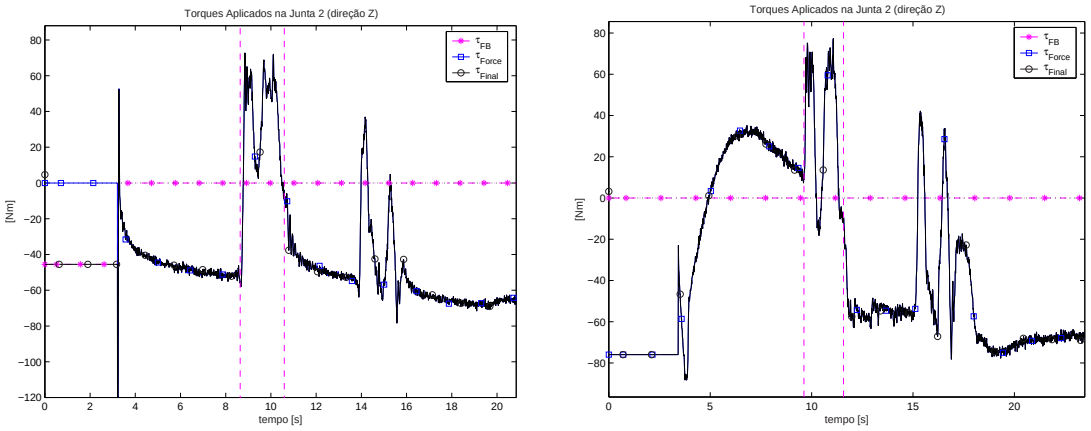
perdido o contato com a peça.

As figuras 6.14 e 6.13 mostram os resultados obtidos pelos controladores Proporcional de PI da malha de controle de força.



(a) Controlador Proporcional de força. (b) Controlador PI de força.

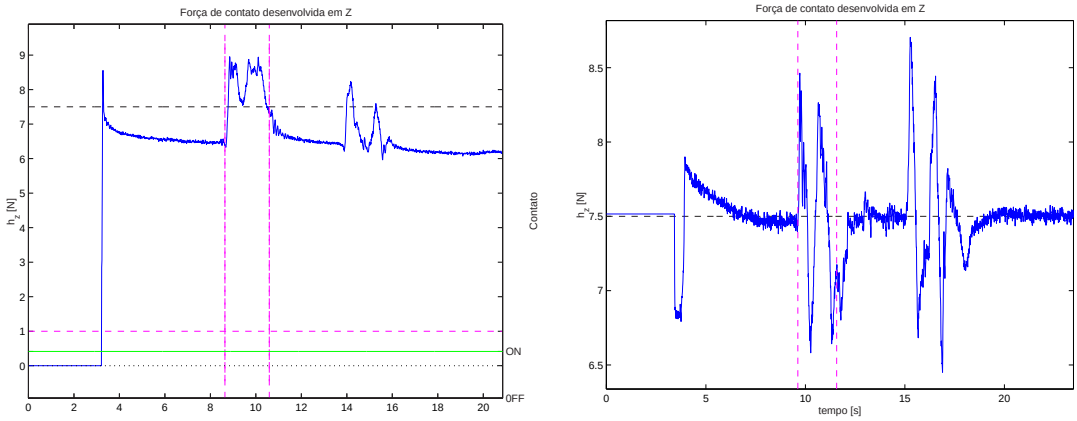
Figura 6.12: Profundidade dos contatos para teste sobre borracha.



(a) Controlador Proporcional de força. (b) PI de força.

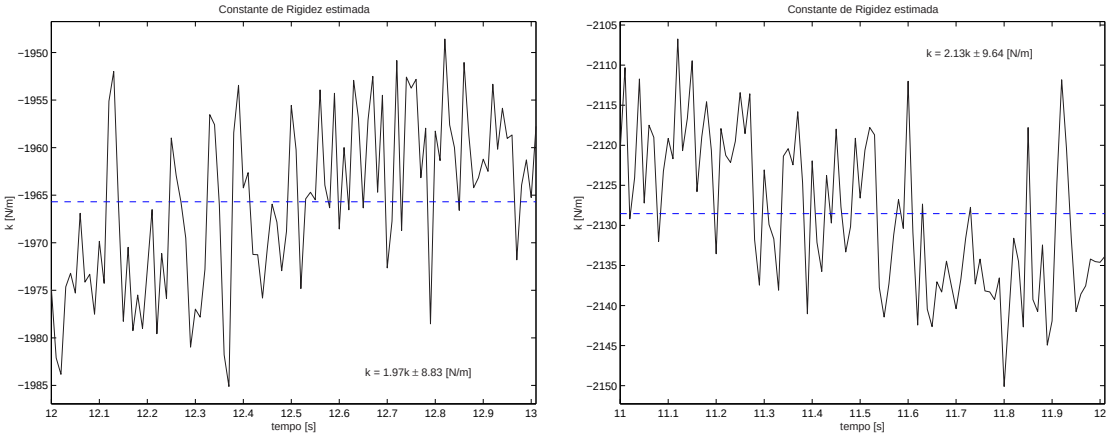
Figura 6.13: Torques aplicados à junta 2 no teste sobre a borracha.

Os ganhos utilizados para os controladores P e PI de força apresentados nos testes foram: $K_p = 50$ para o controlador Proporcional e $K_p = 50$ e $K_i = 0.5$ para o controlador PI. Estes valores foram obtidos experimentalmente (método da tentativa e erro). Notar que é difícil se estimar a constante de rigidez do meio (figura 6.15) porque é difícil determinar a altura em Z onde inicia o contato.



(a) Controlador Proporcional de força. (b) Controlador PI de força

Figura 6.14: Forças de contato desenvolvidas no teste sobre borracha.



(a) P de força, $\hat{k} = 1.97K \pm 8.83$ (N/m). (b) PI de força, $\hat{k} = 2.13K \pm 9.64$ (N/m).

Figura 6.15: Estimativa da constante de rigidez da borracha.

Conclusões preliminares: dos gráficos anteriores se pode concluir que:

a) Referente à malha de controle de posição: O controlador PD-Op não é capaz de rejeitar perturbações tão bem quando o PID-Op (figura 6.11). Repare no erro de orientação final do robô (ângulo θ) que no caso do controlador PD-Op não baixa de 1.2 [graus]. Dai conclui-se a importância da ação integral para garantir erro nulo em regime permanente para as posições finais desejadas.

b) Referente à malha de controle de força: **é nítida a necessidade de ação integral na malha de controle de força** (figura 6.14). Sem a ação integral, um simples ganho proporcional na malha de controle de força que comanda a junta 2 não é capaz de vencer o atrito envolvido com a movimentação desta junta. Durante os ensaios verificados, percebeu-se que valores de torque abaixo de 67 [N] são insuficientes para gerar movimento na junta 2. Já o controlador PI, devido a ação integral, consegue desenvolver o torque necessário para vencer o atrito envolvido com a movimentação desta junta, e assim consegue manter a força de contato desejada a mais próxima possível da referência além de garantir erro nulo em regime permanente para a força de contato desejada. De qualquer forma além de diminuir o problema de regulação, a força de contato oscilou entre os extremos de 16% à -15% para o controlador PI, enquanto que para o controlador Proporcional apenas, a força de contato oscilou entre picos de $\pm 20\%$ em relação à força de contato desejada.

c) Referente ainda à malha de controle de força: a única desvantagem com o uso do controlador PI está no fato deste gerar respostas mais abruptas à variação de contato, gerando ações de controle um pouco mais oscilatórias. Maior ganho proporcional implica em picos de torque maiores sendo gerados para a junta 2 e conseqüentemente maiores erros com relação à força de contato desejado, além do robô penetrar mais que o necessário no meio (podendo causar deformação no meio). Ganhos integrais muito maiores saturam rapidamente a ação de controle num dos extremos do maior valor de torque que pode ser enviado para a junta 2, implicando em aumentos nos erros de seguimento de força desejada além de respostas mais oscilatórias e eventualmente perdas de contato com a peça devido às reações mais abruptas que induz contra o meio. Foi percebido que uma leve ação integral na malha de controle de força é altamente desejável e muito bem vinda.

6.3.4 Teste sobre Isopor

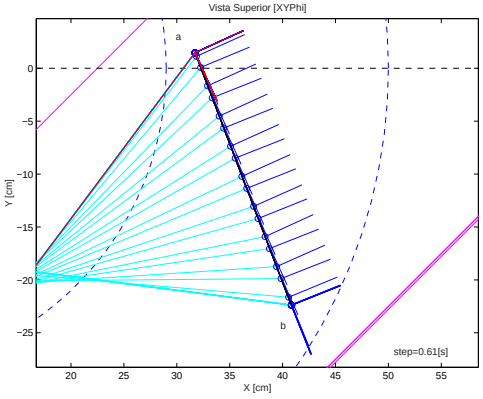
A trajetória utilizada para este teste é semelhante a adotada anteriormente; seus dados são mostrados na tabela 6.5. O robô sai da posição (a), se desloca para a posição (b) e volta para a posição (a) desenvolvendo uma velocidade máxima de 5 [cm/s] na direção Y. Foi especificada como força de contato à ser mantida, $h_{d_z} = 3.0$ [N] $\cong 300$ [gramas] e limiar para força de contato: $f_{THR} = 1.0$ [N]. Uma segunda trajetória de teste foi prevista na qual a velocidade de deslocamento sobre o plano XY foi aumentada para 10 [cm/s] a fim de verificar se aumentam as oscilações na força de contato com o aumento da velocidade de deslocamento sobre o meio. Os dados desta segunda trajetória aparecem na figura 6.16 – notar que a distância percorrida no plano XY aumentou em relação ao teste anterior (movimento em diagonal). Nos resultados apresentados foi utilizado para a malha de controle de posição o PID-Op e para a malha de controle de força, o controlador PI, com ganhos variando entre 30 e 50 e velocidade de deslocamento no plano XY de 5 à 10 [cm/s].

Dados	X	Y	Z	θ
x_a	0.31	0.045	0.2312	-0.1148
x_b	0.48	-0.068	0.2297	-0.1186
	[m]	[m]	[m]	[rad]
$\dot{x}_{máx}$	0.05	0.007	-0.0005	-0.0154
	[m/s]	[m/s]	[m/s]	[rad/s]
$\ddot{x}_{máx}$	0.22	-0.044	-0.0192	0.1225
	[m^2/s]	[m^2/s]	[m^2/s]	[rad^2/s]

Tabela 6.5: Dados da primeira trajetória do teste sobre o isopor.

Dados	X	Y	Z	θ
x_a	0.3172	0.0147	0.2300	-1.1438
x_b	0.4082	-0.2238	0.2289	-0.1922
	[m]	[m]	[m]	[rad]
$\dot{x}_{máx}$	-0.0178	0.0468	-0.0002	-0.0841
	[m/s]	[m/s]	[m/s]	[rad/s]
$\ddot{x}_{máx}$	0.0825	-0.2094	0.0141	0.2945
	[m^2/s]	[m^2/s]	[m^2/s]	[rad^2/s]

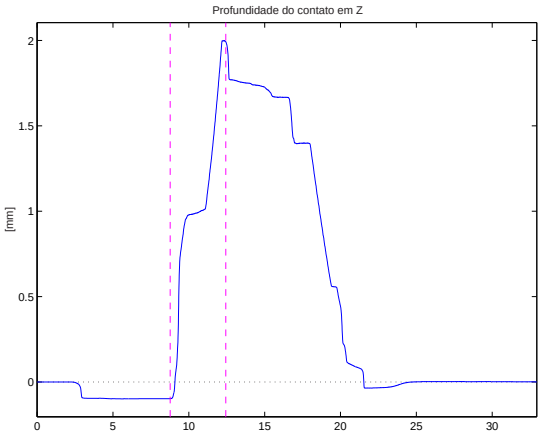
(a) Dados da trajetória.



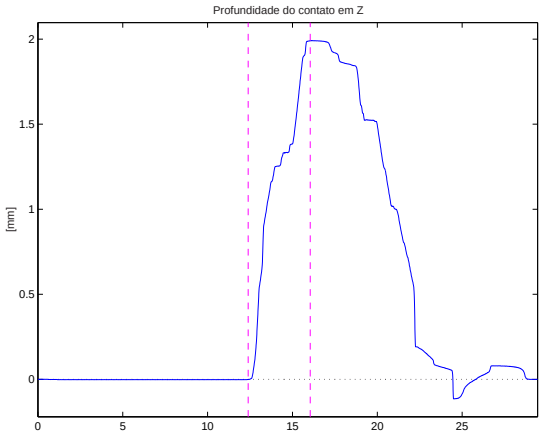
(b) Trajetória executada.

Figura 6.16: Dados da segunda trajetória de teste sobre o isopor.

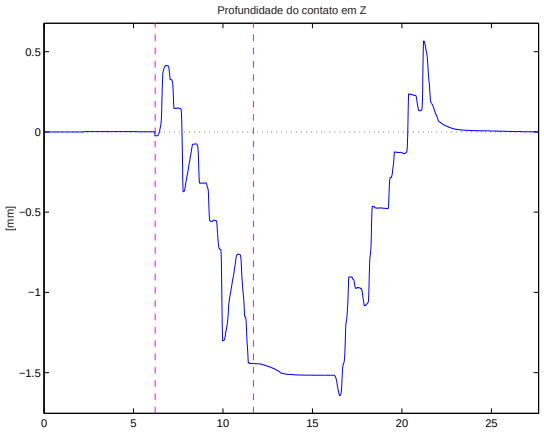
Os resultados obtidos com estes testes aparecem nas figuras 6.17, 6.18 e 6.19.



(a) $K_p = 30$ e $vel_y = 5$ (cm/s).



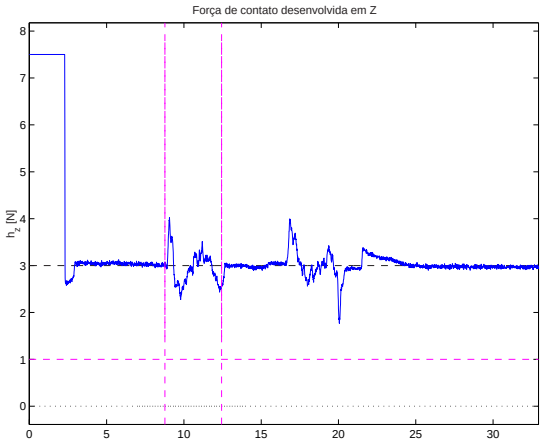
(b) $K_p = 50$ e $vel_y = 5$ (cm/s).



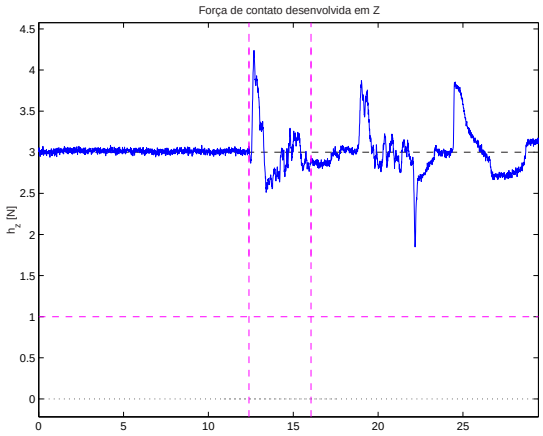
(c) $K_p = 50$ e $vel_y = 10$ (cm/s).

Figura 6.17: Profundidade dos contatos para teste sobre isopor.

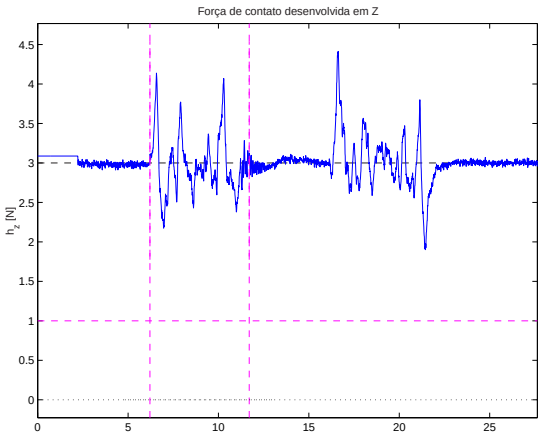
A constante de rigidez estimada para este material é mostrado na figura 6.20.



(a) $K_p = 30$ e $vel_y = 5$ (cm/s).

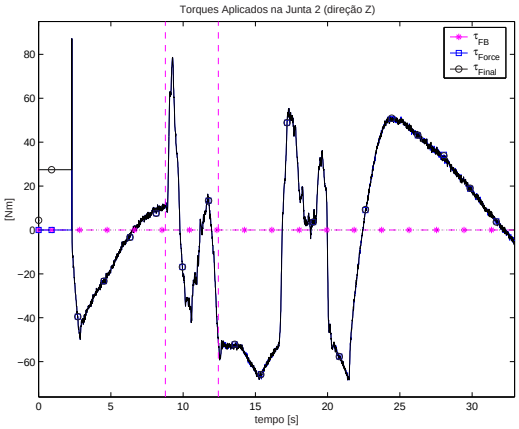


(b) $K_p = 50$ e $vel_y = 5$ (cm/s).

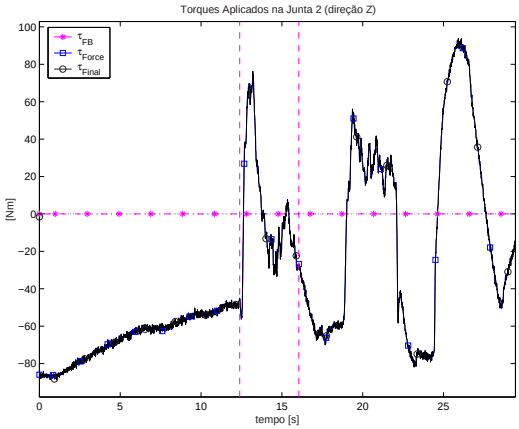


(c) $K_p = 50$ e $vel_y = 10$ (cm/s).

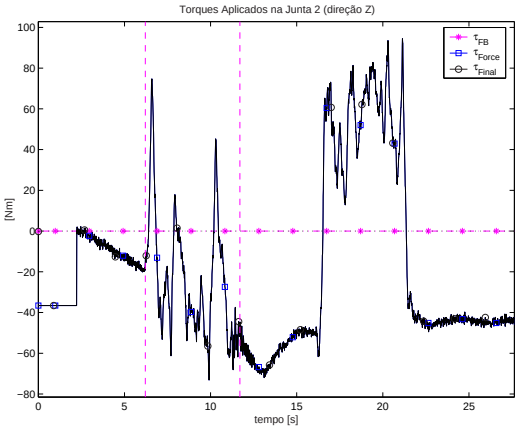
Figura 6.18: Forças de contato desenvolvidas no teste sobre o isopor.



(a) $K_p = 30$ e $vel_y = 5$ (cm/s).

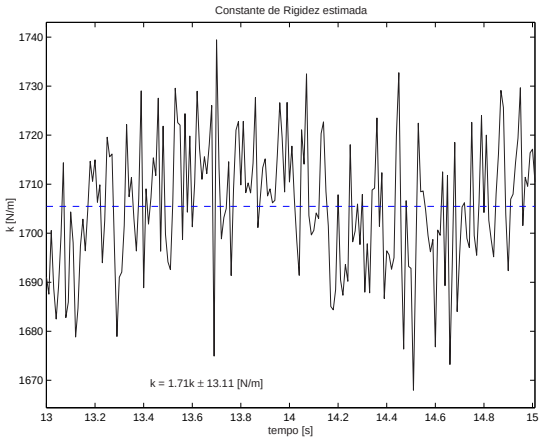


(b) $K_p = 50$ e $vel_y = 5$ (cm/s).

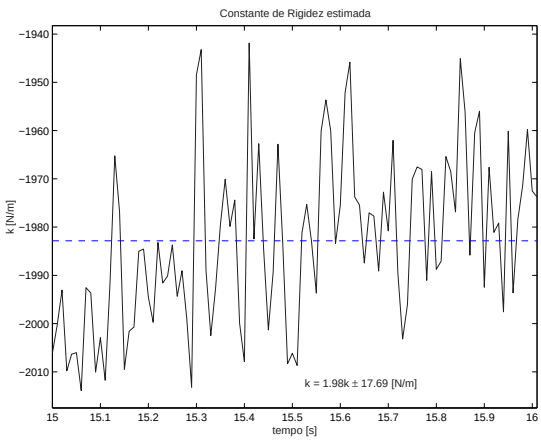


(c) $K_p = 50$ e $vel_y = 10$ (cm/s).

Figura 6.19: Torques aplicados à junta 2 no teste sobre o isopor.



(a) PI de força, $K_p = 30$, $K_i = 0.5$, $\hat{k} = 1.71K \pm 13.11$ (N/m).



(b) PI de força, $K_p = 50$, $K_i = 0.5$, $\hat{k} = 1.98K \pm 17.69$ (N/m).

Figura 6.20: Estimativa da constante de rigidez do isopor.

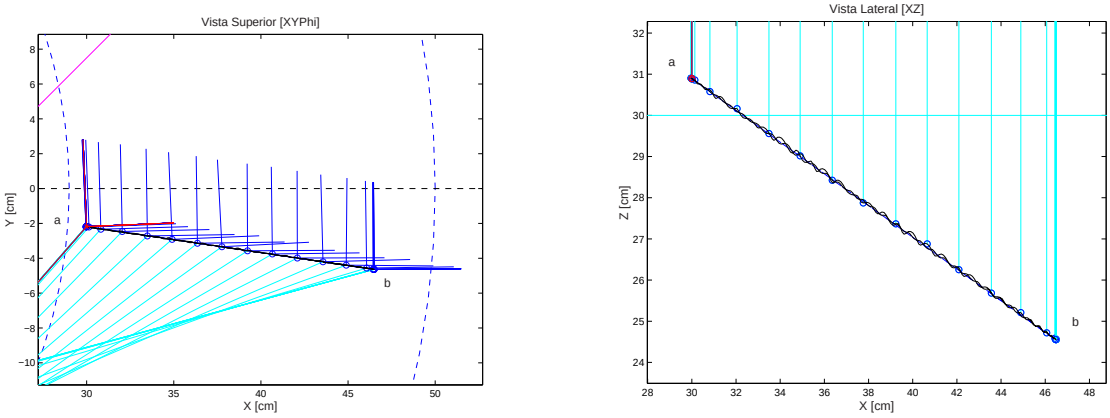
Conclusões preliminares:

a) Ganhos Proporcionais maiores implicam em contato mais acentuado com o meio (maiores profundidades de contato, figura 6.17.

b) Velocidades maiores dificultam a malha de controle de força. As forças de contato mantidas durante a trajetória se mostram mais oscilatórias (figura 6.18) assim como os torques gerados (figura 6.19). É mais fácil se manter a força de contato mais próxima dos valores desejados para velocidades mais baixas de deslocamento sobre o meio, provavelmente devido a deformação causada no meio.

6.3.5 Teste sobre o Plano Inclinado de Madeira

A figura 6.21 mostra a trajetória do teste executada sobre um plano inclinado de madeira. Para este ensaio foi utilizado o controlador PID-Op para a malha de controle de posição (realizando controle de posição em X, Y e θ) e controlador PI para a malha de controle de força, com ganhos iguais à: $K_p = 35$ e $K_i = 0.25$ e velocidade especificada para cumprir a trajetória igual à 10 [cm/s]. Os resultados do ensaio com o plano inclinado aparecem na figura 6.22.

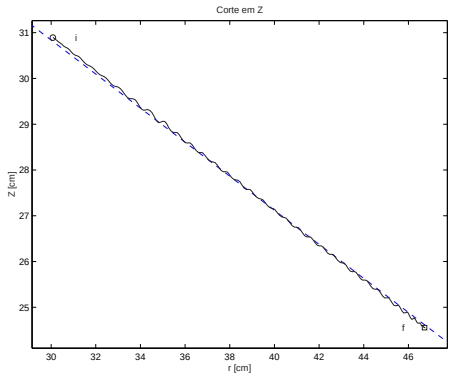


(a) Vista Superior. (b) Vista Lateral.

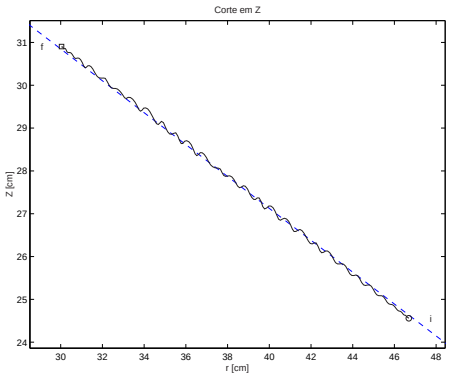
Figura 6.21: Trajetória executada sobre o plano inclinado de madeira.

Analisando os gráficos da figura 6.22 percebemos que o robô perdeu contato com o meio diversas vezes, mais durante a descida que durante a subida.

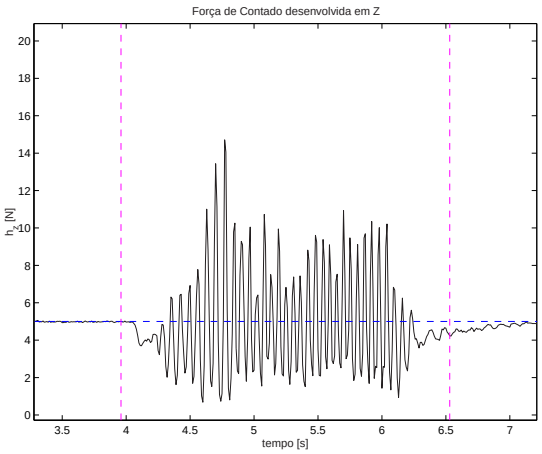
A figura 6.23 mostra um caso em que em regime permanente, com ganhos para o controlador de força em: $K_p = 50$ e $K_i = 0.5$, $f_{THR} = 1.0$ [N] e $h_{d_z} = 3.0$ [N], foram verificadas



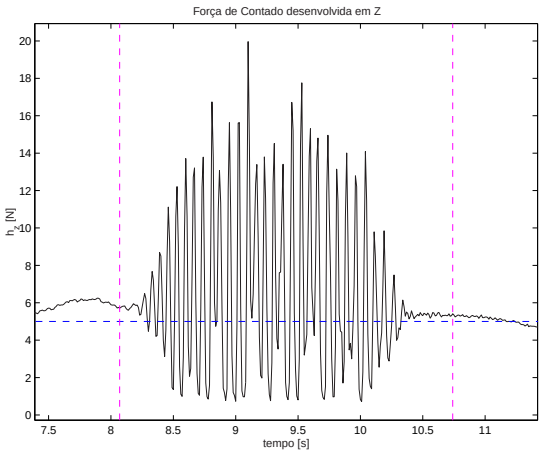
(a) Trajetória executada em Z (descida).



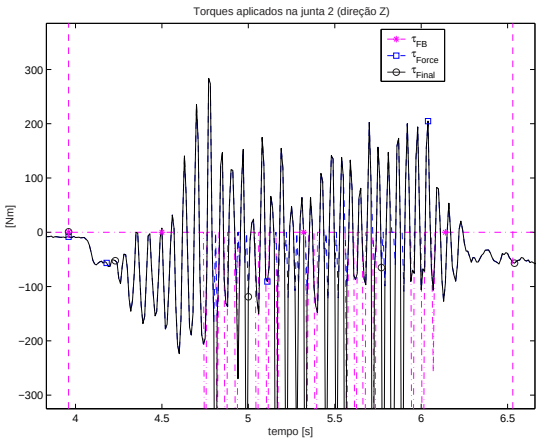
(b) Trajetória executada em Z (subida).



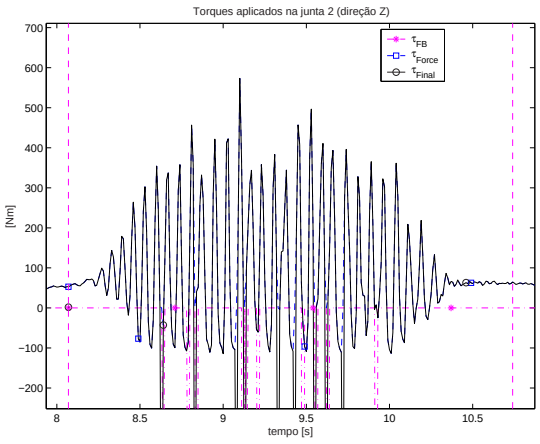
(c) Força de Contato em Z, h_z (descida).



(d) Força de Contato em Z, h_z (subida).



(e) Troques enviados para junta 2 (descida).



(f) Torques enviados para junta 2 (subida).

Figura 6.22: Resultados do teste sobre o plano inclinado.

fortes oscilações na malha de controle de força. Percebe-se pela figura fortes oscilações provocadas pelo chaveamento realizado entre o controle de força e o de posição na direção Z.

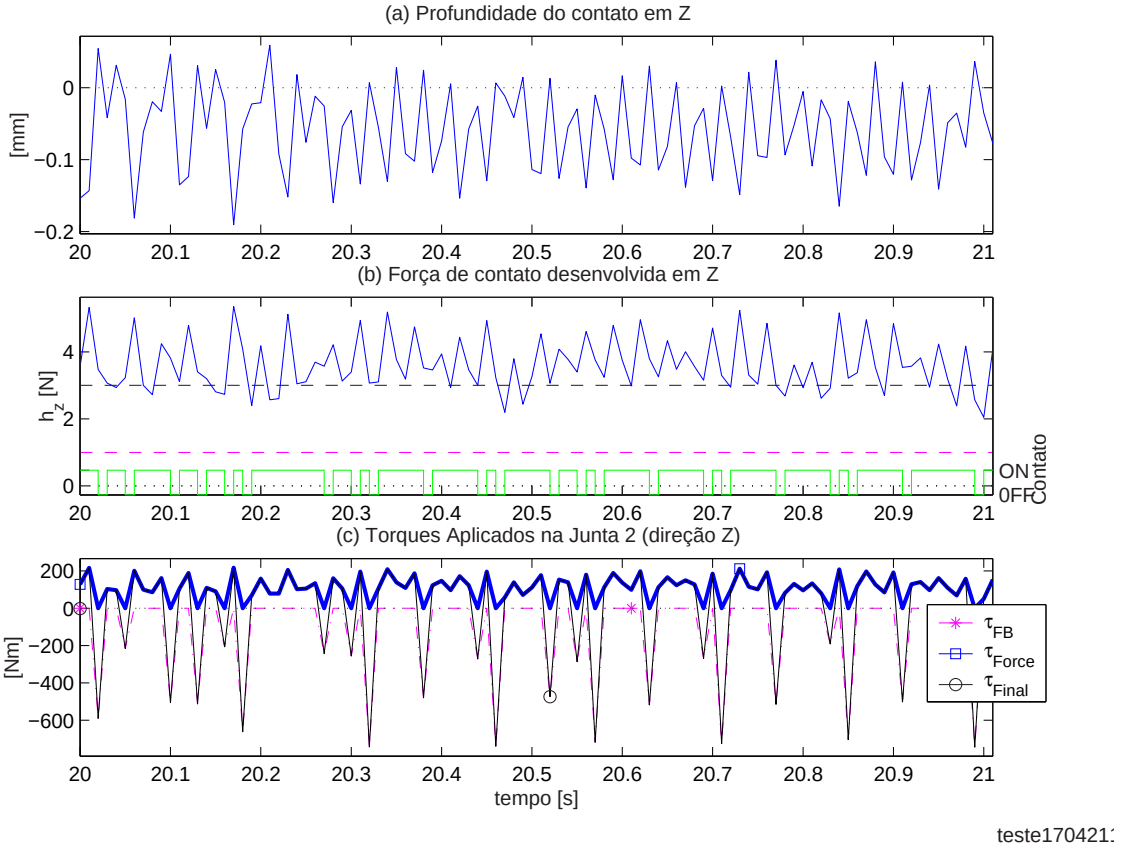


Figura 6.23: Controle de força sobre madeira (plano inclinado), com $K_p = 50$ e $K_i = 0.5$.

Os resultados com este teste apenas confirmam que o controlador de força é capaz de seguir o contorno de uma peça em Z, subindo ou descendo o plano inclinado em Z. Os resultados obtidos provavelmente seriam melhores se houvesse sido definida uma velocidade menor que 10 [cm/s] para executar a trajetória de subida ou descida da peça e se houvessem sido realizados ajustes finos para o ganho proporcional e integral do controlador de força em Z, mas o objetivo deste controlador não é realizar um seguimento de contorno de uma peça.

6.3.6 Discussão a respeito dos Controladores Convencionais de Força

a) O controlador Proporcional de Força não consegue manter a força de contato desejada contra o meio (ou não consegue anular o erro em regime permanente para a força de contato desejada); isto é, um simples controlador proporcional de força não é capaz de gerar torque suficiente na junta 2 para compensar os atritos envolvidos com pequenas movimentações nesta junta;

b) Já um controlador de força PI acompanha muito melhor a força de contato desejada durante o período de movimentação do robô sobre o meio e consegue anular o erro em regime permanente para a força de contato à ser mantida contra o meio;

c) O PD-Op de posição não consegue anular o erro em regime permanente nas direções em que existe alguma perturbação ao movimento, isto é, (é incapaz de rejeitar perturbações);

6.4 Controladores Neurais de Força

Nesta seção seguem os resultados dos testes obtidos com as 4 redes neurais avaliadas para a malha de controle de força.

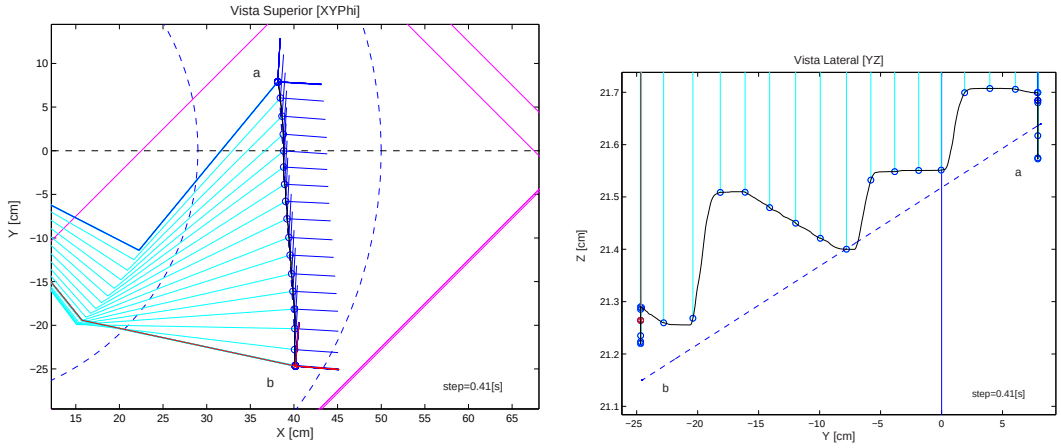
6.4.1 RN para Controle de Força (RNf-1)

Teste com rede MLP: RNf-1 (MLP)

Foi testada uma rede neural MLP para controle de força formada por 4 camadas contendo 14 x 10 x 6 x 4 neurônios, com $\eta = 0.0350$ e $\alpha = 0.5$. Esta rede foi operada em paralelo com o PI de força. Os ganhos deste controlador foram variados entre $50 \leq K_p \leq 550$ e $0.5 \leq K_i \leq 100$, e os melhores resultados (execução da trajetória sem perda de contato e *overshoots* menores) foram obtidos para: $K_p = 200$ e K_i=20. Este teste foi realizado sobre uma superfície de borracha, tendo sido especificada uma força de contato igual à $h_{d_z} = 5$ [N], com limiar de contato estabelecido em: $f_{THR} = 0.5$ [N]. A velocidade máxima de deslocamento sobre esta superfície foi fixada em 5 [cm/s] conforme esclarece a tabela 6.6 contendo os dados da trajetória executada. A figura 6.24 mostra a trajetória executada. Os resultados obtidos são apresentados na figura 6.25. Este teste terminou com a rede tendo executado 39766 ciclos de treinamento, atingindo um erro médio quadrático, MSE= 0.00110060.

Posição	Orientações			
	X	Y	Z	θ
x_a	0.3652	0.0940	0.2300	1.5800
x_b	0.4000	-0.2450	0.2250	1.5800
	[m]	[m]	[m]	[rad]
$\dot{x}$	0.0052	-0.0500	-0.0008	-0.0001
	[m/s]	[m/s]	[m/s]	[rad/s]
$\ddot{x}$	-0.0430	0.2206	0.0211	0.0069
	[m^2/s]	[m^2/s]	[m^2/s]	[rad^2/s]
Distância percorrida em XY = 34.08 [cm]				
Desnível em Z $\cong$ 5.0 [mm]				
Inclinação em Z = -0.8405 [^{o}]				

Tabela 6.6: Dados da trajetória usada para testes de controle de força sobre a superfície de borracha.



(a) Vista XYθ (b) Vista XZ

Figura 6.24: Trajetória executada sobre a borracha.

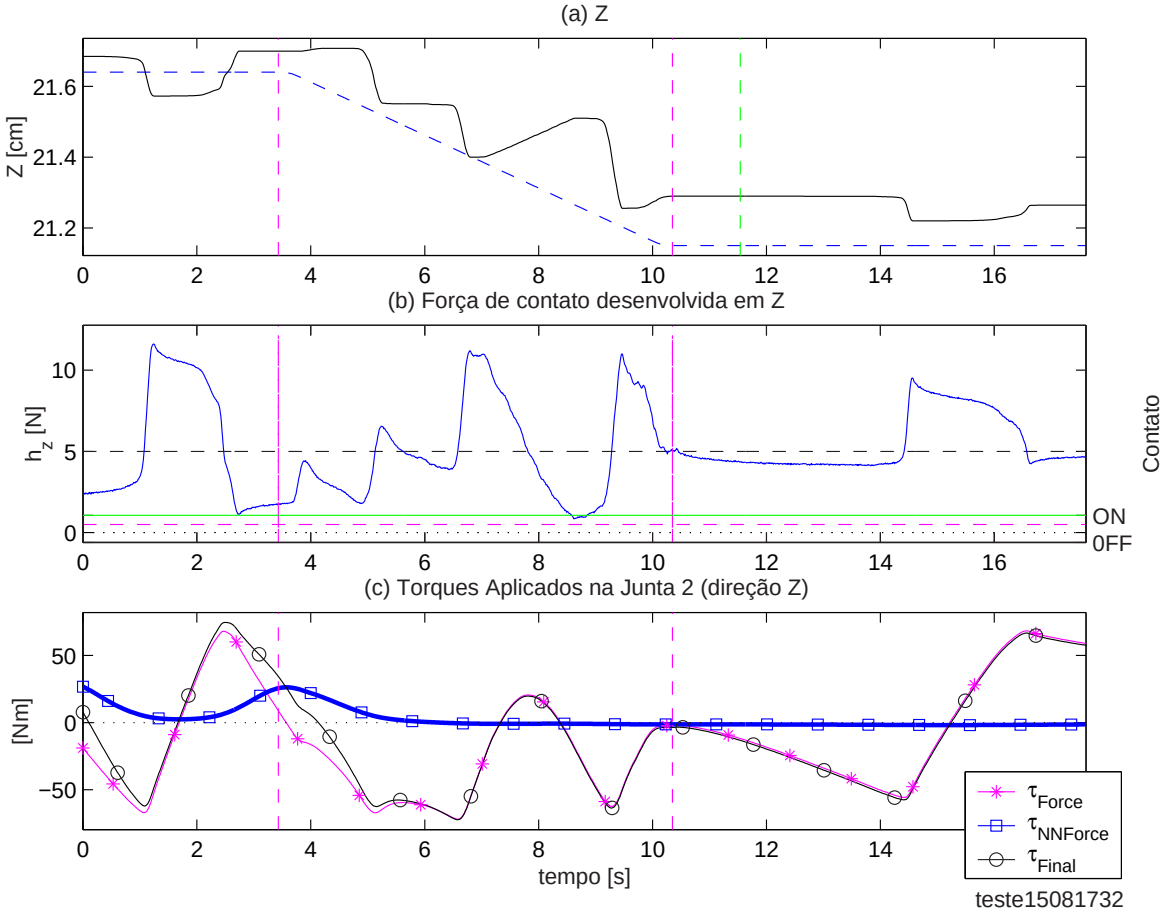


Figura 6.25: Teste do controlador de força integrado RNf-1(MLP) ($\eta = 0.035$ e $\alpha = 0.5$).

Observando-se a figura 6.25(c) percebe-se que a rede não contribuiu de forma a melhorar o desempenho da malha de controle de força. Foi gerado um *overshoot* pouco maior que o verificado no teste com o PI trabalhando isoladamente ($Máx\{h_z\} = 11.61$ [N], $\bar{h} = 4.92 \pm 2.92$ [N], entre $t_a = 4.43$ [s] e $t_b = 10.35$ [s], para força de contato especificada como: $h_{d_z} = 5.0$ [N]). Além disto, a figura 6.25(b) denuncia um comportamento algo oscilatório (e fora das expectativas) para a malha de controle de força. Em comparação com o ensaio anterior, o PI trabalhando isolado apresentou melhor desempenho ($Máx\{h_d\} = 10.87$ [N], $\bar{h}_d = 5.15 \pm 2.53$ [N]). Notar que a rede apenas contribuiu com torque não nulo nos instantes iniciais de treinamento até o instante de tempo igual aproximadamente igual à 6 segundos, quando o robô ainda não havia concluído o percurso sobre o meio – ver figura 6.26. O mais interessante é que esta rede percebe que o controle de força está sendo executado para a junta 2, mas

mesmo assim "estabiliza" suas saídas em torno de um valor nulo passados alguns ciclos de treinamento, figura 6.27.

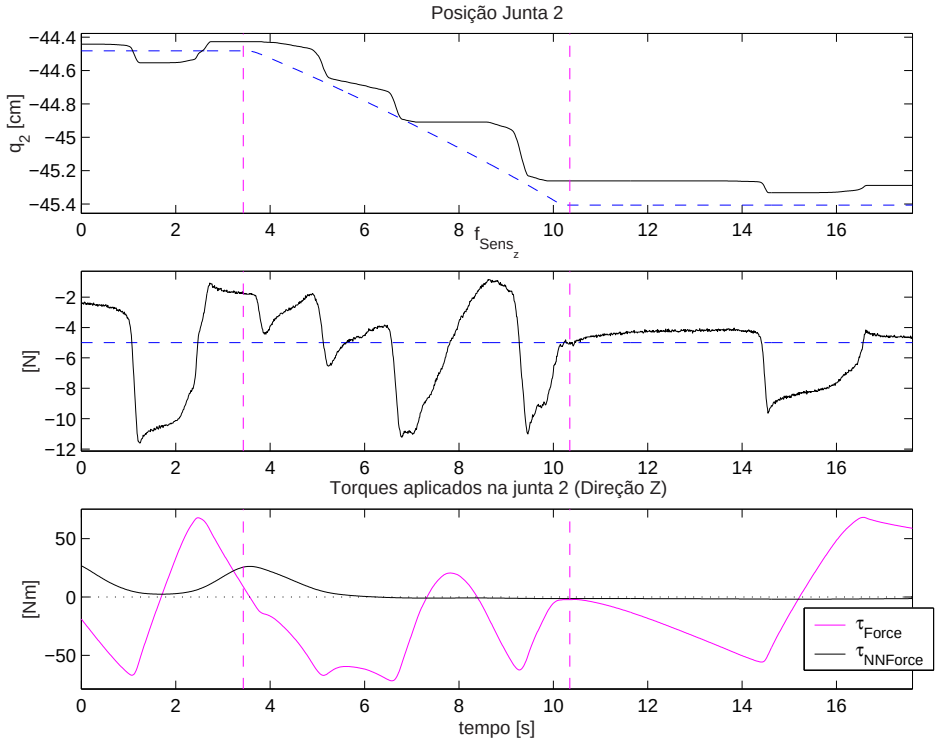


Figura 6.26: Fluxo de dados passando pela rede RNf-1(MLP).

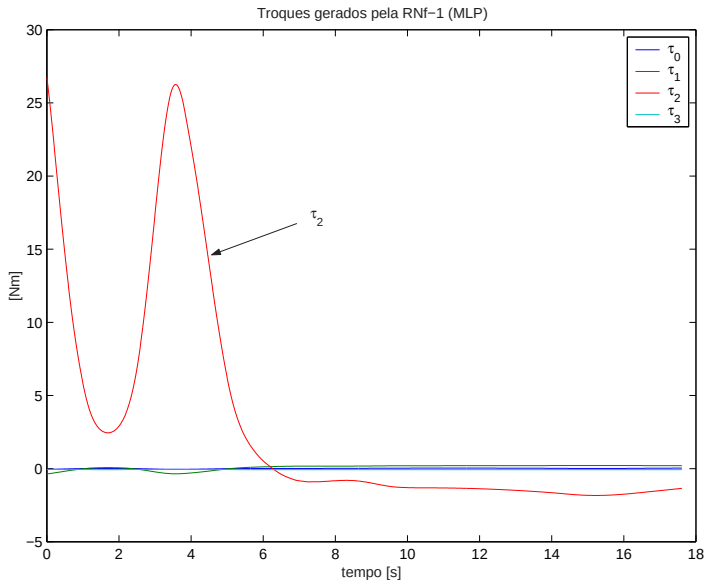


Figura 6.27: Torques gerados pela rede RNf-1(MLP).

Outro teste foi realizado com **taxa de aprendizado mais elevada**, $\eta = 0.05$. Os resultados obtidos aparecem na figura 6.28. O aumento da taxa de aprendizado implicou em oscilações na direção Z; os torques gerados pela rede para a junta 2 aumentaram ($Máx\{\tau_{NNForce_2} = 76.91\}$ [Nm]), o que trouxe uma certa instabilidade à malha de controle de força. Mas novamente, o desempenho da rede continua insatisfatório. Além do que, o efetuador final do robô perdeu contato com o meio mais de uma vez como pode ser percebido pela figura 6.28(b) e ainda antes de ter sido iniciado o movimento sobre a superfície ($t < 2$ [s]), justamente nos instantes iniciais em que a rede neural foi colocada em operação juntamente com o controlador convencional PI. A rede repetiu o mesmo comportamento que no teste anterior, baixando seus torques de saída para zero de maneira mais rápida que no ensaio anterior (os pesos desta rede foram re-inicializados para este outro ensaio).

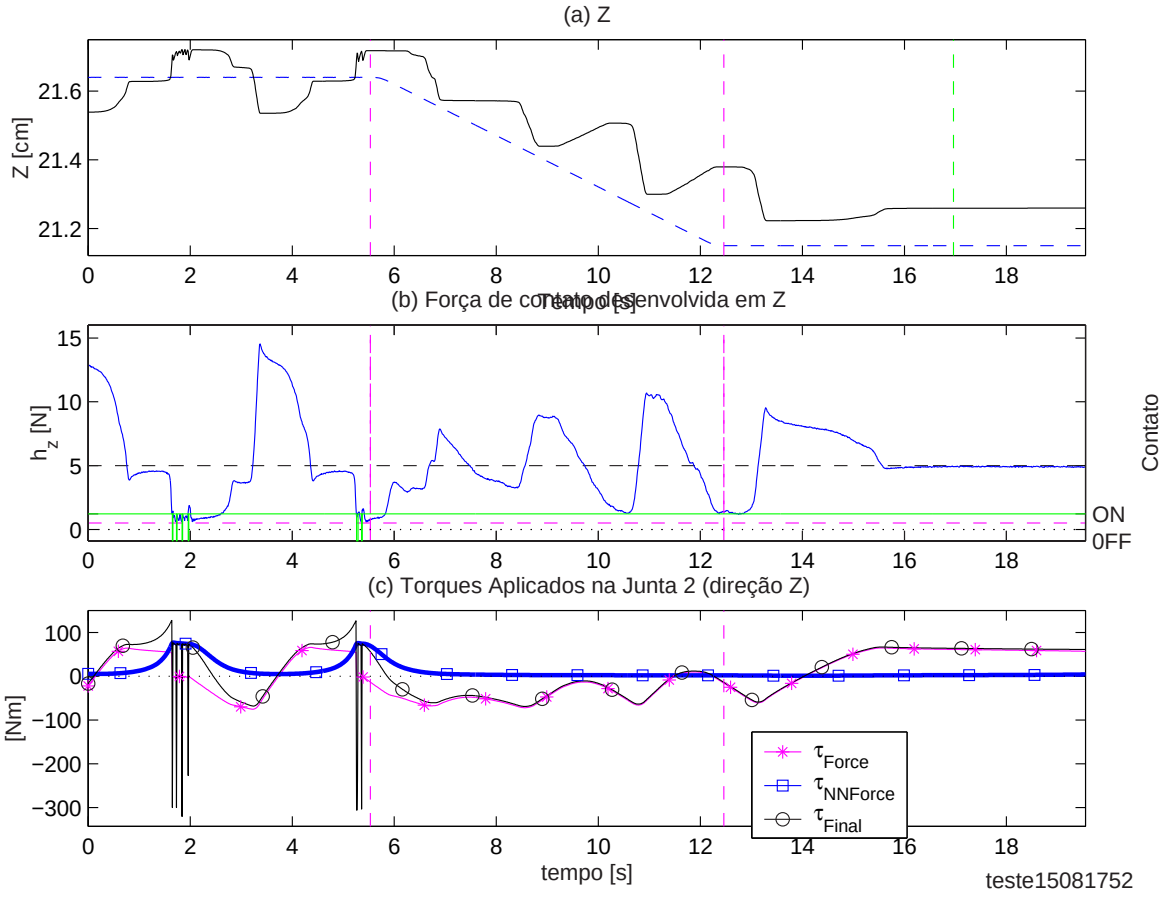


Figura 6.28: Teste do controlador de força integrado RNf-1(MLP) ($\eta = \mathbf{0.05}$ e $\alpha = 0.5$).

Conclusões preliminares

a) Estas redes trabalharam com os seguintes dados de entrada: $x = [q \quad f_{Sens} \quad h_d]$ e para o vetor de erro usado para seu treinamento foram usados os torques gerados pelo controlador convencional de força: $\epsilon = [\tau_{Force}]$. Estes não parecem ser dados adequados para operação da rede para a malha para controle de força.

b) outro problema pode ter sido com o fator escala relacionado com informações vindas do sensor de força. Notar que a faixa de valores sendo utilizada variava entre: $-0.1 < f_{Sens_Z} < 15.0[N]$; para esta faixa são gerados valores de entrada para rede variando entre: $-0.0004545 < PE_{in} < -0.0682$ – o que pode justificar a baixa "sensibilidade" desta rede. A tabela 6.7 mostra os fatores escala originalmente empregados e que trabalham com os valores limites impostos pelo sensor de força JR3.

Forças			Momentos		
Forças	*Máx*	*Fator escala*	Momentos	*Máx*	*Fator Escala*
f_x	$\pm$100 [N]	0.009091	μ_x	$\pm$6.3 [Nm]	0.1443
f_y	$\pm$100 [N]	0.009091	μ_y	$\pm$6.3 [Nm]	0.1443
f_z	$\pm$200 [N]	0.004545	μ_z	$\pm$6.3 [Nm]	0.1443

Tabela 6.7: Fatores escala originalmente empregados para as redes neurais de controle de força.

Teste com rede RBF, RNf-1 (RBF)

Foi adotada a mesma trajetória do teste anterior: mesmo meio de contato (borracha), força de contato desejada: $h_{d_z} = 5$ [N], limiar de contato estabelecido em: $F_{THR} = 0.5$ [N], e velocidade máxima de deslocamento sobre o meio de 5 [cm/s], controlador de força convencional PI com ganhos: $K_p = 200$ e $K_i = 20$. Foi testada uma rede RBF contendo 5 funções gaussianas por elemento de entrada na sua camada intermediária. Sendo que foram usadas 3 classes de dados de entrada: $x = [q \quad f_{Sens} \quad h_d]$, cada qual contendo: 4, 6 e 4 neurônios na sua entrada, totalizando uma camada intermediária com 70 neurônios. Foram utilizados: $\eta = 0.005$ e $\alpha = 0.5$. Mesmo antes de iniciar a execução da trajetória, esta rede introduziu mais vibrações na malha de controle de força que a rede MLP. E estas vibrações continuaram durante toda a execução da trajetória. Energeticamente a rede terminou executando 14034 ciclos de aprendizado e com um valor de MSE abaixo de 10^{-8} (ou seja, baixo índice de aprendizado). A figura 6.29 mostra os resultados obtidos no teste com esta rede.

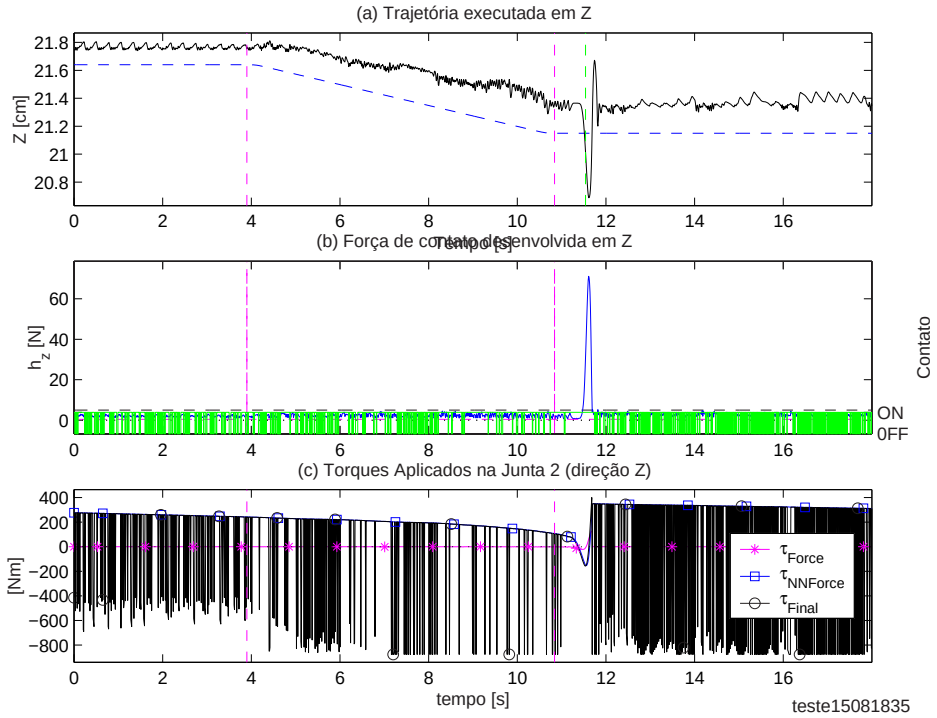


Figura 6.29: Resultados teste do controlador integrado RNf-1 (RBF), com $\eta = 0.005$ e $\alpha = 0.5$.

Uma breve inspeção na figura 6.29(b) permite concluir que este controlador não conseguiu manter a referência desejada para a força de contato, gerando um valor médio de contato de: $\bar{h} = 2.7913 \pm 4.8334$ [N]. Foi atingido um limite para a força de contato no valor de 71.1087 [N] muito acima do valor especificado para a força de contato: $h_{d_z} = 5.0$ [N]. Pior: este controlador integrado perdeu o contato diversas vezes com o meio resultado nos torques elevados e bastante oscilatórios gerados pelo chaveamento entre controle de força e controle de posição provocados pela estratégia para detecção das forças de contato em Z (o controlador convencional de posição, PD de fábrica, entra em ação toda vez que $f_{Sens} < 0.5[N]$). O controlador de posição em Z chegou a atuar em quase 80% do tempo. Certamente um comportamento bastante aquém do esperado para a rede RBF operando em paralelo com o controlador PI de força.

A figura 6.30 mostra o fluxo de dados envolvidos com esta rede neural – se percebe que esta rede aplicou torques excessivos sobre a junta 2 (o que provocou as fortes oscilações de contato, incluindo as freqüentes perdas de contato). E a figura 6.31 permite perceber melhor os torques envolvidos no comando da junta 2.

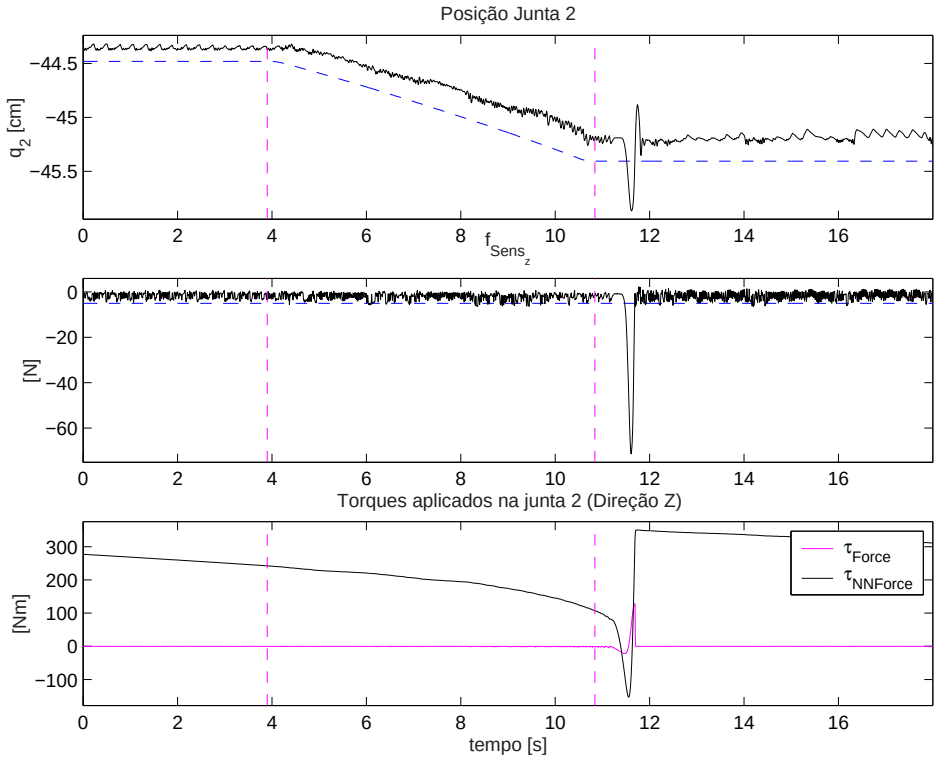


Figura 6.30: Fluxo de dados sobre a rede RNf-1 (RBF).

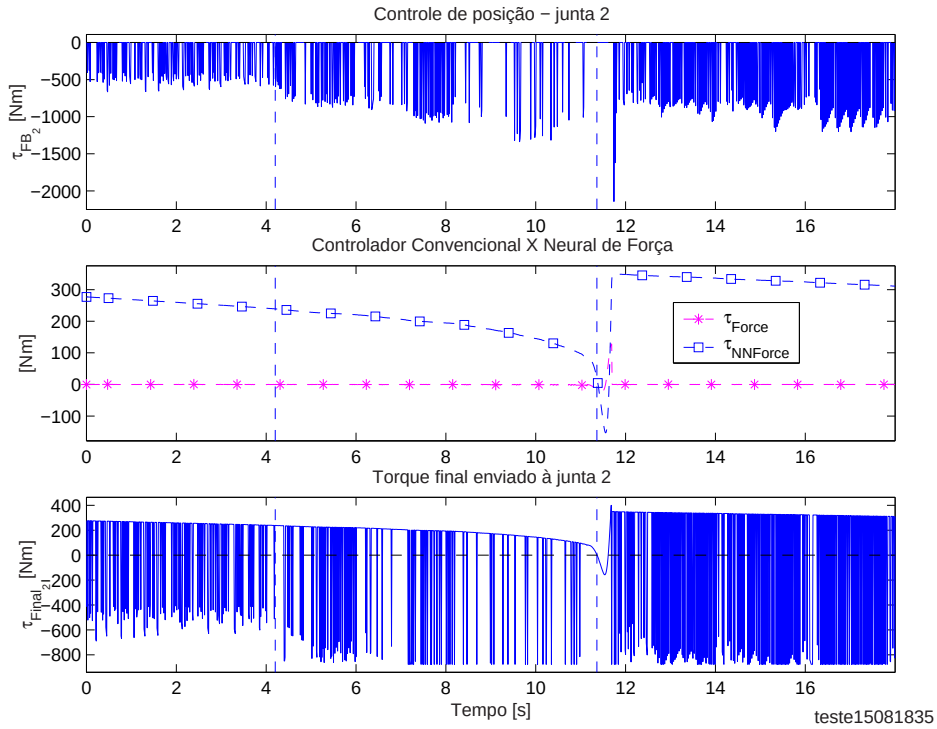


Figura 6.31: Torques gerados no teste do controlador RNf-1 (RBF).

Conclusões preliminares

a) Assim como no caso do controlador RNf-1 com a rede MLP, este controlador usando rede RBF apresentou um desempenho ainda inferior.

b) Mais uma vez a suspeita sobre a possível justificativa para este problema recaiu para a forma como as grandezas de valores dos dados relacionados com as informações vindas do sensor de força são organizadas para a rede RBF (para a rede MLP parece ser um problema com o fator escala empregado para estas informações). O fato desta rede apresentar um desempenho muito inferior aos controladores PI e PI+MLP se deve à forma como são organizadas suas funções gaussianas referentes aos dados que envolvem grandezas de força: classe de entrada referentes à f_{Sens} e h. A tabela 6.8 à seguir mostra como ficam organizadas as funcões gaussianas para estas classes de entrada nesta primeira versão dos dados de entrada para a rede RBF.

Classe 2: f_{Sens} (informações brutas do sensor de força)							
	Centro x_i da $i-$ésima função gaussiana						
Entrada	x_1	x_2	x_3	x_4	x_5	d_{max}	σ
f_x	-100.00	-50.00	0.00	50.00	100.00	50.000	15.811
f_y	-100.00	-50.00	0.00	50.00	100.00	50.000	15.811
f_z	-200.00	-100.00	0.00	100.00	200.00	100.000	31.623
μ_x	-6.30	-3.15	0.00	3.15	6.30	3.150	0.996
μ_y	-6.30	-3.15	0.00	3.15	6.30	3.150	0.996
μ_z	-6.30	-3.15	0.00	3.15	6.30	3.150	0.996
Classe 3: h_d (forças desejadas)							
	Centro x_i da $i-$ésima função gaussiana						
Entrada	x_1	x_2	x_3	x_4	x_5	d_{max}	σ
h_{d_X}	-100.00	-50.00	0.00	50.00	100.00	50.000	15.811
h_{d_Y}	-100.00	-50.00	0.00	50.00	100.00	50.000	15.811
h_{d_Z}	-200.00	-100.00	0.00	100.00	200.00	100.000	31.623
μ_{d_Z}	-6.30	-3.15	0.00	3.15	6.30	3.150	0.996

Tabela 6.8: Distribuição das funções gaussianas relativas às informações de força para a primeira rede RBF testada.

c) Foi percebido que para a faixa de valores usada para realizar a força de contato desejada as únicas funções gaussianas ativadas são as centrais, para $\varphi\big(f_z \cong 5\big)_{x_3} = 0.999$ (responsável por mapear força de contato nula!) – ou seja, do ponto de vista desta rede praticamente não existe contato! Seria necessário um algoritmo de aprendizado que adaptasse continuamente os valores dos centros (x_i) e das aberturas (d_{max}) das funções gaussianas utilizadas, mas isto traria mais complexidade às redes RBF implementadas pois exige

acréscimos ao algoritmo de aprendizado utilizado (atualmente, este apenas atualizar os pesos da sua camada de saída). Seria necessário prever também mais outros parâmetros de aprendizado (taxa de aprendizado e termo *momentum*) para definir a velocidade da adaptação dos valores dos centros e largura das funções gaussianas utilizadas. Ou pode-se optar por uma solução mais simples de forma a verificar se esta é realmente a causa do problema da baixa "sensibilidade" desta rede para a grandeza de valores trabalhados com a força de contato: **reescalonar os limites iniciais e finais com que trabalham as funções gaussianas relacionadas com as grandezas de força**. Este teste é realizado à seguir. Note que este mesmo procedimento de redefinir os valores máximos e mínimos para as entradas relacionadas com força em redes RBF também pode ser aplicado à redes MLP. Os testes realizados a seguir tentam confirmar esta hipótese.

6.4.2 Reescalonando informação de entrada para redes RNf-1

A fim de verificar se o baixo desempenho apresentado pelas redes para controle de força se deve aos valores fixos máximos adotados para escalonar ou mapear as informações relacionadas com força, novos valores limites foram adotados para representar informações de força conforme demonstra a tabela 6.9 que também mostra o impacto gerado sobre o fator escala para informações de força usados para as redes MLP – note uma redução para 25% dos valores máximos suportados pelo sensor de força JR3.

Forças			Momentos		
Forças	*Máx*	*Fator escala*	Momentos	*Máx*	*Fator Escala*
f_x	± 25 [N]	0.036364	μ_x	± 6.3 [Nm]	0.5772
f_y	± 25 [N]	0.036364	μ_y	± 6.3 [Nm]	0.5772
f_z	± 50 [N]	0.018182	μ_z	± 6.3 [Nm]	0.5772

Tabela 6.9: Novos fatores escala empregados para as redes neurais de controle de força.

Estes novos valores limites implicaram em redefinições para as funções gaussianas usadas pela rede RBF conforme pode ser observado na tabela 6.10.

Desta forma, tanto a rede MLP quanto a rede RBF foram preparadas para lidar com grandezas de força variando entre $-25.0 \leq h \leq 25.0$ [N], uma faixa de valores mais "compatível" com a região de operação do robô. Esperava-se desta maneira, aumentar a sensibilidade das redes para as variações de contato percebidas pelo controlador integrado de força. A seguir, seguem os novos testes.

Classe 1: q (posições angulares atuais das juntas do robô)							
	Centro x_i da $i-$ésima função gaussiana						
Entrada	x_1	x_2	x_3	x_4	x_5	d_{max}	σ
q_0	-2.25	-1.12	0.00	1.12	2.25	1.125	0.356
q_1	-1.90	-0.95	0.00	0.95	1.90	0.950	0.300
q_2	-0.49	-0.42	-0.35	-0.28	-0.21	0.070	0.022
q_3	-2.40	-1.20	0.00	1.20	2.40	1.200	0.379
Classe 2: f_{Sens} (informações brutas do sensor de força)							
	Centro x_i da $i-$ésima função gaussiana						
Entrada	x_1	x_2	x_3	x_4	x_5	d_{max}	σ
f_x	-25.00	-12.50	0.00	12.50	25.00	12.500	3.953
f_y	-25.00	-12.50	0.00	12.50	25.00	12.500	3.953
f_z	-50.00	-25.00	0.00	25.00	50.00	25.000	7.906
μ_x	-1.58	-0.79	0.00	0.79	1.58	0.788	0.249
μ_y	-1.58	-0.79	0.00	0.79	1.58	0.788	0.249
μ_z	-1.58	-0.79	0.00	0.79	1.58	0.788	0.249
Classe 3: h_d (forças desejadas)							
	Centro x_i da $i-$ésima função gaussiana						
Entrada	x_1	x_2	x_3	x_4	x_5	d_{max}	σ
h_{d_X}	-25.00	-12.50	0.00	12.50	25.00	12.500	3.953
h_{d_Y}	-25.00	-12.50	0.00	12.50	25.00	12.500	3.953
h_{d_Z}	-50.00	-25.00	0.00	25.00	50.00	25.000	7.906
μ_{d_Z}	-1.58	-0.79	0.00	0.79	1.58	0.788	0.249

Tabela 6.10: Redistribuição das funções gaussianas relativas às informações de força para as redes RBF.

Novo teste com rede MLP, RNf-1 (MLP)^{New}

Neste novo teste com a rede MLP, depois de reescalonados os valores limites para as forças de contato percebidas pela rede, foi utilizado como parâmetros para treinamento da mesma: $\eta = 0.035$ e $\alpha = 0.5$; e como ganhos para o controlador convencional PI, os valores: $K_p = 500$ e $K_i = 20$, com a força especificada para contato de $h_d = 5.0[N]$ e força que estabelece o limiar de contato: $f_{THR} = 0.5[N]$. O resultado obtido é mostrado na figura 6.32.

Percebe-se pela figura 6.32 que um re-escalonamento nos dados de entrada relativa à forças captadas pelo sensor não melhorou o desempenho da rede. O valor médio da força de contato desenvolvida durante a trajetória foi de: $\bar{h} = 4.9450 \pm 3.45$ [N] (no intervalo entre: $t_a = 2.39[s]$ e $t_b = 9.32[s]$, tendo sido percorrido 33.12 [cm] em $\Delta t = 6.93[s]$), tendo sido atingido um valor máximo para o contato de: $Máx\{h_z\} = 14.5359$ [N] e um valor mínimo de contato igual à: $Min\{h_z\} = 0.5739$ [N] (um valor muito próximo do mínimo estabelecido para manter o controlador de força atuando).

Além disto foi constatado que o ganho proporcional adotado para o controlador PI de força ($K_p = 500$) é excessivamente elevado uma vez que tornou oscilatório a resposta em

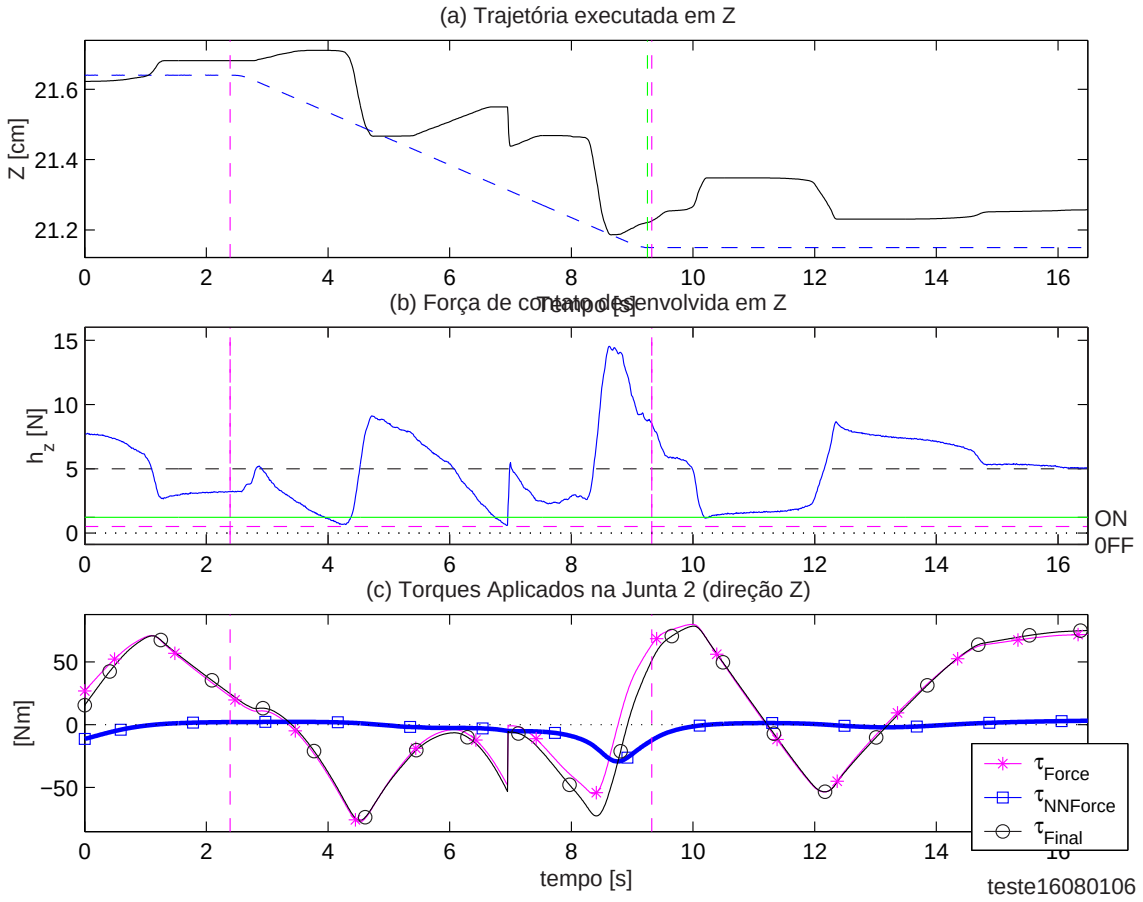


Figura 6.32: Teste do controlador RNf-1 (MLP), com entradas re-escalonadas.

regime permanente para este sistema (robô parado, instantes após atingir o ponto final da trajetória (b), $x = [0.4020 \quad -0.2463 \quad 0.2126 \quad -0.0705]$ ou $q = [-0.8892 \quad 0.6790 \quad -0.4529 \quad 0.1397]$), conforme demonstra a figura 6.33, acarretando inclusive freqüentes perdas de contato. Foram captadas variações de contato pelo sensor de força variando entre: -1.2 (perda de contato) $\leq f_{Sens_z} \leq 10.5$ [N]. Em aproximadamente 5% do tempo o contato foi perdido e enviado uma ação de controle elevada como resultado da entrada do controlador de posição que passa a atuar nos casos de perda de contato). Naturalmente um comportamento bastante indesejável para um controlador.

Conclusões preliminares:

Outros testes foram realizados diminuindo-se os ganhos do controlador de força convencional, diminuindo-se os ganhos do controlador de posição (na direção Z) e variando-se

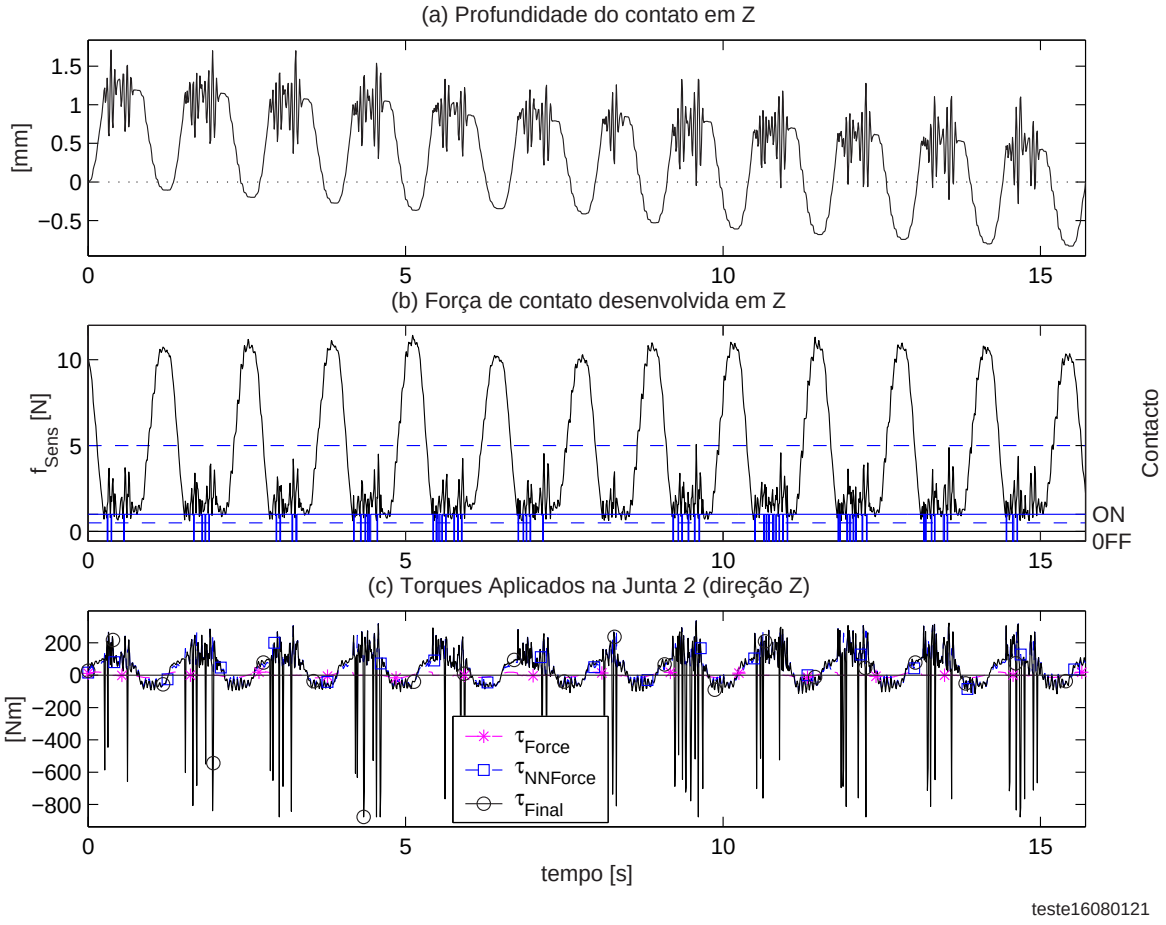


Figura 6.33: Resultado do ganho proporcional elevado para controlador PI operando com a RNf-1 (MLP).

a taxa de aprendizado da rede, entre: $0.05 \leq \eta \leq 0.01$, mas mesmo assim, os resultados obtidos foram similares aos demonstrados anteriormente, ou seja, insatisfatórios (mesmo reinicializando-se para cada teste os pesos da rede).

Novo teste com rede RBF, RNf-1 (RBF)^{New}

Neste novo teste com a rede RBF, depois de re-escalonados os valores limites para as forças de contato percebidas pela rede, foi utilizado com parâmetros para treinamento da mesma: $\eta = 0.005$ e $\alpha = 0.5$ (mesmos parâmetros adotados para a rede da malha de controle de posição); e como ganhos para o controlador convencional PI, os valores: $K_p = 500$ e $K_i = 20$, com a força especificada para contato de $h_d = 5.0[N]$ e força que estabelece o limiar de contato: $f_{THR} = 0.5[N]$.

Desta vez, logo após a ativação da rede RBF na malha de controle de força, ocorreu uma parada de emergência com o robô: o robô atingiu a chave de fim de curso da junta 2, além de ter sido gerado o torque máximo suportado pelo atuador da junta 2 (877 [Nm]) e ter sido detectado "impacto" na direção Z (força de contato superior à: $f_{Sens_z} > 80$ [N]). A rede RBF já havia executado 69 ($\times 2$ [ms] = 0.138 [s]) ciclos de treinamento. No instante de amostragem anterior ao travamento do robô foram detectados os estados mostrados na tabela 6.11 (não houve tempo hábil para disparar a tarefa de captura de dados em tempo-real, os dados mostrados nesta tabela são cópias dos conteúdos de certas variáveis do robô).

Entradas:						
Pos. Junta	Junta 0	Junta 1	Junta 2	Junta 3		
q	-0.4740	1.3558	-0.4512	-0.9411		
	[rad]	[rad]	[m]	[rad]		
Pos. Esp. Op.	X	Y	Z	θ		
x	0.3814	0.0788	0.2101	-0.0594		
	[m]	[m]	[m]	[rad]		
Forças e Momentos	X	Y	Z	μ_x	μ_y	μ_z
f_{Sens}	0.3491	0.1282	**-71.4380**	0.0630	-0.0074	-0.0080
h	-0.4631	0.2769	**76.5560**			
$\tilde{h}$			**71.5560**			
	[N]	[N]	[N]	[Nm]	[Nm]	[Nm]
Forças Desejadas	h_{d_x}	h_{d_y}	h_{d_z}	μ_{d_z}		
h_d	ZERO	ZERO	5.00	ZERO		
	[N]	[N]	[N]	[Nm]		
Saídas:						
Torques gerados	Junta 0	Junta 1	Junta 2	Junta 3		
τ_{FB}	8.69	-3.38	ZERO	-0.80		
τ_{Force}	ZERO	ZERO	-28.21	ZERO		
NN_{out}	0.0397	-0.0041	-03624	-0.0526		
$\tau_{NNForce}$	13.23	-0.64	**-317.85**	-0.88		
τ_{Final}	8.69	3.38	**-346.06**	-0.80		
	[Nm]	[Nm]	[Nm]	[Nm]		

Tabela 6.11: Estados atingidos pela rede RNf-1 (RBF) com entradas re-escalondas.

Conclusões preliminares:

Esta rede saturou muito mais rápido que a anterior que trabalhava com os limites máximos e mínimos de força e momentos relacionados com a faixa de operação do sensor de força. Conclui-se que **redefinir os centros e larguras das funções gaussinas** para uma região mais próxima do ponto de operação adotado com o robô para a tarefa em questão, **não é** a causa do mau funcionamento deste tipo de rede.

6.4.3 Conclusões gerais com relação às redes RNf-1

A seguir são traçados as conclusões gerais extraídas à partir dos testes com as redes: RNf-1 (MLP), RNf-1 (RBF), RNf-1 (MLP)^{New} e RNf-1 (RBF)^{New}:

a) Foi comprovando que o problema com o controlador integrado (RNf-1), não está relacionado com o fator escala dos dados de entrada do sensor de força.

b) Um **novo "pacote de entrada de dados"** deve ser especificado para a rede neural da malha de controle de força. Apenas os dados de entrada: $x_{NNForce} = [\ q \quad f_{Sens} \quad h_d\ ]$ não se mostraram adequados e suficientes.

c) Ou ainda, uma freqüência de amostragem superior à adotava deve ser utilizada de forma a passar dados suficientes para que a rede consiga inferir a dinâmica (extremamente rápida) deste sistema.

6.4.4 Segunda Versão da RN para Controle de Força (RNf-2)

Foi testada a rede MLP, com taxa de aprendizado: $\eta = 0.001$ (diminuiu), e termo *momentum*: $\alpha = 0.5$. Os ganhos utilizados no controlador PI da malha de força foram: $K_p = 200$ e $K_i = 40$ (aumentou). O teste foi executado novamente sobre a superfície de borracha numa trajetória semelhante à adotada nos itens anteriores (figura 6.24, pág. 241), tendo sido especificado uma velocidade máxima de deslocamento de 5 [cm/s], pressão de contato desejada: $h_d = 8.0[N]$, limiar de força para manter controlador de força ativado: $f_{THR} = 0.5[N]$. O resultado pode ser visto na figura 6.34 (note que foi especificada um movimento de ida e volta: do ponto (a) para (b) e de volta para o ponto (a)).

Esta rede já interagiu um pouco melhor como o controlador convencional de força, conforme mostram somente as curvas de torque mostradas na figura 6.35.

Percebemos pela figura 6.34(b) que em nenhum instante foi perdido o contato com o meio apesar das oscilações iniciais de contato verificadas – o trecho demarcado entre os instantes de tempo: $4.97 \leq t \leq 11.9[s]$ reflete a primeira parte do movimento realizado sobre a peça, do ponto (a) para o ponto (b); a parte posterior retrata o resultado obtido com um movimento de retorno do ponto (b) para o ponto (a). Percebe-se que no movimento do retorno, a saída do sistema se manteve mais próxima da referência desejada. A tabela 6.12 mostra o desempenho alcançado por este controlador integrado de força. É nítida uma

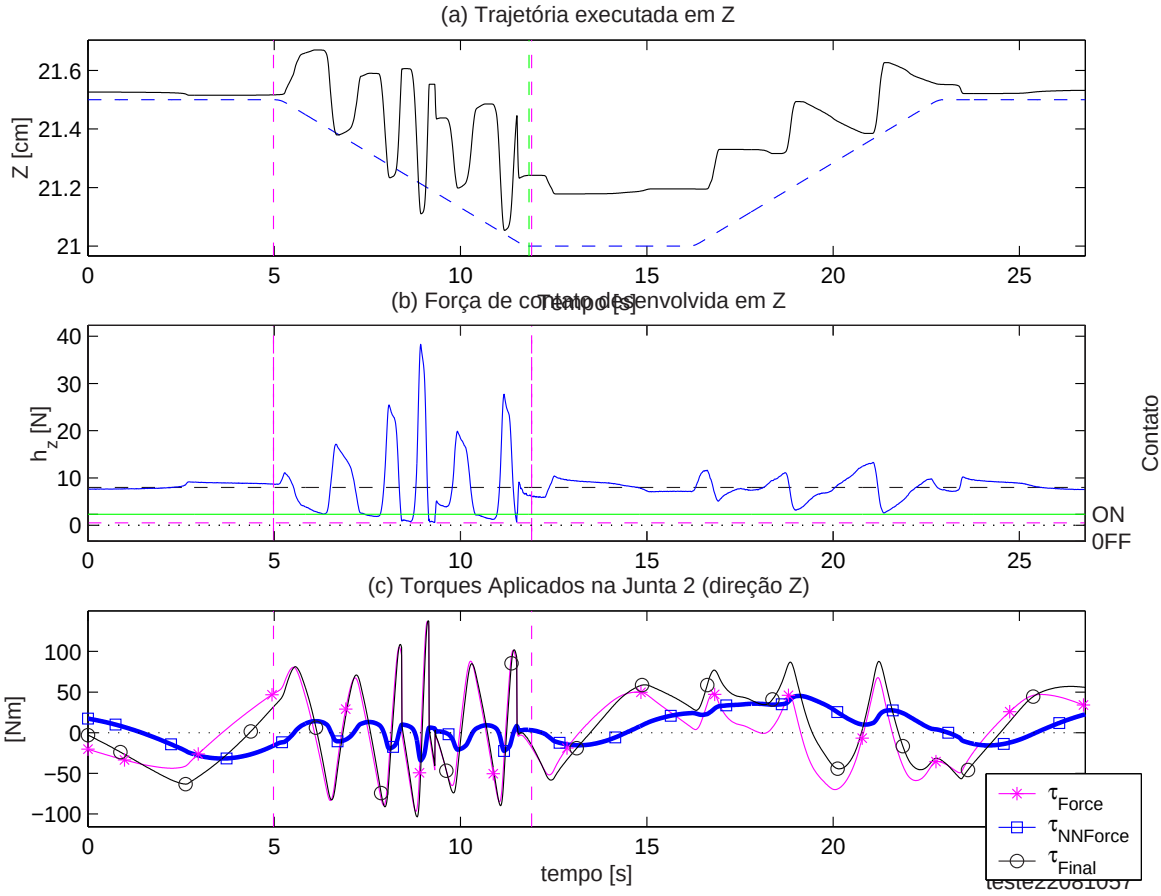


Figura 6.34: Teste do controlador RNf-2.

melhora no desempenho em comparação com a RNf-1, principalmente no movimento de retorno (do ponto (b) para o ponto (a)).

Conclusões preliminares à respeito da rede RNf-2

a) É nítida uma melhora no desempenho em comparação com a RNf-1; a rede interagiu melhor com o controlador convencional de força PI, principalmente no movimento de retorno (do ponto (b) para o ponto (a)), mesmo com repetições no teste.

b) Mesmo assim, ainda não foi alcançado o desempenho esperado pela operação da rede em paralelo com o controlador convencional de força.

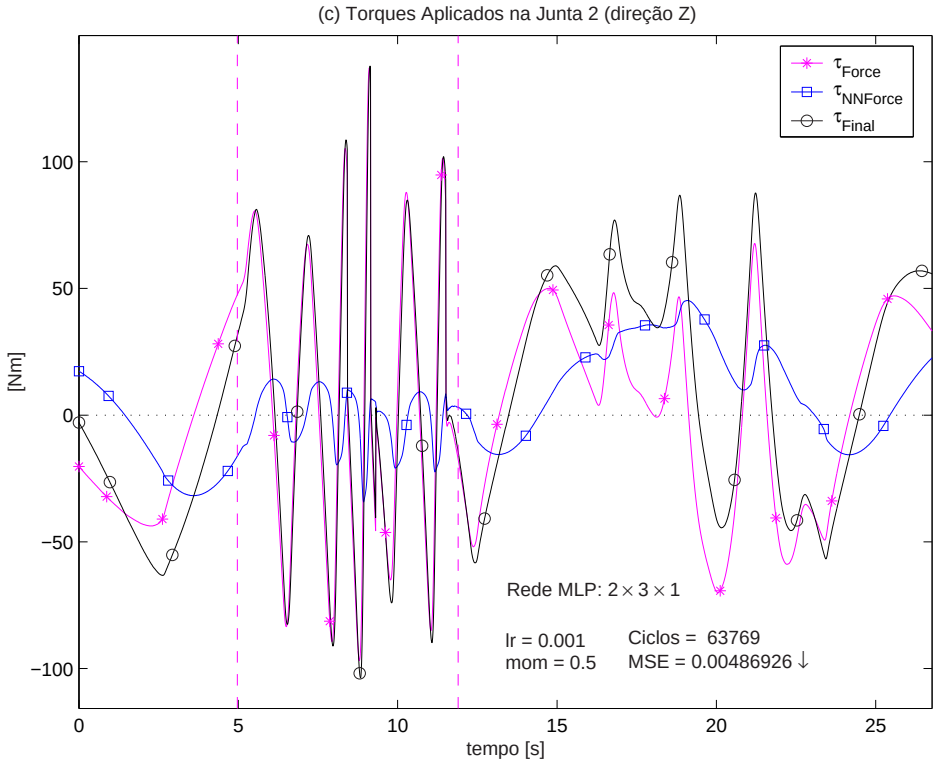


Figura 6.35: Torques gerados pelo controlador RNf-2.

Dados da Trajetória executada:

	X	Y	Z	θ
x_i	0.3822	0.0814	0.2152	-0.1001
x_f	0.4031	-0.2454	0.2124	-0.1090
	[m]	[m]	[m]	[rad]
$\dot{x}$	0.0052	0.0549	-0.0647	0.2629
	[m/s]	[m/s]	[m/s]	[rad/s]

Índices de Desempenho:

	Máx	Mín	Méd	VAR	IAE	MSE
h_z	38.326	0.570	8.697	63.84	4.28k	64.24
	[N]	[N]	[N]	[N^2]	[N]	[N^2]
τ_2	137.72	-103.68	4.66	3.22k		
	[Nm]	[Nm]	[Nm]	[Nm]		
k[ini]= 498, k[fim]=1191, $\Rightarrow$ t_{ini}= 4.97[s], t_{fim}=11.90[s]						

(a) Trajetória de (a) para (b).

Dados da Trajetória executada:

	X	Y	Z	θ
x_i	0.4031	-0.2454	0.2120	-0.1070
x_f	0.3822	0.0814	0.2155	-0.0930
	[m]	[m]	[m]	[rad]
$\dot{x}$	0.0052	0.0549	-0.0647	0.2629
	[m/s]	[m/s]	[m/s]	[rad/s]

Índices de Desempenho:

	Máx	Mín	Méd	VAR	IAE	MSE
h_z	13.299	2.580	7.822	6.34	1.39k	6.36
	[N]	[N]	[N]	[N^2]	[N]	[N^2]
τ_2	87.69	-45.55	18.07	1.77k		
	[Nm]	[Nm]	[Nm]	[Nm]		
k[ini]=1608, k[fim]=2302, $\Rightarrow$ t_{ini}=16.07[s], t_{fim}=23.01[s]						

(b) Trajetória de (b) para (a).

Tabela 6.12: Resultados obtidos com a trajetória de ida (RNf-2).

6.4.5 Terceira Versão da RN para Controle de Força (RNf-3)

Nesta nova abordagem para a rede neural de controle de força foram passadas amostras atrasadas do sinal de força capturado pelo sensor de força para dentro da rede. A trajetória utilizada para teste foi semelhante à adotada nos testes anteriores. Foi especificada uma força de contato um pouco superior: $h_{d_z} = 10$ [N] $\cong 1$ [Kg], limiar da força de contato: $f_{THR} = 1.0$ [N]. Foi utilizado o controlador PI com ganhos: $K_p = 200$ e $K_i = 40$. A rede operou com $7 \times 4 \times 1$ neurônios por camada, totalizando uma rede com 3 camadas; taxa de aprendizado em $\eta = 0.035$ (taxas de aprendizado inferiores praticamente "paralisam" a rede — esta gera torque praticamente nulo de saída) e termo *momentum* em $\alpha = 0.5$. O resultado obtido é mostrado na figura 6.36 (trajetória executada do ponto (b) para o ponto (a), figura 6.24).

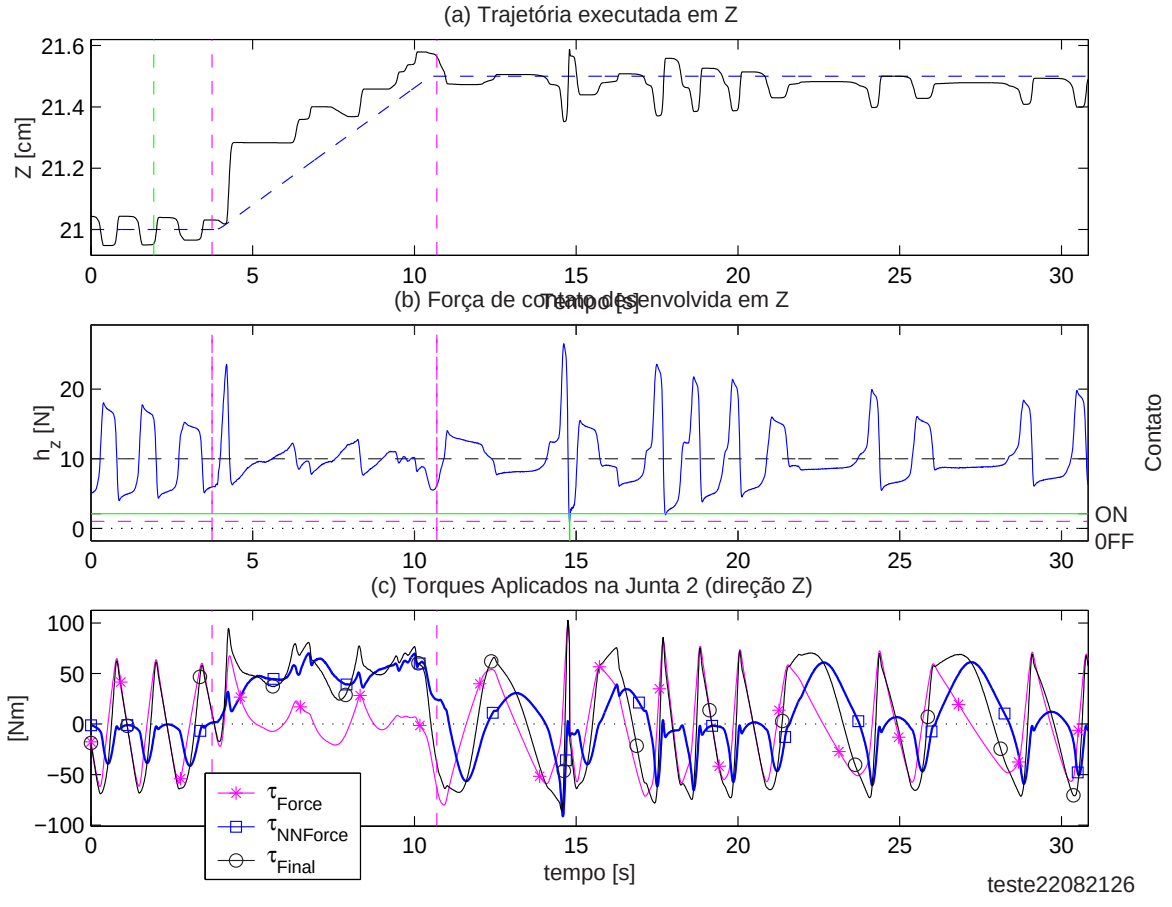


Figura 6.36: Teste do controlador RNf-3.

Nota-se um comportamento bastante oscilatório para as ações de controle geradas pelo

controlador convencional e pela rede neural em regime estático (robô parado sobre o ponto (b)). Durante a execução da trajetória a rede parece compensar bem o efeito dinâmico envolvido no contato, gerando a ação de controle preponderante (figura 6.36, intervalo de tempo entre t=3.74 [s] e t=10.69 [s]), mas depois de terminada a execução da trajetória, com o robô tendo atingido a sua posição final, a rede parece entrar num estado oscilatório. A tabela 6.13 compara o desempenho obtido pelo controlador convencional PI trabalhando sem a rede e com a rede RNf-3 durante o período de movimentação sobre o meio.

Dados da Trajetória executada:

	X	Y	Z	θ
x_i	0.4031	-0.2455	0.2103	-0.1004
x_f	0.3821	0.0814	0.2156	-0.0940
	[m]	[m]	[m]	[rad]
$\dot{x}$	-0.0048	0.0560	0.0368	0.1127
	[m/s]	[m/s]	[m/s]	[rad/s]

Índices de Desempenho:

Controlador de Força	Parâmetros	Máx	Mín	Méd	VAR	IAE	MSE
PI somente	h_z	24.326	1.908	10.109	20.80	2.59k	20.78
PI + RNf-3	h_z	23.603	5.490	9.731	6.04	1.11k	6.10
		[N]	[N]	[N]	[N^2]	[N]	[N^2]
PI somente	τ_2	86.57	-62.25	1.73	1.93k		
PI + RNf-3	τ_2	94.82	-38.77	48.09	547.53		
		[Nm]	[Nm]	[Nm]	[N]		

Tabela 6.13: Indices de desempenho para trajetória de (b) para (a).

O desempenho obtido com o controlador de força PI + RNf-3 foi superior ao PI somente (tabela 6.13), entretanto, o controlador PI + RNf-3 não oferece desempenho satisfatório em regime permanente (nem baixando-se a taxa de aprendizado, que induz paralisia na rede, torque nulo de saída). Interessante perceber que a "energia" da rede (figura 6.37) baixa de forma mais acentuada durante o regime dinâmico da execução da trajetória e que depois, o erro da rede continua caindo, mas não mais com a mesma intensidade.

Foram realizados outros testes com taxa de aprendizado maiores que: $\eta = 0.01$, mas então foi verificado que a rede passou a gerar torques maiores durante a fase estática do robô em contato com o meio e torques quase nulos durante a fase em que o robô se movimentava sobre o meio. Os torques maiores durante o período estático além de introduzir oscilações na malha de controle de força (o PI de força tentando contrabalançar o torque errôneo gerado pela rede) chegaram nos piores casos, a ativar a rotina de segurança do robô

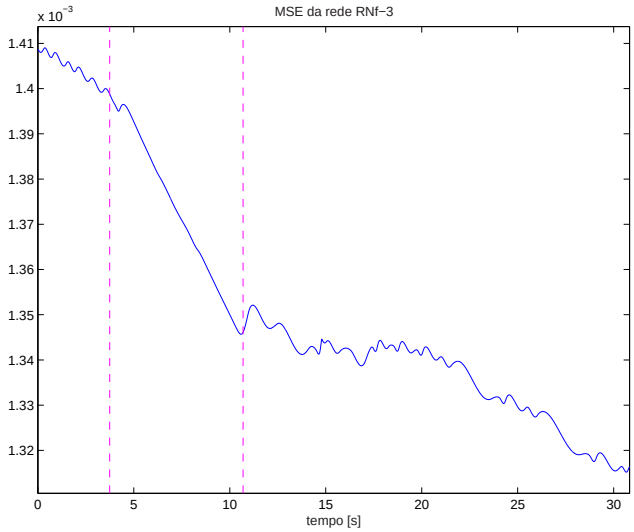


Figura 6.37: Estado energético da rede RNf-3.

($f_{Sens_z} > 80$ [N]) e para estes piores casos pôde ser percebida que não havia ganho suficientemente elevando no PI da malha de força para fazê-lo reagir a tempo de evitar a paralisação do robô. De qualquer forma torques elevados no controlador PI operando isoladamente levam à instabilidade na malha de controle de força.

6.4.6 Conclusões preliminares à respeito da rede RNf-3

a) O controlador integrado RNf-3 apresentou um desempenho superior às outras redes durante o movimento do robô sobre o meio com o qual estava realizando contato. Durante o movimento a rede foi responsável por gerar a ação de controle predominante. Mas com o robô estático sobre o meio foi comprovada uma tendência à oscilação no controlador RNf-3.

b) O fato do controlador integrado RNf-3 não ter se comportado bem durante o período estático do robô em contato com o meio sugere que a inclusão de dados relacionados com velocidade do deslocamento no plano XY, pode evitar com que este controlador ingresse num estado oscilatório. Ou seja, mais uma vez, é necessária uma revisão no pacote de dados de entrada gerado para a rede neural de controle de força.

6.4.7 Quarta Versão da RN para Controle de Força (RNf-4)

Esta rede leva em conta informação atual captada pelo sensor de força, erro entre força desejada e força atual e velocidade de deslocamento da junta 2, desta forma, estaria-se passando informação dinâmica sobre o contato realizado pelo robô contra o meio. Para teste deste controlador foi adotada o mesmo perfil utilizado nos testes anteriores: força de contato: $h_{d_z} = 10$ [N] $\cong 1$ [Kg], limiar da força de contato: $f_{THR} = 1.0$ [N], velocidade máxima de execução da trajetória em 5 [cm/s], com PI de força trabalhando com os ganhos: $K_p = 200$ e $K_i = 40$. A figura 6.38 mostra o resultado com esta rede operando com taxa de aprendizado em $\eta = 0.02$ e $\alpha = 0.5$.

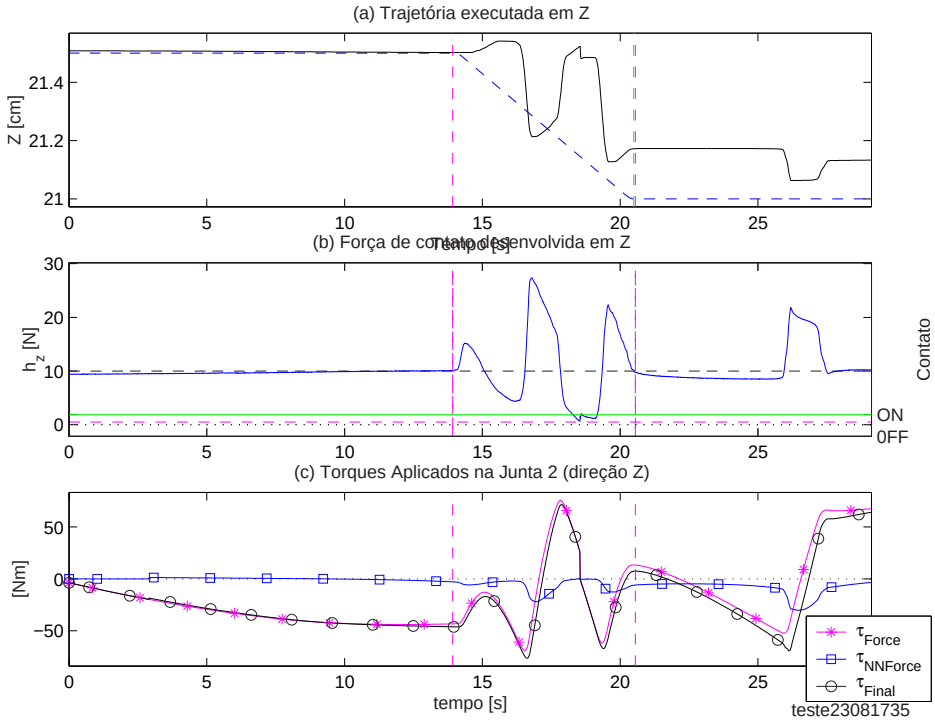


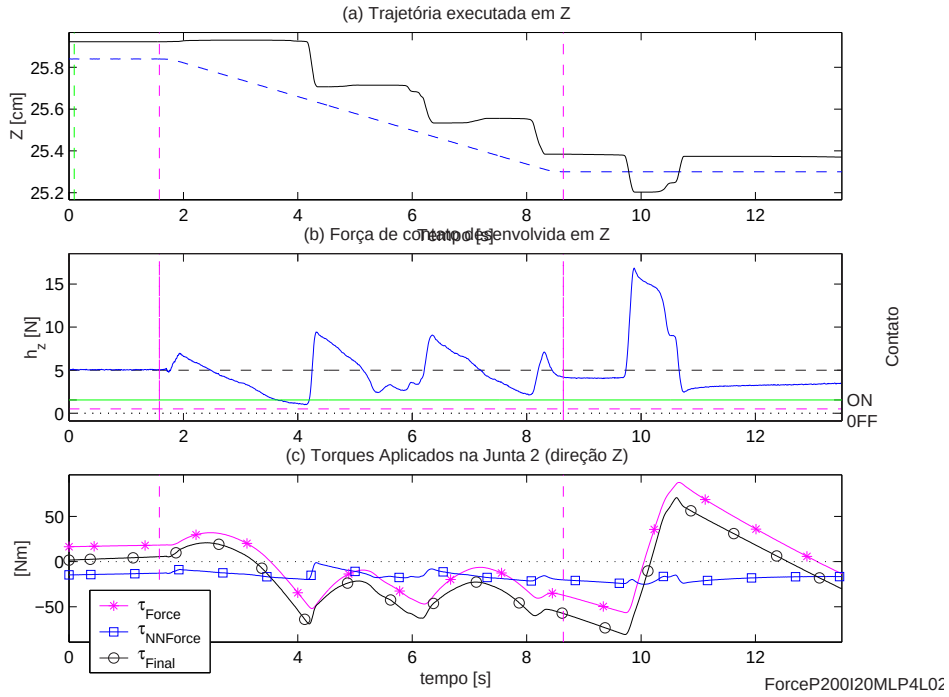
Figura 6.38: Resultado do teste com RNf-4, com $lr = 0.02$ e $mom = 0.5$.

Taxas de aprendizado inferiores paralisam a rede (torque nulo de saída) enquanto que taxas de aprendizado superiores fazem com que a rede gere torques abruptamente elevados durante o regime estático do controle de força (chegando a ativar a função de segurança do robô, $f_{Sens_z} > 80$ [N], robô "afunda" sobre o meio) e durante o período de deslocamento do robô sobre o meio o torque gerado pela rede praticamente se mantém nulo.

Foi testado o controlador PI operando com ganho integrativo menor: $K_i = 20$ e a rede continuou gerando torque praticamente nulo durante o movimento do robô sobre o meio,

mas desta vez, sem gerar torques errôneos na fase estática do robô em contato com o meio. Também este ganho integrativo baixo não foi suficiente para garantir contato do robô com o meio durante todo o período de execução da trajetória – foi perdido contato com a borracha no mínimo uma das vezes durante os testes realizados.

Experimentou-se também repetir a execução desta trajetória diversas vezes a fim de verificar alguma alteração no comportamento da rede (figura 6.39). Mas nenhuma mudança substancial foi detectada.



Ganhos do controlador PI: $K_p = 200$, $K_i = 20$ e rede com $\eta = 0.02$ e $\alpha = 0.5$

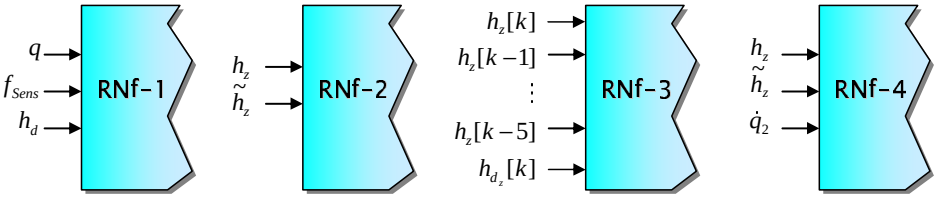
Figura 6.39: Teste com trajetória repetida para controlador RNf-4.

Também foi realizado teste desta rede operando apenas com um controlador Proporcional na malha de de controle de força, mas os resultados observados se mostraram tão insatisfatórios quanto os anteriormente verificados.

6.4.8 Conclusões preliminares à respeito da rede NNf-4

a) Esta rede (RNf-4) apresentou um comportamento similar ao da rede RNf-1, ou seja, torques inexpressivos durante a fase de deslocamento do robô sobre o meio sob contato.

b) Pior: este controlador apresentou um caracter instável durante o período estático do robô em contato com o meio (regime permanente).



onde:

q: vetor ($\Re^4$) de posição angular instantânea das juntas do robô;

f_{Sens}: vetor ($\Re^6$) relativo às informações brutas geradas pelo sensor de força. Envolve dados como: $f_{Sens} = [\ f_x \quad f_y \quad f_z \quad \mu_x \quad \mu_y \quad \mu_z\ ]$, onde: f_i se relaciona com informações de forças detectadas nas direções XYZ e μ_i se relaciona com os momentos detectados em torno dos eixos XYZ;

h_d: vetor ($\Re^4$) relativo às forças de contato desejadas nas direções XYZ e momento desejado (limitado) em torno do eixo Z. Estas informações são passadas com referência ao espaço operacional (coordenadas de base do robô);

h_z: escalar relacionado com a força atual (no espaço operacional) detectada na direção do eixo Z;

h_{d_z}: especifica a força de contato desejada na direção do eixo Z;

$\tilde{h}_z$: erro envolvido na força de contato: $\tilde{h} = h_z - h_{d_z}$;

$h_z[k]$: informação do instante de amostragem atual relacionada com a força (já traduzida para o espaço operacional) detectada na direção do eixo Z;

$h_z[k-i]$: se refere à $i-$ésima amostra atrasada da informação de força na direção do eixo Z;

$\dot{q}_2$: velocidade angular instantânea (atual) à respeito da junta 2 do robô (terceira junta, que trabalha em paralelo com o eixo Z).

Figura 6.40: Pacotes de dados de entrada usados para as redes de controle de força.

c) A inclusão da informação de velocidade com a qual o robô faz contato com o meio também não se mostrou relevante para a rede de controle de força. Novamente, outro pacote de entrada de dados ainda deve ser explorado.

6.5 Discussão à respeito dos Controladores Neurais de Força

Foram testados 4 tipos diferentes de entradas de dados para a rede da malha de controle de força. A figura 6.40 resume os pacotes de dados de entrada testados.

Cabe ressaltar que nenhum destas variações nas entradas de dados para a rede de controle de força produziu resultados satisfatórios. Em resumo:

a) o **Fator de escala** dos dados relacionados com o sensor de força não é o problema detectado para a operação das redes. Inicialmente trabalhou-se como os mesmos limites de operação do sensor de força e depois estes limites foram reduzido para 25% deste valor sem correspondente melhora no desempenho das redes. Aliás, a rede RBF saturou mais rapidamente.

b) Observou-se um comportamento dinâmico bem acelerado para o sistema, correspondendo aos comentários realizados por (Volpe, 1990) relativo a elevadas freqüências de amostragem exigidos para controladores explícitos de força.

c) **RNf-1**: a rede pouco interage com o controlador convencional da malha de controle de força. Começa a produzir torques de saída (baixos entretanto) e depois anula seu próprio torque de saída. Não prejudica o controlador mas também não agrega nenhuma vantagem ao sistema.

d) **RNf-2**: esta rede interage um pouco melhor com o controlador convencional da malha de controle de força. Entretanto, o desempenho global deste controlador integrado é inferior a de um simples controlador PI de força.

e) **RNf-3**: esta rede interagiu mais fortemente com o controlador convencional de força, principalmente durante o período em que o robô se deslocou sobre o meio com o qual estava sendo mantido o contato. Durante esta fase de movimentação, a rede atrelada à malha de controle de força permitiu alcançar resultados superiores ao de um simples controlador PI de força. Neste período a rede gerou torques de controle predominantes na malha de controle de força, repetindo um comportamento semelhante ao desenvolvido pelo controlador integrado de posição. Entretanto, esta rede entra num estado oscilatório no momento em que o robô permanece em situação estática de contato contra o meio, o que desabona a aplicação deste tipo de rede (pacote de entrada de dados) para a malha de controle de força. Não fosse este indesejável comportamento e esta combinação de entrada de dados seria a ideal para trabalhar na malha de controle de força.

f) **RNF-4**: outra rede que interage pouco com o controlador convencional de força. Esta rede gera torques irrelevantes para a malha de controle de força, desta forma não agregando nenhuma característica adicional desejável ao controlador convencional presente na malha de controle de força.

g) os resultados obtidos parecem refletir um caso de **baixa freqüência de amostragem** eventualmente adotado na malha de controle de força. Conforme já citado por (Volpe, 1990), malhas de controle de força explícitas exigem taxas de amostragem elevadas devido as dinâmicas de alta freqüência presentes no sistema robô-meio. Eventualmente se a taxa

de amostragem pudesse ser baixada, novos testes com os controladores neurais confirmariam se estas redes seriam capazes de inferir o comportamento dinâmico rápido deste sistema. Entretanto, a CPU atual do robô Inter/Scara não permite baixar o período de amostragem para menos de 2 [ms] se o usuário desejar usar redes neurais para controle de força. Faltam recursos computacionais neste caso. Note que se fosse possível baixar o período de amostragem de 2 [ms] para a 1 [ms], teríamos um acréscimo de 100% na freqüência de amostragem adotada para os controladores integrados de força. Mas devido as limitações atuais da CPU do robô, seria necessário elevar a sua freqüência de *clock* ou paralelizar as operações relacionadas com o processamento de redes neurais numa outra CPU que não a do robô, o que envolve detalhes operacionais de sincronia de operações e troca de dados que foge ao escopo deste trabalho.

6.6 Discussão Final sobre Controle de Força

Pelos resultados obtidos com os controladores convencionais e os controladores integrados de força que foram propostos neste trabalho, nota-se que:

a) Controladores convencionais simples de força conseguem trabalham com períodos de amostragem variando entre 1 [ms] e 2 [ms]. Entretanto, conforme aumenta o período de amostragem, diminui a margem de ganho destes controladores (o ganho máximo que pode ser praticado diminui).

b) Os controladores integrados de força propostos neste trabalho não conseguiram alcançar resultados desejáveis para a malha de controle de força. A rede que alcançou melhores resultados foi a que operava com amostras atrasadas de informações coletadas diretamente do sensor de força, porém, em regime permanente a rede atrelada a este controlador integrado entra em oscilação o que acaba descartando seu uso.

c) Sendo assim, parece ser necessário passar para a rede neural de controle de força, informação que permita a mesma inferir o estado atual de movimentação do robô sobre o meio e não apenas contra o meio. Ou seja, não basta fornecer apenas amostras atrasadas de informações geradas pelo sensor de força e até a velocidade com a qual o robô possa estar se deslocando para dentro (ou fora) do meio, mas também é necessário se passar dados de velocidades nas direções em que o movimento não é restringido. Isto

de certa forma, faria com que a rede para controle de força trabalhasse em parte com os mesmos dados usados pela rede da malha de controle de posição, o que implica numa certa ”redundância”: duas redes objetivando compensação dinâmica na sua malha específica estariam trabalhando com dados de entrada similares. Isto aponta para uma solução que adota uma única rede capaz de operar simultaneamente na malha de controle de posição e na malha de controle de força. Esta rede poderia receber os dados que a rede original para controle de posição já usa + informações relativas à malha de controle de força como: força atual captada pelo sensor de força + amostras atrasadas desta mesma informação + dados sobre que forças de contato devem ser realizadas e em que direção. Isto traria ainda como vantagem, menor custo computacional para a CPU do robô uma vez que apenas uma única rede seria utilizada tanto para gerar torques para a malha de controle de força quanto para a malha de controle de posição.

Capítulo 7

Conclusões e Perspectivas

Este capítulo está dividido em 6 itens. O primeiro traz um resumo sobre o que trata este trabalho. O segundo item enumera as contribuições advindas deste trabalho. O terceiro item traz as conclusões alcançadas com respeito aos controladores de posição integrados avaliados neste trabalho. O quarto separa as conclusões referentes aos controladores de força testados neste trabalho. O quinto item relaciona as limitações e pontos não abordados neste trabalho e por fim, este capitulo termina com as sugestões para trabalhos futuros.

7.1 Pontos Abordados

Este trabalho teve um caráter prático, explorando uma técnica da área de inteligência computacional, mais especificamente o uso de redes neurais artificiais aplicadas para tratar de um problema complexo da área de robôs manipuladores como o do controle de posição/força. Os controladores sendo propostos neste trabalho foram implementados e testados na prática sobre um robô manipulador real do tipo Scara (com 4 graus de liberdade), equipado com sensor de força que está disponível no Laboratório de Automação Industrial (LAI) da Universidade Federal de Santa Catarina (UFSC). Basicamente foi explorada a arquitetura de aprendizagem por retropropagação de erros (*feedback error learning*), na qual uma rede neural trabalha em "paralelo" com um controlador convencional e a informação de erro a ser utilizada para o treinamento das redes parte do próprio sinal de controle gerado pelo controlador convencional. A vantagem no uso desta arquitetura é que a mesma permite aproveitar o controlador convencional previamente existente no sistema sem que seja

necessário modificar seu algoritmo. Neste trabalho, resolveu-se referenciar esta arquitetura de controle mesclando um controlador convencional com uma rede neural como "controlador integrado". Dois tipos diferentes de redes neurais foram exploradas: redes *perceptrom* multicamadas (MLP) e rede de função de base radial (RBF). Para controle de força/posição do robô manipulador foi selecionada a abordagem de controle híbrido de posição/força. A idéia por trás desta técnica é realizar controle de força nas direções nas quais o movimento é restringido e realizar controle de posição nas direções ortogonais restantes (direções nas quais o movimento não é restringido). Este conceito intuitivo permite testar e avaliar com facilidade controladores de posição/força separados para cada uma das malhas: malha de controle de posição e malha de controle de força. Este trabalho objetivou testar diferentes controladores integrados para cada uma destas malhas, já que esta técnica de controle híbrido não limita o tipo de controlador que o projetista deseja utilizar para cada uma das malhas de controle.

Desta forma, foram inicialmente testados controladores integrados de posição e após controladores integrados de força. Resultados bastante promissores foram alcançados com os controladores integrados de posição implementados, conforme o esperado, refletindo o sucesso já alcançado por trabalho similar já realizado na área por Battistela (1999). Apenas que neste caso foi aplicada ainda uma rede neural do tipo RBF que permitiu alcançar resultados superiores. Adicionalmente até uma vantagem extra: baixísssimo nível de ruído em movimentações do robô quando comandado pelo controlador integrado baseado em redes RBF. Já para o controlador integrado de força foram encontrados novos desafios. O uso de redes neurais trabalhando em paralelo com os controladores convencionais adiciona um forte custo computacional ao sistema original de controle de robô que antes operava apenas com um controlador convencional simples: PD de velocidade no espaço de juntas (com período de amostragem de 1 [ms]). Tal sobrecarga de processamento exigiu que o período de amostragem adotado para a rotina do controlador fosse elevado de 1 [ms] para 2 [ms] sob pena de não sobrarem recursos computacionais para fazer operar 2 controladores integrados simultaneamente (um para a malha de controle de posição e outro para a malha de controle de força) ou mesmo operar um controlador de posição integrado usando uma rede RBF com 9 funções gaussianas na sua camada intermediária. Além disto, como a primeira versão testada para o controlador integrado de força não alcançou resultados esperados, diferentes

"pacotes de entrada" de dados foram avaliados para as redes neurais usadas no controlador integrado sem que resultados aceitáveis houvessem sido alcançados. Isto reflete em parte a dificuldade encontrada na área de controle de força para a modelagem do sistema formado pela interação robô-meio. Outros trabalhos práticos realizados nesta área, como o de Volpe (1990), sugerem uma mudança na forma de abordar o problema: é necessário se estabelecer uma malha interna de realimentação baseada em informação de posição ou velocidade do robô contra o meio, além da malha de realimentação baseada em informações geradas pelo sensor de força. Tal abordagem leva ao projeto de controladores conhecidos como por "impedância mecânica", que levam em conta o sistema gerado pela interação robô-meio, cujas dinâmicas são muito mais aceleradas que um simples sistema robô manipulador de controle de posição. Ao final deste capítulo são sugeridas novas estruturas para o controlador integrado de força: uma que emprega apenas uma rede neural para atuar tanto na malha de controle de força quanto na malha de controle de posição e que trabalha com dados de entrada usados tanto pelos controladores convencionais de posição quanto de força; e outra estrutura, baseada no conceito de controle por impedância mecânica e controle por modelo de referência (MRAC).

7.2 Principais Contribuições

Ressalta-se nesta seção, as principais contribuições obtidas com este trabalho:

a) Foi realizada uma análise comparativa de desempenho entre controladores convencionais e baseados em redes neurais para o controle de posição e força em um robô manipulador real (não somente simulações).

b) Foi proposta, implementada e testada arquitetura de controladores integrados (uso de redes neurais) com elevado grau de eficiência para controle de posição e seguimento de trajetórias que exigem um mínimo esforço adicional de projeto (apenas um parâmetro extra de ajuste: taxa de aprendizado das redes neurais utilizadas).

c) Este trabalho inovou na forma de aplicar uma rede neural RBF na estrutura do controlador neural integrado de posição baseado na retropropagação de erros. A idéia de organizar os dados de entrada na forma de diferentes "*classes de entrada de dados*", inspirado na

abordagem similar adotada em sistemas baseados em lógica *fuzzy* se mostrou bem sucedida. Foi a rede que, apesar do maior custo computacional exigido, permite alcançar o melhor desempenho para o controle de posição com uma vantagem adicional: incorpora baixíssimo nível de ruído durante as movimentações do robô.

d) Este trabalho buscou definir índices de desempenho adequados para a avaliação dos controladores. Dentre eles cabem ressaltar os índices:

- Referente ao erros de seguimento de trajetória:
  1) R1: q_f ou x_f, que se refere aos erros finais de seguimento de trajetória;
  2) R2: $MÁX\{\tilde{q}\}$ ou $MÁX\{\tilde{x}\}$, referente aos maiores erros de seguimento de trajetória verificados;
  3) R5: $MSE\{\tilde{q}\}$ ou $MSE\{\tilde{x}\}$ que acentuam os erros de seguimento de trajetória (mais que o índice de integração de erros absolutos: R3 ($IAE(\cdot)$) ou erro médio de seguimento: R4 ($MSE(\cdot)$));
  4) R6: $VAR\{\tilde{q}\}$ ou $VAR\{\tilde{x}\}$ que medem a variância do erro de seguimento de trajetória e permitem estimar um comportamento de saída muito oscilatório ou não para o sistema.
- E referente ao esforço de controle realizado:
  1) R7: $MÁX\{\tau_{Final}\}$, que permite detectar os maiores picos de ação de controle realizados;
  2) R8: $MED\{\tau_{Final}\}$, que permitiria estimar o valor médio do esforço de controle realizado não fosse o fato de certas movimentações do robô (principalmente no espaço operacional) exigirem mudanças de sentido no torque aplicado à certa junta – nestes casos, este índice perde sua utilidade;
  3) R9: $VARi\{\tau_{Final}\}$, que permitiria estimar o controlador que executou a ação de controle mais suave. Não é calculado da mesma forma que o índice estatístico variância pelo motivo explicado no item anterior.

e) Foram avaliadas a eficácia de várias arquiteturas de controle integrado de força utilizando redes neurais, apontando suas deficiências e mostrando que certas combinações de dados de entrada freqüentemente utilizadas com outros controladores não funcionam bem na prática.

f) Novas propostas de controladores integrados de força são feitas neste trabalho baseado nas conclusões obtidas com ensaios também experimentais de controle de força realizados por Volpe (1990) e numa arquitetura de controlador baseado em modelo de referência operando sobre uma malha de controle de força/posição por impedância mecânica.

7.3 Conclusões sobre Controle de Posição Integrado

As seguintes conclusões podem ser extraídas dos resultados obtidos com os controladores integrador de posição:

a) A aplicabilidade real dos controladores integrados propostos. Foi verificado que uma rede neural trabalhando em paralelo com um controlador convencional de posição, usando como sinal de erro para ajuste dos pesos da rede, o sinal de saída gerado pelo controlador convencional (*feedback error learning*) é eficiente e possível de ser adotado na prática com acréscimos de desempenho consideráveis.

b) Controladores neurais trabalhando na arquitetura baseada na retropropagação de erros (*feedback error learning*), aqui referenciados como "controladores integrados" são uma opção interessante para controle de posição de robôs manipuladores. Este esquema de controle "híbrido" na qual o controlador convencional trabalha em paralelo com uma rede neural executando controle/compensação dinâmica pode ser usado, na qual o controlador convencional lida com os parâmetros conhecidos e disponíveis sobre o sistema e a rede neural lida com as incertezas e dinâmicas não modeladas deste sistema.

c) Neste trabalho o desempenho alcançado pelos diferentes controladores integrados trabalhando com diferentes tipos de rede neurais foi comparado com outros controladores: PD e PID além de um controlador robusto baseado na teoria de modos deslizantes (*sliding mode controller*). Foi detectado que um controlador por modos deslizantes pode até alcançar resultados melhores aos alcançados por um controlador do tipo PD mas desde que este esteja muito bem sintonizado e ajustar este controlador foi um dos problemas práticos verificados. Porém, notar que o controlador por modos deslizantes implementando neste trabalho não incluía ação integral (o que melhoraria seu desempenho) nem ajuste por majorações (o que traria robustez para certa faixa de variação de parâmetros).

d) Este trabalho implementou controladores por Jacobiana Inversa, PD e PID operando diretamente no espaço operacional e não no espaço de juntas. Entretanto estes controladores são de aplicação limitada por dependerem da determinação da matriz Jacobiana com posto completo o que deixa de ocorrer em pontos próximos do raio mais externo de operação do robô Scara no plano XY. Neste caso, a matriz Jacobiana acaba reduzindo seu posto porque o ângulo formado entre a primeira e segunda juntas do robô é nulo ou próximo de zero (condição de singularidade). Nestes casos, estes controladores não são capazes de evitar que o robô entre num estado de paralisia. Portanto, apesar destes controladores terem funcionado na prática, seu uso acaba sendo restringido.

e) A introdução de uma rede neural MLP ou RBF ao controlador convencional de posição já existente no controle de posição de um robô manipulador melhora consideravelmente o desempenho desta malha de controle. Entretanto certas detalhes devem ser levados em conta:

   i) Deve sobrar recursos computacionais no sistema onde se pretende implementar um controlador integrado. Nitidamente, os controladores integrados baseados em redes neurais RBF são os que mais exigem poder computacional (capacidade de executar mais operações em ponto flutuante durante o período de amostragem adotado para o controlador atual);

   ii) Do operador é exigido apenas um esforço extra de aplicação (um parâmetro a mais de sintonia): a taxa de aprendizado da rede neural utilizada. Já que os testes realizados comprovaram que o melhor valor para o termo *momentum* utilizado no algoritmo expandido de retro-propagação de erros (*backpropagation*) pode ser fixado no valor igual à 0.5.

f) A combinação de controlador convencional com rede neural utilizada no controlador integrado de posição que permite alcançar os melhores resultados é o PID devido a sua característica de **ação integral**. A ação integral acelera o papel da rede na determinação do torque preponderante de saída para ser enviado para as juntas do robô.

g) Redes neurais MLP com mais de uma camada invisível não agregam nenhum benefício extra ao sistema. Um pouco pior: deixa a resposta da rede um pouco mais lenta e com um leve efeito residual de memória no caso de operações repetitivas realizadas com o

robô. Uma rede neural MLP com apenas 1 camada invisível permitiu alcançar resultados superiores à rede com 2 camadas invisíveis.

h) A rede neural que permitiu alcançar os melhores resultados entre os controladores integrados testados foi a rede RBF. Esta rede permite inclusive movimentar o robô com baixíssimo nível de ruído (redução das vibrações à um nível mínimo). Entretanto, existe um custo elevado associado à este ganho de desempenho: uma rede neural RBF exige maior capacidade de processamento da CPU do robô. Tanto que executar uma rede neural RBF com 9 funções gaussianas nas suas camandas intermediárias implicou na exaustão dos recursos do sistema XO/2 sobre o qual roda a tarefa de tempo-real associada com o controle do robô. O sistema atual do robô. não consegue executar redes RBF contendo mais que 7 funções gaussinas na sua camada intermediária mantendo taxa de amostragem de 1 [ms]. Esta, no entanto, é uma limitação do processador utilizado no robô Inter/Scara e não da técnica em si.

i) Aumentar o número de funções gaussianas da rede RBF utilizada no controlador integrado não implica em igual relação: custo computacional $\times$ ganho de desempenho. O custo computacional aumenta muito mais que o ganho em desempenho obtido.

7.4 Conclusões sobre Controle de Força Integrado

A partir dos resultados obtidos na seção referente a parte experimental de controle de força usando redes neurais, podemos concluir que:

a) Apesar de não terem sido alcançados resultados promissores com a aplicação dos controladores neurais de força, este trabalho ressalta que certas combinações de dados de entrada para a rede neural não serão capazes de surtir o efeito desejado. Ou seja, as seguintes combinações de dados de entrada não surtem os efeitos desejados:

1) $x_{NN} = [\ q \quad f_{Sens} \quad h_d\ ]$;

2) $x_{NN} = [\ h_d \quad \tilde{h}_d\ ]$;

3) $x_{NN} = \left[\ h_z[k] \quad h_z[k-1] \quad \cdots \quad h_z[k-5] \quad h_{d_z}[k]\ \right]$;

4) $x_{NN} = [\ h_z \quad \tilde{h}_z \quad \dot{q}_2\ ]$;

Entre as combinações acima, a que pareceu a mais promissora foi a terceira ($x_{NN} = \begin{bmatrix} h_z[k] & h_z[k-1] & \dots & h_z[k-5] & h_{d_z}[k] \end{bmatrix}$) que realizou a ação preponderante de controle de força mas apenas durante a etapa de movimentação do robô sobre o meio com o qual estava realizando contato. Porém, assim que o robô atinge sua posição final, esta rede começa a entrar num estado oscilatório, não a ponto de desestabilizar o controlador integrado como um todo, mas estas oscilações não esperadas e nem bem vindas levam em certos casos à que o controlador integrado de força perca o contato com o meio com o robô parada (em regime permanente).

b) Foi percebido durante os ensaios práticos realizados com os controladores convencionais de força que a ação integral é altamente desejável na malha de controle de força como forma de suprimir os atritos presentes nas juntas do robô. Um controlador Proporcional de força não é capaz de garantir erros nulos em regime permanente.

c) Falta investigar ainda com que tipos de informações (dados de entrada) deve trabalhar um controlador de força. Conforme verificado, informação atrasada pode ser útil durante o período de execução de um movimento restringido no espaço, mas apenas esta informação é inútil para garantir estabilidade do controlador em regime permanente.

d) Adicionar informação de velocidade de deslocamento do robô na direção onde o movimento é restringido também não é suficiente;

e) Trabalhar baseado apenas em informação de erro de força de contato realizada não garante bons resultados (regulação de erro), nem evita elevados *overshoots*.

Dos ítens anteriores se pode concluir que falta informação para o controlador (seja os convencionais testados: P ou PI, ou o controlador integrado: P + RN-MLP ou PI + RN-MLP ou PI + RBF).

Mais que informação há detalhes práticos que devem ser levados em consideração. Mesmo um simples controlador proporcional de força pode se revelar inadequado para controle de força por duas razões:

1) Dificuldade em sintonizar os ganhos do Controlador (P ou PI). Os ganhos variam conforme a natureza (características) do meio sobre o qual está sendo realizado o contato. Não podemos usar os mesmos parâmetros de ajuste para executar pressão de contato sobre borracha ou madeira.

2) Um controlador mal sintonizado, mesmo que sua estabilidade tenha sido algebricamente provada, pode levar a ações excessivas de controle e como, neste caso, o robô está em contato com o meio, podemos deformar o meio sob contato ou mesmo comprometer o efetuador final do robô (num caso sem limites controlados por *software*, pode-se mesmo chegar a danificar o efetuador final ou partes do robô).

   Isto leva a outras considerações:

3) Mais do que buscar um controlador cuja estabilidade e robustez possam ser provadas, necessitamos também garantir limites máximos de operação que podem ser desenvolvidos sob pena de danificar o meio sob contato (causar deformações indesejáveis no meio) até quebra no efetuador final do robô (principalmente no caso de uma resposta drástica e excessiva do controlador).

4) Fica clara a necessidade de se modelar também o meio e projetar o controlador levando em conta o modelo do meio sob consideração. De qualquer forma, voltam as mesmas questões: quais seriam os parâmetros (variáveis de entrada) à serem utilizados para modelar o contato robô-meio de forma a garantir um controle mais eficaz.

5) Como não foi possível implementar controle de força com períodos de amostragem menores que 2 [ms], fica em aberto a questão da capacidade da rede neural de aprender comportamentos decorrentes de dinâmicas muito rápidas,uma vez que podem ser essenciais para a estabilidade e mesmo convergência do algoritmo de treinamento.

Estes seriam em resumo, os principais desafios relacionados com a área de controle de força em robôs manipuladores.

7.5 Limitações e Pontos não abordados

Nesta seção são feitos comentários sobre algumas limitações deste trabalho.

O aprendizado de redes neurais RBF pode incluir além do ajuste dos pesos da sua camada de saída, o ajuste da largura e centro de cada uma de suas funções gaussianas utilizadas pelos neurônios da camada intermediária. É esta particularidade de aprendizado que confere a característica de aprendizado local típico de redes RBF. Ocorre que o ajuste dos centros e larguras das funções gausssianas implica em mais 2 parâmetros de aprendizados e

por motivos de tornar o sistema o mais simplificado possível e exigindo a menor quantidade de parâmetros de ajuste possível, não foi implementada esta possibilidade de ajuste das suas funções gaussianas. As redes RBF utilizadas partem com os parâmetros de centros e larguras das funções gaussianas já definidas à *priori*, que permanecem fixos durante toda a utilização desta rede neural. De qualquer forma, mesmo que estes parâmetros pudessem ser ajustados via expansão no seu algoritmo de treinamento, eles teriam de ser inicializados de uma forma conveniente com a que foi tratada: as funções gaussianas foram definidas de forma à cobrir toda a faixa de excursão das suas variáveis de entrada, para evitar que a rede perdesse a capacidade de mapear alguma configuração específica do robô. Pelos resultados observados ao menos com os controladores de posição integrados, esta "perda" de flexibilidade nas redes RBF utilizadas não parece ter prejudicado o alto desempenho alcançado pelas mesmas. E mesmo para o caso da rede RBF aplicada no controlador integrado de força, variar os centros das funções e larguras das suas funções gaussianas para permitir operação mais próxima de um ponto "local", no caso, forças mais concentradas numa faixa entre ± 15 [N] não foi capaz de melhorar o desempenho destes controladores apesar de ter sido verificado que a rede reagiu muito mais rápido às variações presentes na sua entrada. Fica em aberto a questão: variar as larguras e centros das funções gaussianas da rede RBF, não poderia levá-la a uma característica de aprendizado claramente local, com perda de capacidade de generalização (queda no desempenho do controlador integrado) nos casos de mudança na área de operação do robô (mudança de planejamento da tarefa)? A idéia por trás de se fixar os centros e larguras da funções gaussianas se aproxima muito da forma pela qual se definem as funções de pertinência para um sistema baseado em lógica *fuzzy*. Normalmente estas funções são definidas à *priori* de forma a cobrir toda a faixa de excursão das suas variáveis de entrada e assim permanecem fixas durante todo o restante do processo de utilização do sistema *fuzzy*. Nos sistemas *fuzzy* se está mais preocupado em se definir um conjunto de regras eficazes envolvidas com o motor de inferência *fuzzy* adotado. Papel que numa rede RBF é desempenhado pela sua camada de saída. Os pesos presentes na sua camada de saída definem o equivalente ao grau de pertinência daquela regra (conexão entre a saída de uma função gaussiana e correspondente relação com o torque de saída que deveria ser produzido para uma das juntas do robô). O algoritmo de treinamento (*back-propagation* adotado neste trabalho) é o responsável por definir estas "regras". Redes neurais RBF são um pri-

meiro passo para implementação de sistemas neuro-*fuzzy*, abordagem extra não prevista e não utilizada neste trabalho. Para futuros trabalho neste sentido nesta área se recomendam os artigos de Kiguchi and Fukuda (1997), Fritzke (1997), Bouchaffra (2001) e Gupta (1994).

Outra possível limitação deste trabalho está relacionada com o fato de que não foram utilizados filtros passa baixas para diminuir o efeito ruidoso obtido com a derivada numérica pura e simples realizada sobre a informação captada pelos *encoders* do robô para inferir a velocidade atual de suas juntas. Cabe ressaltar que a primeira versão do sistema XO/2 encaminhada com o robô Inter/Scara previa esta filtragem, mas a segunda versão, utilizada neste trabalho, não realizava mais esta filtragem. Provavelmente se obteriam ações de controle muito menos oscilatórias usando um filtro derivativo.

Também poderiam ter sido avaliados outros algoritmos de treinamento para as redes MLP ou a camada de saída das redes RBF, como *quick-propagation* (Fahlman, 1988; Jondarr, 1996) ou *RPROP* (Riedmiller, 1994b; Riedmiller and Braun, 1993). Entretanto, Battistela (1999) já advertiu no seu trabalho, pelas simulações e experimentos realizados, que algoritmos de aprendizado mais rápidos levam à rápida saturação dos neurônios das redes MLP testadas (mesmo com baixas taxas de aprendizado). O algoritmo *backpropagation* expandido apesar de ser até conhecido pela sua "lentidão", se mostrou perfeitamente indicado para os controladores propostos neste trabalho. Foi percebido que este algoritmo foi rápido o suficiente para fazer com que os torques gerados pelas redes neurais preponderassem na ação de controle antes que 1/3 da trajetória houvesse sido completada pelo robô, mesmo no caso de mudanças abruptas de configuração entre um comando e outro de movimentação.

Dados os resultados alçados para os controladores integrados de força falta ainda investigar que tipos de informações (dados de entrada) devem ser usados com a malha de controle de força. Conforme verificado, informação atrasada pode ser útil durante o período de execução de um movimento restringido no espaço, mas apenas esta informação é insuficiente para garantir estabilidade do controlador em regime permanente.

Ainda, neste trabalho não foi possível realizar nenhuma prova de estabilidade relacionada com os controladores integrados propostos. Entretanto, fica como sugestões os trabalhos de:

- Gorez and Neyer (1994) que comentam sobre técnicas de controle de posição/ força em robôs manipuladores usando lógica *fuzzy*, onde os torques gerados pelo controlador

fuzzy estão contidos dentro de uma faixa que varia de $-\tau_{Máx}$ à $+\tau_{Máx}$. O mesmo ocorre com a saída de redes neurais MLP (saída limitadas pela função transferência sigmoidal adotada) ou com redes RBF com saída normalizadas (Haykin, 1999). Estes autores provam que, dada esta faixa limitada dos valores de torque de saída gerados (como se fosse um caso de controle descontínuo, ou um controle trabalhando com superfícies deslizantes), que estes controladores seriam uniformemente localmente estáveis (*uniformly ultimately bounded*). Kim and Lewis (1999) também afirmam que as redes neurais RBF utilizadas no seu controlador integrado proposto também possuem características de serem uniformemente localmente estáveis (idem no trabalho (Kim and Lewis, 2000)).

- Outros autores como: (Cajueiro and Hemerly, 2000) da área de lógica *fuzzy*, e Sio and Lee (1998) relacionado com redes neurais RBF, se baseiam na teoria do ganho pequeno (Crusius, 1996) para tentar provar a estabilidade de seus controladores.

- Kim and Lewis (1998) definem a regra para atualização dos pesos da sua rede baseado na teoria de estabilidade e Lyapunov e ainda baseado em propriedades de aproximação de funções continuas realizadas pelas redes do tipo MLP ou RBF (ver (Girosi and Poggio, 1993)), segundo a qual afirmam que num caso ideal, se a saída da rede pode ser expressada como:

$$y(x) = W^T \varphi(p) + \varepsilon(x)$$

os pesos treinados da rede, W são limitados por valores conhecidos, tal que: $\|W\|_F \leq W_M$.

7.6 Sugestões de Trabalhos Futuros

Algumas sugestões de trabalhos futuros podem ser feitas com base nos testes práticos realizados, entres eles:

a) Outros autores como Kiguchi and Fukuda (1997) partem para outra linha de investigação, se interessando não exclusivamente com as características próprias do controlador de força (seus ganhos), mas também em caracterizar a natureza do contato sendo realizado

em diferentes meios para diferentes tarefas (operação de polimento, desbaste de uma peça). É o que sugere (Volpe, 1990) na seção de "futuras direções": identificação em tempo-real de parâmetros do meio usando teoria de controle adaptativo – para estimação *on-line* dos ganhos mais adequados para controle de força. Foi a sugestão que Kiguchi and Fukuda (1997) considerou no seu sistema de controle de força usando redes neuro-*fuzzy* para identificar características do meio sob contato de forma à definir a velocidade com que uma ferramenta de polimento deveria seguir sobre a peça (na direção não restringida) e ainda definir o ângulo de "ataque" mais adequado da ferramenta para realizar a operação de polimento (note que Kiguchi and Fukuda (1997) apenas realizaram simulações obtendo resultados razoáveis) – ver figura 7.1.

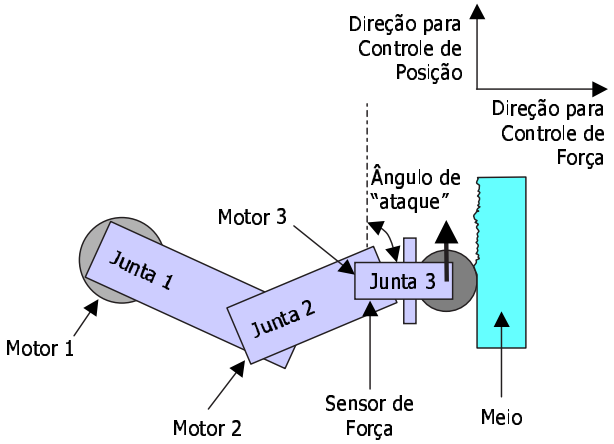


Figura 7.1: Exemplo de efetuador final para tarefa de polimento de peças (Kiguchi and Fukuda, 1997).

b) O mais adequado na área de controle de força/posição parece ser um sistema capaz de extrair características da natureza do contato sendo realizado e com base no mapeamento destas informações, definir a velocidade com que a peça deve ser contornada (implica em interação com o gerador de trajetórias normalmente já atrelado a um sistema de controle de posição/força) e mesmo adaptar de maneira *on-line* os ganhos do controlador de força sendo usado (assim seus ganhos variam conforme as características do meio sob contato), além do que, tal como realizaram Kiguchi and Fukuda (1997) na sua simulação, desenvolver o contato com um efetuador final capaz de permitir variações sutis no contato (como o tal "ângulo de ataque"). Este efetuador final é particularmente relevante para suprimir reações drásticas, abruptas e indesejadas relacionadas com o controlador de força. A figura 7.1 mostra o efetuador final utilizado por Kiguchi and Fukuda (1997).

c) Além disso, é particularmente desejável que o efetuador final agregue consigo alguma ”passividade” (elemento elástico) ao contato realizado principalmente no caso de estruturas muito rígidas (robô + meio) cujas forças de contato envolvidas podem excitar dinâmicas (normalmente não modeladas) de alta ordem o que pode comprometer a estabilidade do controlador dependendo dos ganhos utilizados (a margem de ganho baixa consideravelmente quanto mais rígido for o contato realizado – ver (Eppinger and Seering, 1987; Volpe, 1990)). Quanto mais rígida a forma de contato, mais difícil será realizar o controle de força de forma segura, além de serem exigidas elevadas freqüências de amostragem para o laço de controle de força (Volpe, 1990). Conforme parecem sugerir os testes realizados com o robô Inter/Scara, períodos de amostragem superiores à 1[ms] como no caso 2[ms] que foi o adotado para operação do controlador integrado de força, implicam em possível perda de controle sobre o sistema. As respostas do meio sobre o robô se tornam muito rápidas e rapidamente acentuadas principalmente no caso de ganhos demasiadamente elevados para a malha de controle de força. O sistema gerado pelo contato do robô + meio envolvendo uma realimentação direta com informações advindas de um sensor de força (seja um simples controlador proporcional, PI ou PD de força) se torna equivalente a um controle de impedância de 2a ordem (Volpe, 1990). Nestes casos, ganhos elevados podem ser traduzidos como um fator de amortecimento negativo erroneamente imposto ao sistema (o sistema começa a oscilar), o que na área de mecânica se traduz num fenômeno conhecido como: ”vibrações auto-excitadas” (den. Hartog, 1972).

d) Portanto, estas seriam indicações para direções futuras de pequisa verificando a aplicação de controladores de força com capacidade de auto-adaptação de seus próprios ganhos com base em características do contato extraídas pelo próprio sistema com base nas informações coletadas pelo sensor de força e outras ainda.

e) Avaliar o impacto de outros algoritmos de treinamento, em especial o *quick-propagation* por se tratar de um método mais eficiente para selecionar a direção do treinamento da rede sem o problema de uma eventual paralisação de uma rede num ponto de mínimo local no algoritmo ”puro” *back-propagation*. Entretanto, (Battistela, 1999) afirmou que suas redes neurais com algoritmos de aprendizado muito rápidos levaram à instabilidade de seus controladores integrado também baseados na arquitetura de retropapagação de erros (*feedback-error learning*). Durante os ensaios realizados nesta tese, o tradicional algo-

ritmo *back-propagation* expandido com termo *momentum* se mostrou rápido suficiente para garantir os bons desempenhos demonstrados pelos controladores integrados de posição.

f) Incrementar o controlador integrado de posição baseado na rede neural RBF com algoritmos capazes de adaptar *on-line* os parâmetros das funções gaussinas: como ajustar a distância entre os centros das funções e ajustar sua variância. O objetivo seria verificar se estas novas capacidade de adaptação da rede RBF para este tipo de aplicação induzem a perda de capacidade de aprendizado global desta rede. Eventualmente um algoritmo capaz de ajustar os centros e larguras das funções gaussianas pode induzir a um aprendizado nitidamente local, já que são conhecidas por sua boa capacidade de interpolação mas baixa capacidade de extrapolação (se comparadas com as redes MLP baseadas em funções sigmoidais (Keller, 1999)). O controlador obteria então melhores desempenhos apenas em certas regiões de operação do robô ou nos casos de operações repetitivas com o robô. Entretanto, esta mesma capacidade de aprendizado local das redes RBF pode ser altamente desejável para auxiliar numa tarefa de caracterização do contato no caso de controle de força. Neste último caso, acredita-se que tal rede poderia ser utilizada para ajustar de maneira ótima os ganhos do próprio controlador de força sendo utilizado e ainda interagir com o gerador de trajetória a fim de corrigir a velocidade de movimentação (impor pequenas variações na velocidade já estipulada pelo gerador de trajetórias) por sobre a peça na qual está sendo realizado contato – uma vez que estes parâmetros variam em função do meio com o qual o robô está realizando o contato e com a própria natureza da tarefa envolvida com o contato.

g) Incorporar alguma técnica ou estratégia para melhorar o estágio referente ao limiar de contato com o meio e minimizar a ação de controle no caso de perda abrupta de contato. Volpe (1990) usou uma abordagem baseado em "potencias de repulsão", similares aos adotados na área de robótica móvel para desviar robôs móveis de obstáculos.

h) Incorporar um sensor de proximidade ao efetuador final do robô para melhorar e facilitar a estratégia para minimizar os problemas envolvidos com o primeiro contato.

i) Avaliar o uso mesclado de redes neurais com lógica *fuzzy*, da área de sistemas neuro-*fuzzy* para o controle integrado de posição/força de robôs manipuladores (Gupta, 1994; Lin and Lin, 1997; Shan et al., 1996; Fritzke, 1997) ou o uso de um sistema de inferência*fuzzy*

baseado em redes neurais adaptativas (ANFIS = *Adaptive-Network-Based Fuzzy Inference System*) – citejang93anfis. Por exemplo, Kiguchi and Fukuda (1996) descrevem um sistema neuro-*fuzzy* como um método para compensar atrito para controladores de posição/força de robôs manipuladores. Assim como Kuo (1997) descreve um sistema neuro-*fuzzy* para definir parâmetros de contato para uma tarefa de polimento de superfícies usando robô manipulador.

j) Incorporar ao robô um efetuador capaz de realizar tarefas de polimento de forma a averiguar na prática estratégias para controle de força.

k) Se recomenda a implementação de alguma técnicas para lidar de forma mais adequada com o problema da estratégia do contato com o meio, eventualmente baseadas em forças "artificiais" na qual potenciais de energia são criados e superfícies isopotenciais de contato são definidas em torno do ponto de contato (Volpe, 1990; Volpe and Khosla, 1993).

l) Avaliar uma nova arquitetura para o controlador integrado de força, inspirado nos controladores convencionais de controle por impedância. Segundo (Volpe and Khosla, 1993; Volpe, 1990) o controle de força explícito que fecha a malha de realimentação com base nas informações captadas pelo sensor de força, mesmo usando apenas um controlador proporcional pode ser considerado uma forma de controle de força por impedância. Assim, a idéia é sobrepor um modelo de referência à malha mais externa de controle de força de forma a forçar a saída do sistema para certo tipo de resposta desejada (com certo fator de amortecimento e *overshoots* dentro de faixas toleráveis). Uma rede neural poderia ser atrelada ao sistema de controle de força de forma a realizar as compensações dinâmicas necessárias para corrigir o perfil desejado de resposta que um controlador convencional simples não seria capaz de garantir. Diferentes sinais de erro poderiam ser enviados para a rede: (1) a saída do controlador convencional (saídas não nulas significam que a rede não está realizando a compensação dinâmica necessária) ou (2) o sinal que compara a saída real do sistema com o perfil desejado para a resposta do sistema. Acredita-se que a primeira opção seja a melhor, pois como foi verificado nos ensaios práticos realizados neste trabalho, se a taxa de aprendizado e termo *momentum* são adequados para a rede em questão, toda vez que o controlador convencional começa a reagir e entregar um torque de saída preponderante sobre o torque gerado pela RN, significa que está no momento

da rede reiniciar seu aprendizado como reflexo da necessidade de um novo reajuste na compensação dinâmica que a rede vinha realizando. Acredita-se que a segunda opção faça o controlador integrado perder sua capacidade de re-adaptação em caso de mudança abrupta na forma do robô executar sua tarefa.

m) uma outra arquitetura de controlador integrado (usando rede neural) e mesclando uma técnica da área de controle avançado baseado em modelo de referência (MRAC) (Butler, 1990) está sendo sugerida, conforme mostra a figura 7.2. Modelo inspirado em propostas semelhantes já avaliadas por (Lightbody and Irwin, 1997; Widrow and Plett, 1997; Yilditrim and Sukkar, 1996) e levando em consideração os preceitos citados por Volpe (1990) referente à controle de força usando realimentação com base no sensor de força, que resulta num sistema em malha fechada que pode ser tratado como uma impedância mecânica. O modelo de referência estabeleceria a resposta desejada para o contato, incluindo eventualmente a definição da forma do transitório de contato com o meio. Notar que diferente das propostas anteriores de controladores usando RNs baseados no conceito de MRAC, um controlador convencional continua sendo utilizado em paralelo com a rede neural para garantir robustez ao sistema no caso de saturação dos pesos da rede. Mas ao contrário da abordagem usada antes para os controladores integrados neurais avaliados neste trabalho, o sinal de erro usado para treinamento da rede é baseado no erro entre a resposta real do sistema e a saída desejada para o sistema (bloco "modelo por referência da figura 7.2. Mas continuaria sendo necessário avaliar qual seria o melhor "pacote" de entrada de dados à ser utilizado pela rede. A figura 7.2 não inclui as transformações de espaços envolvidas nas malhas para controle de força e de posição. Não aparece explicitado a transformação de coordenadas do espaço tarefa para o espaço operacional e do espaço operacional para eventualmente espaço de juntas (se o controlador convencional é especificado para o espaço de juntas – opção mais adequada), nem as transformações reversas. E nem mesmo as transformações de informações captadas pelos sensor de força do espaço-sensor para o espaço-tarefa e do espaço tarefa para o espaço operacional do robô (ou diretamente do espaço-sensor para o espaço operacional como foi realizado neste trabalho).

n) Outra sugestão é implementar e testar uma rede neural para trabalhar no esquema de controle híbrido de posição/força, mas neste caso, uma única rede trabalharia com os da-

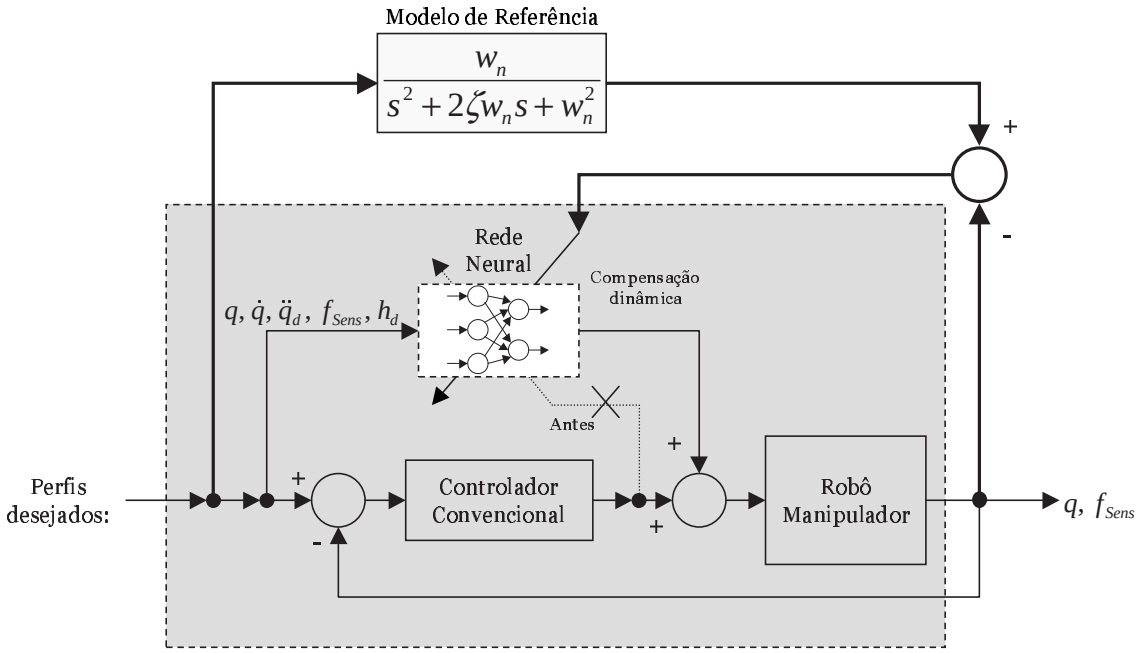


Figura 7.2: Novo modelo proposto de controlador integrado de força/posição usando rede neural, controlador convencional baseado na abordagem MRAC.

dos de entrada usados tanto pelos controladores da malha de controle de posição quanto da malha de controle de força. E esta única rede repassaria seus dados de saída (torques) para as correspondentes malhas de acordo com o que foi especificado pela matriz de seleção, S. Esta rede, referenciada como "RN-fx" trabalharia com o seguinte pacote de entrada de dados: $x_{RN} = [\,q \quad \dot{q} \quad \ddot{q}_d \quad x \quad f_{Sens} \quad h_d\,]$. O vetor de dados x e h_d seriam passados para a rede depois de filtrados pela matriz S de seleção. Desta forma, os dados referentes à configuração atual do robô no espaço operacional que efetivamente são passadas para a rede seriam: $x = [\,0\ 0\ z\ 0\,]$, e os dados referentes ao vetor referência da força de contato desejada, h_d, seriam: $h_d = [\,0\ 0\ h_{d_z}\ 0\,]$. A figura 7.3 mostra o fluxo de dados processado por este tipo de rede.

Diferente das outras redes, a informação de erro usada para disparar a rotina de aprendizado da mesma varia conforme o as direções em que a matriz de Seleção especificam controle de posição ou controle de força. As redes anteriores à esta (da malha de controle de força) usavam como informação de erro de entrada para seu algoritmo de *backpropagation* os torques gerados pelos controladores de força apenas. No caso particular de controle de força na direção Z apenas, com base no especificado através da matriz de seleção, apenas o erro à ser retropropagado relacionado com a saída da rede para a junta

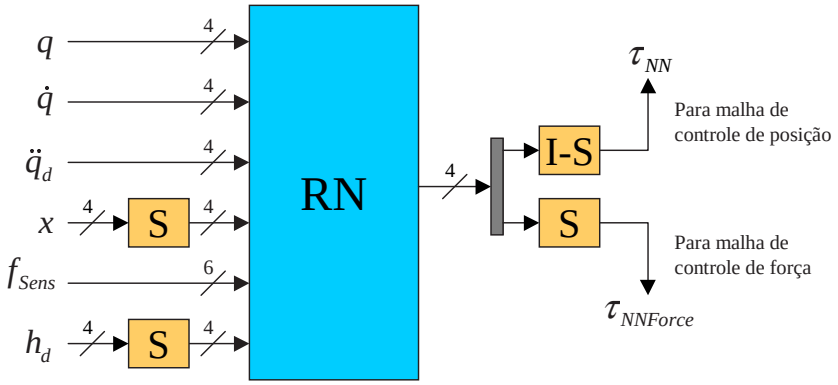


Figura 7.3: Dados de entrada e de saída para proposta da nova rede RN-fx.

2 seria captado à partir do torque gerado pelo controlador convencional de força para esta direção (junta 2). As informações restantes de erro, referentes às saída da rede para as juntas 0, 1 e 3 seriam captadas à partir dos torques gerados pelos controladores de posição.

Isto acaba caracterizando um modo "misto" de operação para esta rede. O objetivo desta rede seria realizar tanto a compensação dinâmica e ação de controle necessária tanto para a malha de controle de posição quanto para a malha de controle de força, evitando-se assim a necessidade de redes neurais diferentes para cada uma das malhas; o que implicaria ainda como vantagem, menor poder de processamento exigido da CPU do robô. Provavelmente dada a capacidade de múltipla correlação entre as variáveis de entrada e de saída da rede neural, é de se esperar um desempenho superior já que a rede seria capaz de lidar com alguma eventual interação que possa haver entre controle de força e controle de posição.

o) Incorporar ao sistema do robô, uma câmera de vídeo para aprimorar a trajetória sendo programada pelo gerador de trajetórias do robô. Uma câmera pode também permitir o robô se desviar de obstáculos dinâmicos que possam se apresentar na sua área de trabalho. Neste caso, abre-se uma nova área de estudo relativa a estratégias para desvio de obstáculos em tempo-real. Provavelmente a atual CPU do robô teria de ser substituída por um processador mais rápido com maior capacidade de cálculo em ponto flutuante a não ser que o sistema que processe as informações geradas por uma câmera de vídeo rode numa outra CPU dedicada só para a parte de processamento de vídeo e integração sensorial com os dados do robô.

p) Estudar de forma aprofundada a problemática de estabilidade dos controladores integrados (neural + convencional).

Apêndice A

Transformações de Coordenadas

A.1 Do Espaço de Juntas para o Espaço Operacional

A.1.1 Cinemática Direta

$$x = K(q) + K_{Offset}(q, dx) \tag{A.1}$$

$$K(q) = \begin{cases} l_1 \cos 0 + l_2 \cos 01 \\ l_1 \sin 0 + l_2 \sin 01 \\ l_{0z} + q_2 + k \cdot q_3 \\ q_0 + q_1 + q_3 \end{cases} \tag{A.2}$$

onde:
$$\begin{aligned} \cos 0 &= \cos(q_0); \\ \sin 0 &= \sin(q_0); \\ \cos 01 &= \cos(q_0 + q_1); \\ \sin 01 &= \sin(q_0 + q_1); \end{aligned}$$

$$K_{Offset}(q, dx) = \begin{cases} dx_0 \cos 013 - dx_1 \sin 013 \\ dx_0 \sin 013 + dx_1 \cos 013 \\ -l_{3z} + dx_2 \\ 0 \end{cases} \tag{A.3}$$

onde: $\cos 013 \quad = \cos(q_0 + q_1 + q_3)$;
$\sin 013 \quad = \sin(q_0 + q_1 + q_3)$;

A idéia é processar algo como mostrado na forma de diagrama em blocos na figura A.1 à seguir.

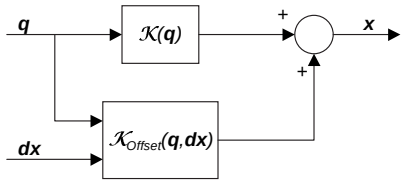


Figura A.1: Diagrama de blocos mostrando processamento da cinemática direta

A.1.2 Jacobiano Analítico

Definição:

$$\dot{x} = \underbrace{\frac{\partial K(q)}{\partial q}}_{J_A(q)} \cdot \dot{q} + \underbrace{\frac{\partial K_{Offset}(q,dx)}{\partial q}}_{J_{A_{Offset}}(q,dx)} \cdot \dot{q}$$

ou:

$$\dot{x} = \frac{\partial K_{Full}(q,dx)}{\partial q} \cdot \dot{q}$$

que resulta em:

$$\begin{array}{rcccccc}
\dot{x}_0 & = & -l_1\ s0 \cdot \dot{q}_0 & -l_2 s01 \cdot (\dot{q}_0 + \dot{q}_1) & -\mathbf{dx_0} s013 \cdot (\dot{q}_0 + \dot{q}_1 + \dot{q}_3) & -\mathbf{dx_1} c013 \cdot (\dot{q}_0 + \dot{q}_1 + \dot{q}_3) \\
\dot{x}_1 & = & l_1 c0 \cdot \dot{q}_0 & +l_2 c01 \cdot (\dot{q}_0 + \dot{q}_1) & +\mathbf{dx_0} c013 \cdot (\dot{q}_0 + \dot{q}_1 + \dot{q}_3) & -\mathbf{dx_1} s013 \cdot (\dot{q}_0 + \dot{q}_1 + \dot{q}_3) \\
\dot{x}_2 & = & \dot{q}_2 & +\mathbf{k} \cdot \dot{q}_3 & & \\
\dot{x}_3 & = & \dot{q}_0 & +\dot{q}_1 & +\dot{q}_3 &
\end{array}$$

Ou na forma matricial:

$$J_A(q) = \begin{bmatrix} -l_1 s0 - l_2 s01 & -l_2 s01 & 0 & 0 \\ l_1 c0 + l_2 c01 & l_2 c01 & 0 & 0 \\ 0 & 0 & 1 & k \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{A.4}$$

Observação: Note o termo k que reflete o acoplamento mecânico existente entre as duas últimas juntas do robô. Perceba que toda vez que a junta q_3 sofre alteração no seu torque, é propagado um movimento também para a junta anterior q_2, de tal forma que para movimentos positivos de q_3 (giro no sentido anti-horário), o TCP do robô sobe (há um acréscimo em x_2).

e temos ainda:

$$J_{A_{Offset}}(q, dx) = \begin{bmatrix} k_1 & k_1 & 0 & k_1 \\ k_2 & k_2 & 0 & k_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.5}$$

onde: $k_1 = -dx_0 \sin 013 \quad -dx_1 \cos 013$;

$k_2 = dx_0 \cos 013 \quad -dx_1 \sin 013$;

Observação: assume-se que somente as variáveis $q = \mathcal{F}(t)$ e que a variável dx, que indica o *offset* acrescentado ao sistema de coordenadas de base do robô (espaço operacional ou também dito espaço cartesiano) pelo encaixe de uma ferramenta na última junta do robô, não varie com o tempo (o **TCP**(*Tool Center Point*) da ferramenta, não varia com o tempo).

A.1.3 Derivada do Jacobiano Analítico

Definição:

$$\ddot{x} = J_A(q) \cdot \ddot{q} + \dot{J}_A(q, \dot{q}) \cdot \dot{q} + J_{A_{Offset}}(q, dx) \cdot \ddot{q} + \dot{J}_{A_{Offset}}(q, dx, \dot{q}) \cdot \dot{q}$$

onde:

$$\dot{J}_A(q, \dot{q}) = \begin{bmatrix} -l_1 \cos 0 \cdot \dot{q}_0 - l_2 \cos 01 \cdot (\dot{q}_0 + \dot{q}_1) & -l_2 \cos 01 \cdot (\dot{q}_0 + \dot{q}_1) & 0 & 0 \\ -l_1 \sin 0 \cdot \dot{q}_0 - l_2 \sin 01 \cdot (\dot{q}_0 + \dot{q}_1) & -l_2 \sin 01 \cdot (\dot{q}_0 + \dot{q}_1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.6}$$

e:

$$\dot{J}_{A_{Offset}}(q, dx, \dot{q}) = \begin{bmatrix} k_3 & k_3 & 0 & k_3 \\ k_4 & k_4 & 0 & k_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.7}$$

onde: $k_3 = -dx_0 \cos 013 \quad +dx_1 \sin 013;$

$k_4 = -dx_0 \sin 013 \quad -dx_1 \cos 013;$

A.2 Do Espaço Operacional para o Espaço de Juntas

A.2.1 Inversa do Jacobiano Analítico

Definição:

$$\dot{q} = J_A^{-1}(q) \cdot \dot{x}$$

Note que nem sempre esta matriz é inversível, e sua determinação numérica pode ser difícil de ser obtida se o robô se encontrar próximo se uma região de singularidade (alguns termos na matriz acima se tornam nulos e o determinante do Jacobiano se aproxima de zero). Para estes casos, podem ser tentados o método da *pseudo-inversa* (Sciavicco and Siciliano, 1996)(p. 92, eq. 3.55) que consiste em fazer:

$$J_A^{\diamondsuit} = J_A^T \cdot \left(J_A \cdot J_A^T\right)^{-1}$$

ou ainda pode ser usado o método da *inversa do menor quadrado amortecido* (*dampest least-squares inverse [DLS]*) (Sciavicco and Siciliano, 1996)(p. 95, eq. 3.62) que consiste em fazer:

$$J_A^* = J_A^T \cdot \left(J_A \cdot J_A^T + k^2 \cdot I\right)^{-1}$$

onde: k se refere a um fator de amortecimento.

A derivada analítica de $J_A(q)$ fica:

$$J_A^{-1}(q) = \begin{bmatrix} -\frac{\cos 01}{l_1 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & -\frac{\sin 01}{l_1 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & 0 & 0 \\ \frac{l_1 \cdot \cos 0 + l_2 \cdot \cos 01}{l_2 \cdot l_1 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & \frac{l_1 \cdot \sin 0 + l_2 \cdot \sin 01}{l_2 \cdot l_1 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & 0 & 0 \\ \frac{k \cos 0}{l_2 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & \frac{k \sin 0}{l_2 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & 1 & -k \\ -\frac{\cos 0}{l_2 \cdot [\cos 01 \sin 0 - \sin 01 \cos 0]} & -\frac{\sin 0}{l_2 \cdot (\cos 01 \sin 0 - \sin 01 \cos 0)} & 0 & 1 \end{bmatrix} \tag{A.8}$$

Lembrando das relações trigonométricas entre funções de 2 ângulos:

$$\begin{aligned} \sin(\alpha \pm \beta) &= \sin\alpha \cdot \cos\beta \pm \cos\alpha \cdot \sin\beta \\ \cos(\alpha \pm \beta) &= \cos\alpha \cdot \cos\beta \mp \sin\alpha \cdot \sin\beta \end{aligned}$$

e aplicando na equação A.8 notamos então que:

$$\begin{aligned} \cos 01 \sin 0 - \sin 01 \cos 0 &= \sin 0 \cos 01 - \cos 0 \sin 01 \\ &= \sin(-1) \\ &= -\sin 1 \end{aligned}$$

e então, temos que:

$$J_A^{-1}(q) = \begin{bmatrix} \frac{\cos 01}{l_1 \cdot \sin 1} & \frac{\sin 01}{l_1 \cdot \sin 1} & 0 & 0 \\ -\frac{l_1 \cdot \cos 0 + l_2 \cdot \cos 01}{l_2 \cdot l_1 \cdot \sin 1} & -\frac{l_1 \cdot \sin 0 + l_2 \cdot \sin 01}{l_2 \cdot l_1 \cdot \sin 1} & 0 & 0 \\ -\frac{k \cos 0}{l_2 \cdot \sin 1} & -\frac{k \sin 0}{l_2 \cdot \sin 1} & 1 & -k \\ \frac{\cos 0}{l_2 \cdot \sin 1} & \frac{\sin 0}{l_2 \cdot \sin 1} & 0 & 1 \end{bmatrix} \tag{A.9}$$

Mas esta matriz não leva em conta o *offset* produzido pelo acréscimo de uma ferramenta ao robô. Levando em conta a matriz $J_{A_{Offset}}(q, dx)$ podemos definir a matriz: $J_{A_{Full}}$ como sendo:

$$J_{A_{Full}} = \begin{bmatrix} -l_1 s0 - l_2 s01 + \mathbf{k_1} & -l_2 s01 + \mathbf{k_1} & 0 & \mathbf{k_1} \\ l_1 c0 + l_2 c01 + \mathbf{k_2} & l_2 c01 + \mathbf{k_2} & 0 & \mathbf{k_2} \\ 0 & 0 & 1 & k \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{A.10}$$

então:

$$J^{-1}_{A_{Full}} = \begin{bmatrix} -\frac{l_2c01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & -\frac{l_2s01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & 0 & \frac{\mathbf{k_2}\cdot l_2s01+\mathbf{k_1}\cdot l_2c01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} \\ \frac{l_1c0+l_2c01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & \frac{l_1s0+l_2s01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & 0 & -\frac{\mathbf{k_2}\cdot l_1s0+\mathbf{k_2}\cdot l_2s01+\mathbf{k_1}\cdot l_1c0+\mathbf{k_1}\cdot l_2c01}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} \\ \frac{kl_1c0}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & \frac{kl_1s0}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & 1 & -\frac{k(l_2c01\cdot l_1s0+\mathbf{k_2}\cdot l_1s0-l_2s01\cdot l_1c0+\mathbf{k_1}\cdot l_1c0)}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} \\ -\frac{l_1c0}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & -\frac{l_1s0}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} & 0 & \frac{l_2c01\cdot l_1s0+\mathbf{k_2}\cdot l_1s0-l_2s01\cdot l_1c0+\mathbf{k_1}\cdot l_1c0}{l_2c01\cdot l_1s0-l_2s01\cdot l_1c0} \end{bmatrix} \tag{A.11}$$

Por inspeção na equação anterior A.11 notamos que:

$$l_2c01 \cdot l_1s0 - l_2s01 \cdot c0 = -l_1 \cdot l_2 \cdot s1$$

Então finalmente obtemos:

$$J^{-1}_{A_{Full}} = \begin{bmatrix} \frac{c01}{l_1s1} & \frac{s01}{l_1s1} & 0 & -\frac{\mathbf{k_1}c01+\mathbf{k_2}s01}{l_1s1} \\ -\frac{l_1c0+l_2c01}{l_1l_2s1} & -\frac{l_1s0+l_2s01}{l_1l_2s1} & 0 & \frac{\mathbf{k_1}(l_1c0+l_2c01)+\mathbf{k_2}(l_1s0+l_2s01)}{l_1l_2s1} \\ -\frac{kc0}{l_2s1} & -\frac{ks0}{l_2s1} & 1 & -\frac{k(-l_1l_2s1+\mathbf{k_1}l_1c0+\mathbf{k_2}l_1s0)}{-l_1l_2s1} \\ \frac{c0}{l_2s1} & \frac{s0}{l_2s1} & 0 & \frac{-l_1l_2s1+\mathbf{k_1}l_1c0+\mathbf{k_2}l_1s0}{-l_1l_2s1} \end{bmatrix} \tag{A.12}$$

Esta equação pode então ser organizada em 2 partes considerando:

$$\dot{q} = J_A^{-1}(q) \cdot \dot{x} + J^{\dagger}_{A_{Offset}}(q, dx) \cdot \dot{x} \tag{A.13}$$

Agrupando e organizando os termos da equação A.12 com A.13 obtemos:

$$\dot{q} = \underbrace{\begin{bmatrix} \frac{c01}{l_1 s1} & \frac{s01}{l_1 \cdot s1} & 0 & 0 \\ -\frac{l_1 c0 + l_2 c01}{l_2 l_1 s1} & -\frac{l_1 s0 + l_2 s01}{l_2 l_1 s1} & 0 & 0 \\ -\frac{kc0}{l_2 s1} & -\frac{ks0}{l_2 s1} & 1 & -k \\ \frac{c0}{l_2 s1} & \frac{s0}{l_2 s1} & 0 & 1 \end{bmatrix}}_{J_A^{-1}} \cdot \dot{x} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & -\frac{(\mathbf{k_1} c01 + \mathbf{k_2} s01)}{l_1 s1} \\ 0 & 0 & 0 & \frac{\mathbf{k_1}(l_1 c0) + \mathbf{k_2}(l_1 s0 + l_2 s01)}{l_1 l_2 s1} \\ 0 & 0 & 0 & \frac{k(\mathbf{k_1} c0 + \mathbf{k_2} s0)}{l_2 s1} \\ 0 & 0 & 0 & -\frac{(\mathbf{k_1} c0 + \mathbf{k_2} s0)}{l_2 s1} \end{bmatrix}}_{J^{\dagger}_{A_{Offset}}} \cdot \dot{x} \tag{A.14}$$

A.2.2 Aceleração do Espaço Operacional para o Espaço de Juntas

$$\ddot{q} = J^{-1}_{A_{Full}}(q) \cdot \left(\ddot{x} - \dot{J}_{A_{Full}}(q, \dot{q}) \cdot \dot{q} \right) \tag{A.15}$$

A.3 Simulações

As equações anteriores foram validadas antes no MATLAB antes de serem aplicadas do robô – ver figura A.2.

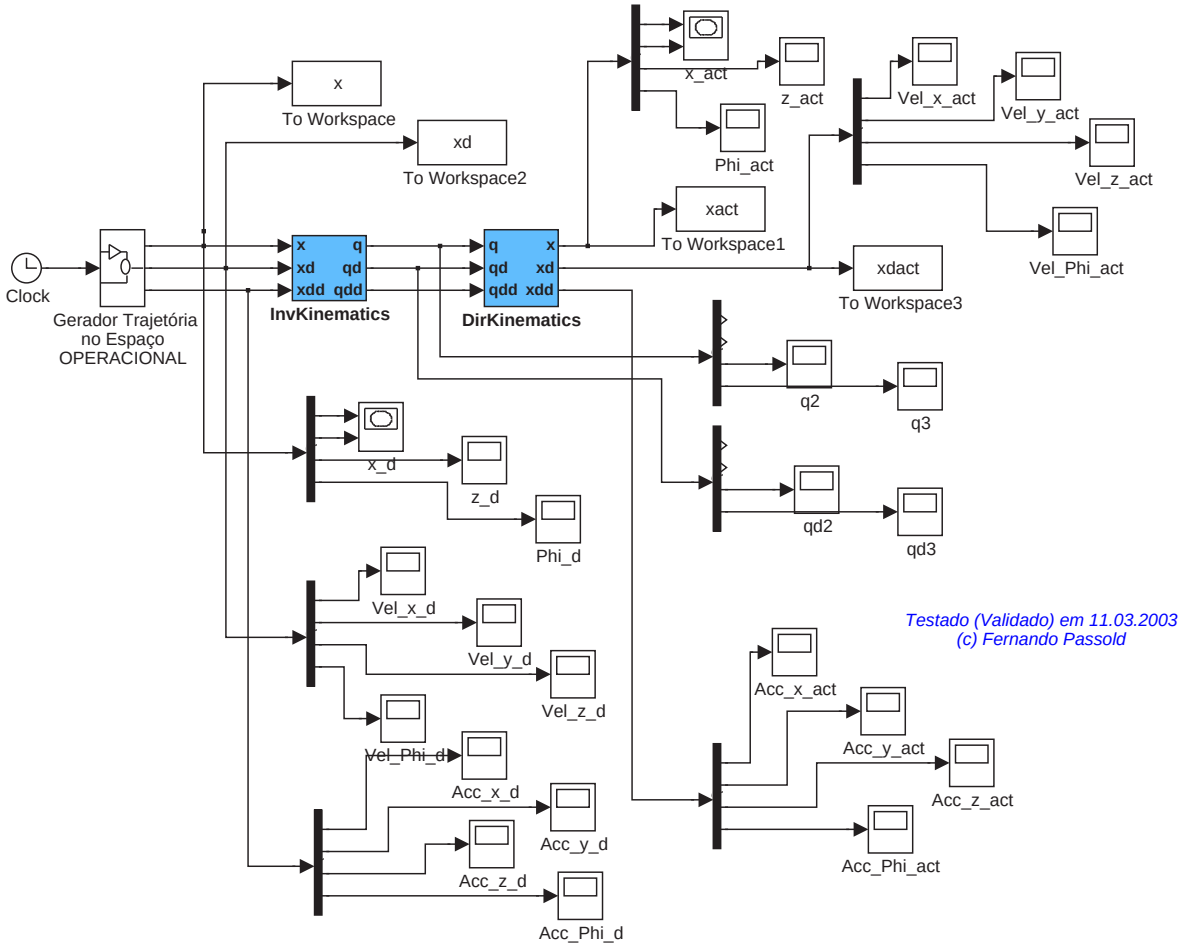


Figura A.2: Teste dos blocos de cinemática direta e inversa no MATLAB

Apêndice B

Controladores PID Digitais

B.1 Formas de Implementação

A forma no domínio de sinais contínuos no tempo de um PID pode ser descrita como (Seborg et al., 1989):

$$u(t) = K_p e(t) + K_i \int_o^t e(t)dt + K_d \frac{\partial e(t)}{\partial t} \tag{B.1}$$

A versão digital adotada para implementar este algoritmo aparece descrito na equação abaixo:

$$u[k] = K_p \left[e[k] + \frac{T}{\tau_i} \sum_{k=0}^{n} e[k] + \frac{\tau_d}{T} \left(e[k] - e[k-1] \right) \right] \tag{B.2}$$

ou como foi finalmente implementado:

$$u[k] = K_p e[k] + K_i \sum_{k=0}^{n} e[k]T + K_d \frac{(e[k] - e[k-1])}{T} \tag{B.3}$$

Esta implementação se baseou no modo direto de derivação da lei de controle do PID da equação B.1. A integral foi aproximada via um somatório e a componente derivativa foi obtida através de uma equação de diferenças de primeira ordem. Esta forma de implementação também é conhecida como PID no formato de posição (Seborg et al., 1989). Existem outras formas ainda de implementar algoritmos de PID digitais, entre elas o formato de velocidade (Seborg et al., 1989) e o PID 2 DOF. Optou-se pela forma mais simples e rápida.

No formato de velocidade a equação do PID passa a ser (Seborg et al., 1989):

$$u[k] = u[k-1] + K_c \left[(e[k] - e[k-1]) + \frac{T}{\tau_i} e[k] + \frac{\tau_d}{T} (e[k] - 2e[k-1] + e[k-2]) \right] \quad \text{(B.4)}$$

ou (Nascimento Jr. and Yoneyama, 2000):

$$u[k] = u[k-1] + a_0 e[k] + a_1 e[k-1] + a_2 e[k-2] \quad \text{(B.5)}$$

onde:

$$\begin{aligned} a_0 &= K_p + K_i T + \tfrac{K_d}{T} \\ a_1 &= -\left(K_p + 2\tfrac{K_d}{T} \right) \\ a_2 &= \tfrac{K_d}{T} \end{aligned}$$

O formato de velocidade para implementação do PID traz como vantagens o fato de eliminar o problema da saturação na ação integral uma vez que não é calculado de forma explícita o somatório dos erros.

B.2 Ajuste de Controladores PID

O ajuste dos controladores foi realizado baseado no método de Ziegler-Nichols (Nise, 2000; Franklin et al., 1994; Seborg et al., 1989), conforme ilustra a tabela B.1 à seguir. Onde K_u refere-se ao ganho limite, máximo ou "*ultimate gain*" que pode ser aplicado num sistema e que torna a saída da planta oscilatória (com amplitude constante ainda). O período no qual o sistema começa a oscilar é caracterizado como T_u (ou "*ultimate period*"). Se for utilizado um controlador Proporcional puro com ganho K_p maior que K_u o sistema se torna instável. Se este ganho, $K_p < K_u$ a saída do sistema tende a ser sub-amortecida.

Controlador	K	τ_i	τ_d
P	$0.5\,K_u$	–	–
PI	$0.45\,K_u$	$T_u/1.2$	–
PID	$0.6\,K_u$	$T_u/2$	$T_u/8$

Tabela B.1: Relações de Ziegler-Nichols para ajuste de controladores PID.

Apêndice C

Algoritmo *back-propagation* expandido com termo *momentum*

C.1 Introdução

Todos os pesos de uma rede do tipo MLP ou os pesos da camada de saída de uma rede RBF podem ser reajustados de maneira supervisionada usando o algoritmo muito popular conhecido como algoritmo de retropropagação de erros ou mais simplesmente como algoritmo *back-propagation*. Este algoritmo é baseado na regra de aprendizado de correção de erros que pode ser tratada também como uma generalização de um algoritmo de filtragem adaptativo: o algoritmo de menor erro quadrático médio ou LMS (*Least-Mean-Squared*). Basicamente este algoritmo se divide em duas etapas (duas passagens sobre as diferentes camadas da rede): uma de processamento direto (*forward pass*) e uma etapa ou passagem de retropropagação de erros. No modo direto, um certo padrão de entrada, é aplicado à camada sensorial da rede e seu efeito propagado pelo restante da rede, camada por camada, sempre no sentido da camada de entrada para a de saída, até que seja alcançada a camada de saída. Finalmente então um conjunto de sinais de saída é produzido na camada de saída correspondendo a resposta atual da rede. Durante este processo, os pesos sinápticos da rede permanecem todos fixos (não mudam). Pode ocorrer então a etapa de retropopagação dos erros, quando então, os pesos sinápticos são ajustados de acordo com alguma regra de correção de erros. Mais especificamente, a resposta atual da rede é subtraída da resposta desejada de forma a produzir um sinal de erro. Este sinal de erro é então propagado de trás

para frente da rede (da camada de saída para a camada de entrada da rede), dai advindo o nome do algoritmo de retropropagação de erros o mais simplemente *back-propaagation*. Os pesos sinápticos são então ajustados de maneira a fazer com que a resposta atual da rede se aproxime o máximo possível da resposta desejada para o padrão atual aplicado à sua entrada.

C.2 O Algoritmo *Back-propagation*

Seja uma rede MLP contendo L camadas cada qual contendo m neurônios. Numa primeira etapa para uso deste algoritmo se faz necessário antes cumprir a etapa de processamento de modo direto da rede, em que a informação contida no vetor de entrada $\mathbf{x}$, é propagada pelos pesos sinápticos da rede, desde a sua camada de entrada até a de saída, gerando por fim o vetor de saída calculada da rede, $\mathbf{y}$. Somente então, este algoritmo pode ser aplicado, pois é baseado no gradiente dos erros de saída da rede.

Quando se implementa de maneira *on-line* o algoritmo *back-propagation*, as etapas direta e de retropropagação de erros são executadas a cada vez que um novo padrão de treinamento n é apresentado à rede. Para cada padrão corresponde então uma entrada e sua correspondente saída, perfazendo um conjunto de pares de treinamento: $\{(x(n), t(n))\}_{n=1}^{N}$. Normalmente a rede têm seus pesos sinápticos inicializados com valores aleatórios uniformemente distribuídos cuja média é zero e cuja variância é determinada de tal forma a forçar com que os potenciais de indução de cada neurônio se concentrem entre a parte linear e de saturação da função sigmóide usada como curva de ativação para os mesmos.

Na primeira etapa, a de modo direto, o vetor de entrada $\mathbf{x}$ é aplicado a entrada da rede. Cada uma destas entradas, induz um potencial de ativação, v_j^l à cada um dos j neurônios da camada l seguinte, dado por:

$$v_j^{(l)} = \sum_{i=0}^{m} w_{ij}^{(l)} \cdot y_i^{(l-1)} \tag{C.1}$$

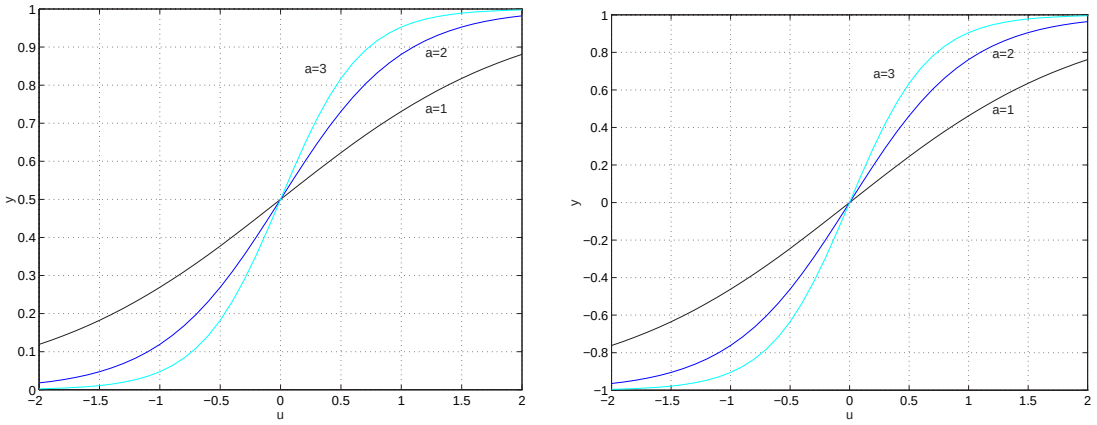
onde:
- l representa o número da camada atualmente sendo processada;
- m representa o total de neurônios presentes na camada l sendo processada (o valor de m exclui a entrada de *bias*, ou de polarização dos neurônios daquela camada);
- $y_i^{(l-1)}$ representa o sinal de saída do neurônio i pertencente à camada anterior $(l-1)$;
- $w_{ij}^{(l)}$ representa o peso sináptico entre o neurônio j da camada l atualmente sendo processada e o sinal que está vindo do neurônio i da camada anterior $(l-1)$. Quando $i = 0$, se está na camada de entrada da rede e neste caso, $y_0^{(l-1)}$ corresponde à entrada de *bias* e $w_{0j}^{(l)}$ corresponde então ao peso de *bias* aplicada ao neurônio j da camada l. A entrada de *bias* é fixada num valor como $y_0 = +1$ ou $y_0 = -1$ e permanece constante durante todo o uso da rede.

O sinal de saída do neurônio j da camada l é calculado então como:

$$y_j = \varphi_j\left(v_j\right) \tag{C.2}$$

onde: φ corresponde a função de ativação utilizada para a rede em questão que normalmente é um operador não-linear. Pode até ser empregado um operador linear mas neste caso a rede perde sua desejável capacidade de correlação não linear entre os dados da sua camada de entrada e de saída (Haykin, 1999; Sarle, 2002). Este operador não-linear pode inclusive ser modelado como uma função com saturação ($Sgn(\cdot)$) mas isto compromete o algoritmo de treinamento da rede que normalmente está baseado na derivada desta função (caso do algoritmo *back-propagation*). Por outro lado é desejado que esta **função seja limitada**, ou seja, que sua saída fique restrita a uma certa faixa de valores, como é o caso então das função sigmoidal ou tangente hiperbólica (que são diferenciáveis). A figura C.1 mostra algumas funções de ativação tipicamente empregadas em redes MLP. Outras funções não-lineares também poderiam ser empregadas, como as tangentes hiperbólicas ($tanh$) e as funções de base radial (típicas de redes RBF).

Continuando da equação C.2, se j corresponde à um neurônio da primeira camada in-



(a) $\varphi(u) = \frac{1}{1+exp(-a \cdot u)}$ (b) $\varphi(u) = \frac{1-exp(-a \cdot u)}{1+exp(-a \cdot u)}$

a representa o grau de inclinação da função (normalmente = 1)

Figura C.1: Funções de ativação típicas de redes MLP.

visível (isto é, $l = 1$), se faz com que:

$$y_j^{(0)} = x_j(n)$$

onde $x_j(n)$ corresponde ao j-ésimo elemento do vetor de entrada $\mathbf{x}(n)$. Quando o neurônio j pertence à camada de saída (isto é, $l = L$), então pode-se calcular o sinal de erro como sendo:

$$e_j = t_j - y_j \tag{C.3}$$

onde: j se refere a um dos neurônios da camada de saída rede;
t_j representa a saída desejada para o neurônio j, e;
y_j equivale à saída alcançada pela rede para o neurônio j.

Com base no sinal de erro e_j pode-se determinar o erro energético atual do neurônio j como sendo $\frac{1}{2}(e_j)^2$ – calculado para cada padrão que se apresente à rede. E pode-se ainda determinar o valor do erro energético total da rede somando o sinal $\frac{1}{2}(e_j)^2$ por sobre todos os neurônios da camada de saída, caracterizando o SE (*Sum of squared Errors*), como sendo:

$$SE = \frac{1}{2} \sum_j e_j^2 \tag{C.4}$$

Se N representa o total de padrões já apresentados à rede, pode-se determinar o a erro energético quadrático médio (ou mais simplesmente: MSE – *Mean of Squared Errors*) como

sendo:

$$MSE = \frac{1}{N}\sum_{n=1}^{N} SE \tag{C.5}$$

Para um certo conjunto de treinamento, o MSE representa a *função de custo* da rede, ou uma forma de se medir o desempenho do aprendizado da rede. Para o caso de treinamento *on-line* da rede, naturalmente é desejado que o valor de MSE diminua continuamente com o tempo, senão pode significar que a rede não esteja convergindo no seu treinamento. O objetivo do processo de treinamento da rede é ajustar os parâmetros livres da rede (pesos sinápticos e de *bias*) de forma a minimizar o MSE da rede. Para realizar esta minimização pode-se usar o método de mínimo erro quadrático (LMS) que aplica uma correção aos pesos sinápticos w_{ij} de maneira proporcional à derivada parcial de $^{\partial SE}/_{\partial w_{ij}}$. Esta derivada parcial, ou gradiente do erro, representa um fator de sensibilidade da rede, determinando a direção na qual é realizada a busca no espaço de modificação dos pesos sinápticos (Haykin, 1999). Esta derivada deve ser avaliada para cada um dos neurônios da rede (gradiente local) desde a camada de saída até a primeira cada oculta da rede, método conhecido pelo nome de "regra delta" (δ_j), que para o caso específico dos neurônios da camada de saída é calculado como:

$$\frac{\partial SE}{\partial w_{ij}} \Rightarrow \delta_j = -e_j \cdot \varphi'_j(v_j) \tag{C.6}$$

onde φ'_j equivale a derivada da função ativação aplicado sobre o valor do potencial de ativação v_j atingido pelo neurônio j da camada de saída da rede.

A correção que deve ser aplicada à cada peso sináptico, Δw_{ij}, é calculada então como sendo:

$$\Delta w_{ij} = -\eta \cdot \delta_j \cdot y_i \tag{C.7}$$

onde η representa a taxa de aprendizado da rede ou do algoritmo de retropopagação dos erros. O uso do sinal negativo na equação C.7 se refere ao *gradiente descendente* no espaço de pesos, ou seja, à busca pela direção de mudança no peso sináptico de forma que reduza o valor de SE da rede. Por fim, as equações C.6 e C.7 levam mais simplesmente à:

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot y_i \tag{C.8}$$

onde o *gradiente local*, δ_j, é definido como sendo:

$$\begin{aligned} \delta_j &= -\frac{\partial SE}{\partial w_{ij}} \\ &= e_j \cdot \varphi'_j(v_j) \end{aligned} \tag{C.9}$$

sendo que o gradiente local aponta na direção requerida para a ataulização dos pesos sinápticos.

As equações anteriores de C.6 à C.9 se aplicam facilmente quando j se refere a um neurônio localizado na camada de saída da rede ($l = L$). Mas os outros neurônios das camadas ocultas também são responsáveis por induzir erros cometidos pela camada de saída da rede. Para o caso destes neurônios, não existe uma "resposta desejada" e portanto se faz necessário determinar de maneira recursiva sinais de erro para todos estes neurônios, propagando estes sinais na direção da camada de saída para a de entrada. Neste caso, a equação de retropropagação de erros para cálculo dos gradientes locais, δ_j se modifica para ((Haykin, 1999):

$$\delta_j = \varphi'_j(v_j) \cdot \sum_k \delta_k \cdot w_{jk} \tag{C.10}$$

onde j agora se refere agora a um neurônio da camada oculta (δ_k se refere agora ao gradiente calculado anteriormente para o neurônio k da camada seguinte à atual).

De maneira resumida, o algoritmo *back-propagation* pode ser generalizado e ainda ser expandido com o termo *momentum*, α. A função do termo *momentum* é acelerar o decaimento do algoritmo de forma a ser seguida uma trajetória descendente para o erro (isto ocorre enquanto $0 \leq |\alpha| < 1$ – a série temporal $\Delta w_{ij}(t)$ converge). O termo *momentum* permite eliminar as oscilações da derivada parcial $\partial SE(t)/\partial w_{ij}(t)$ e evitar que o processo cíclico de aprendizado faça com que seja atingido um ponto de mínimo local (e não global) na superfície de erros da rede (SE) (Haykin, 1999). Desta forma, a regra delta, generalizada, desde o cálculo dos gradientes locais fica como (Haykin, 1999; Zell, 1995):

$$\delta_j^{(l)} = \begin{cases} \left(\varphi'_j(v_j^{(L)}) + c\right) \cdot e_j^{(L)} & \Rightarrow \text{Se a unidade } j \text{ é da camada de saída } (L). \\ \left(\varphi'_j(v_j^{(l)}) + c\right) \cdot \sum_k \delta_k^{(l+1)} \cdot w_{jk}^{(l+1)} & \Rightarrow \text{Se a unidade } j \text{ é de uma camada oculta } (l). \end{cases} \tag{C.11}$$

e

$$\Delta w_{ij}^{(l)}[t] = \eta \cdot \delta_j^{(l)} \cdot y_i^{(l-1)} + \alpha \cdot \Delta w_{ij}^{(l)}[t-1] \tag{C.12}$$

onde:
- l representa o número da camada atual (varia de 1 até L);
- i se refere ao i-ésimo neurônio de entrada da camada atual;
- j se refere ao j-ésimo neurônio de saída da camada atual;
- k se refere ao k-ésimo neurônio de saída da camada seguinte à atual $(l+1)$;
- t se refere ao instante do treinamento ($t-1$ = instante anterior de treino);
- c termo para eliminação do efeito de *flat spot* na derivada da função de ativação (opcional, varia de 0 à 0.25, tipicamente empregado 0.1);
- δ_j se refere ao gradiente local;
- φ trata-se da função de ativação;
- φ' se refere à derivada da função de ativação;
- e_j trata do erro do neurônio j da camada de saída;
- w_{ij} se refere ao peso da conexão (sináptico) do neurônio i para o neurônio j;
- η taxa de aprendizagem (varia de 0.1 à 1.0, usualmente empregado $\eta = 0.1$);
- α termo *momentum* (varia de 0 à 1, usualmente empregado $\alpha = 0.5$);
- e_j se refere ao erro de saída relativo ao neurônio j da camada de saída ($e_j = t_j - o_j$);
- t_j trata da saída desejada para o neurônio j;
- o_j trata da saída calculada pela rede para o neurônio j.

Fukuda et al. (1996), expandiu a equação acima C.12 com outros dois termos ainda:

$$\Delta w_{ij}^{(l)}[t] = \eta \cdot \delta_j^{(l)} \cdot y_i^{(l-1)} + \alpha \cdot \Delta w_{ij}^{(l)}[t-1] + \beta \cdot \Delta w_{ij}^{(l)}[t-2] + \gamma \cdot \Delta w_{ij}^{(l)}[t-3] \tag{C.13}$$

note que os termos β e γ simplesmente ponderam mais amostras passadas de variação dos pesos sinápticos.

Neste trabalho C.13 está implementada no módulo Neural.Mod mas não foi utilizado depois pois acrescenta mais 2 parâmetros extras em relação a taxa de aprendizado da rede (η) e termo *momentum* (α) que são sintonizados na prática. Valores muito elevados para a taxa de aprendizado fazem com a que a rede sature rapidamente seus pesos (erro de aprendizagem). Valores muito baixos fazem com a rede aprenda muito lentamente (na prática, mesmo para

valores de $\eta = 0.0001$, a rede ainda era capaz de aprender, mas o desempenho do controlador neural como um todo fica prejudicado devido ao baixo rendimento da rede). Assim também, valores maiores que 0.5 para a taxa *momentum* quase paralisam o capacidade de aprendizado da rede, tornando a rede insensível para variações rápidas do erro na sua entrada.

Referências Bibliográficas

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller, *Journal od Fynamic, Measurement, and Control (ASME)* **97-G**(3): 220–227.

Aleksander, I. and Morton, H. (1990). *An Introduction to Neural Computing*, Chapman & Hall, London, UK.

Amaral, S. d. (2000). *Controle a estrutura variável de robôs manipuladores interagindo com ambientes passivos*, Master's thesis, Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Florianópolis - SC.

An, C. H. and Hollerbach, J. M. (1989). The role of dynamic models in cartesian force control of manipulators, *Int. J. of Robotics Res.* **8**(4): 51–71.

Anderson, R. and Spong, M. W. (1988). Hybrid impedance control of robotc manipulators, *IEEE J. of Robotics and Automat.* **4**(5): 549–556.

Antsaklis, P. J. (1992). Neural networks in control systems, *IEEE Control Systems* **12**(2): 8–10.

Arimoto, S., Kawamura, S. and Miyazaki, F. (1984). Bettering operations of robots by learning, *Journal of Robotics Systems* **1**(1): 123–140.

Asada, H. and Slotine, J. J. E. (1986). *Robot Analysis and Control*, John Wiley and Sons, New York, USA.

Barrientos, A., Peñín, L. F., Balaguer, C. and Aracil, R. (1997). *Fundamentos de Robótica*, McGraw-Hill, Madrid.

Barto, A. G. (1989). Connectionist learning for control: An overview, *COINS technical report 89-89*, Dept. of Computer and Information Science, University of Masschusetts.
URL: *ftp://archive.cis.ohio-state.edu/pub/neuroprose/barto.control.ps.Z*

Battistela, S. (1999). *Controle de força e posição de robôs manipuladores utilizando redes neurais artificiais*, Master's thesis, Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Florianópolis - SC.

Bier, C. C. (2000). *Implementação de um algoritmo de controle de força em um manipulador scara*, Master's thesis, Pós-Graduação em Engenharia Mecânica - Universidade Federal de Santa Catarina, Florianópolis - SC.

Bier, C. C. and Guenther, R. (2000). Um controlador de força para seguimento de contornos planos, *CBA'2000 - XIII Congresso Brasileiro de Automática*. Submitted...

Bouchaffra, D. (2001). CSE 513 soft computing, *Powerpoint presentation*, Oakland University. 10 Dez 2002.
URL: *http://www.oakland.edu/ bouchaff/softcomputing/ch9[513]part2prt.ppt*

Burton, T. D. (1986). *Introduction to Dynamic Systems Analysis*, McGraw-Hill, New York, USA.

Butler, H. (1990). *Model Reference Adaptative Control*, ETH-Bibliothek (Swiss Federal Institute of Technology), Zuerich.

Cajueiro, D. O. and Hemerly, E. M. (2000). Direct adaptive control using feedforwared neural netowoks, *XIII Congresso Brasileiro de Automática (CBA 2000)*, Florianópolis, SC, pp. 79–84.

Carelli, R., Camacho, E. F. and Patiño, D. (1995). A neural network based feedforward adaptative controller for robots, *IEEE Transactions on Systems, Man and Cybernetics* **25**(9): 1281–1288.

Carelli, R., Kelly, R. and Ortega, R. (1990a). Adaptive force control of frobot manipulators, *International Journal of Control* **52**(1): 37–54.

Carelli, R., Kelly, R. and Ortega, R. (1990b). Adaptive force control of robot manipulators, *International Journal of Control* **52**(1): 37–54.

Chiaverini, S. and Sciavicco, L. (1993). The parallel approach to force/position control of robotic manipulators, *IEEE Transactions on Robotics and Automation* **9**(4): 361–373.

Clarke, D. W., Mohtadi, C. and Tuffs, P. S. (1987a). Generalized predictive control – part i. the basic algorithm, *Automatica* **23**(2): 137–148.

Clarke, D. W., Mohtadi, C. and Tuffs, P. S. (1987b). Generalized predictive control – part ii. extensions and interpretations, *Automatica* **23**(2): 149–160.

Colbaugh, R., Seraji, H. and Glass, K. (1993). Direct adaptive impedance control of robot manipulators, *Journal of Robotics Systems* **10**(2): 217–248.

Commuri, S. and Lewis, F. L. (1997). CMAC neural networks for control of nonlinear dynamical systems: Strucutre, stability and passivity, *Automatica* **44**(4): 635–641. Brief Paper.

Corbet, T., Sepehri, N. and Lawrence, P. D. (1996). Fuzzy control of a class of hydraulically actuated industrial robots, *IEEE Transactions on Control Systems Technology* **4**(4): 419–426.

Craig, J. J. (1984). Adaptive control of manipulators through repeated trails, *American Control Conference* pp. 1566–1573.

Crusius, C. A. R. (1996). *Formulação lmi para problemas de performance e robustez*, Master's thesis, Pós-Graduação em Engenharia Elétrica - Universidade Federal de Santa Catarina, Florianópolis - SC.

Cui, X. and Shin, K. G. (1996). Intelligent co-ordination of multiple systems with neural networks, *in* A. M. S. Zalzala and A. S. Morris (eds), *Neural Networks for Robotic Control – Theory and Applications*, Ellis Horwood, Great Britain, chapter 5, pp. 106–136.

da Mota Almeida, O., Passold, F. and da Silva Borges, P. S. (2000). Design issues and laboratory experiments in fuzzy control teaching, *COBENGE 2000 XXVII Congresso Brasileiro de Ensino de Engenharia*, Escola de Minas da UFOP (Universidade Federal de Ouro Preto), Ouro Preto, Minas Gerais.

de Leon Ferreira de Carvalho, A. C. P., de Pádua Braga, A. and LudermirSarle, T. B. (1998). *Fundamentos de Redes Neurais Artificiais*, 11^a Escola de Computação, 20-24 de Julho de 1998, Núcleo de Computação Eletrônica, COPPE Sistemas, Universidade Federal do Rio de Janeiro.

Demath, H. and Beale, M. (1999). Neural Network Toolbox For Use with MATLAB, User's Guide, Version 3.0, *Manual*, Mathworks Software.
URL: *http //www.mathworks.com/support/ftp/ftpindexv4.html*

den. Hartog, J. P. (1972). *Vibracoes nos sistemas mecanicos*, E. Blucher, Sao Paulo, chapter Cap. 7) Vibrações auto-excitadas, pp. 232–273.

Dégoulange, E. and Dauchez, P. (1994). External force control of an industrial PUMA 560 robot, *J. of Robotics Sys.* **11**(6): 523–540.

Eppinger, S. D. and Seering, W. P. (1987). Understanding bandwidth limitations in robot force control, *IEEE Conference on Robotics and Automation* pp. 904–909.

Er, M. J. and Liew, K. C. (1997). Control of adept one SCARA robot using neural networks, *IEEE Transactions on Industrial Electronics* **44**(6): 762–768.

Fahlman, S. E. (1988). An empirical study of learning speed, *Technical report, CMU-CS-88-162*, CMU.
URL: *http //www.cmu.edu/.../qp-tr.ps*

Ficher, W. D. and Mujtaba, S. (1992). Hybrid position/force control: A correct formulation, *International Journal of Robotics Research* **11**(4): 299–311.

Franklin, G. F., Powell, J. D. and Emami-Naeini, A. (1994). *Feedback Control of Dynamic Systems*, 3^{rd} edn, Addison-Wesley Publishing, New York, USA.

Freeman, J. A. and Skapura, D. M. (199?). *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, Massachusetts, USA.

Freund, E. and Pesara, J. (1998). High-bandwidht force and impedance control for industrial robots, *Robotica* **16**: 75–87.

Fritzke, B. (1997). Incremental neuro-fuzzy systems, *Applications of soft computing, SPIE International Symposium on Optical Science, Engineering and Insrumentation*, San Diego.
URL: *citeseer.nj.nec.com/fritzke97incremental.html*

Fukuda, T., Kurihara, T., Shibata, T., Tokita, M. and Mitsuoka, T. (1996). Neural servo controller for position, force and stabbing control of robotic manipulators, *in* A. M. S. Zal-

zala and A. S. Morris (eds), *Neural Networks for Robotic Control – Theory and Applications*, Ellis Horwood, Great Britain, chapter 3, pp. 64–79.

Gabrijel, I. and Dobnikar, A. (1997). Adaptative RBF neural network, *SOCO'97 Conference*, Nimes, France, pp. 164–170.
URL: *http://cherry.fer.uni-lj.si:80/∼gabriel/soco97/soco97.zip*

Gasalino, G., Davoli, F., Minciardi, R. and Zappa, G. (1987). On implicit modelling theory: Basic concepts and application to adaptative control, *Automatica* **23**(2): 189–201.

Ge, S. S., Hang, C. C. and Woon, L. C. (1997). Adapatative neural network control of robot manipulators in task space, *IEEE Transactions on Industrial Electronics* **44**(6): 746–752.

Girosi, F. and Poggio, T. (1993). Networks and the best approximation property, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, pp. 257–264.

Golin, J. F., Weihmann, L. and Guenther, R. (2000). *Manual do Usuario do Robo Inter - SCARA*, 2^a edn, Laboratório de Robótica, Universidade Federal de Santa Catarina.

Gorez, R. and Neyer, M. D. (1994). Fuzzy control of robotic manipulators and mechanical systems, *in* S. G. Tzafestas and A. N. Venetsanopoulos (eds), *Fuzzy Reasoning in Information, Decision and Control Systems*, Kluwer Academic Publishiers, Netherlands, pp. 451–491.

Gupta, M. M. (1994). Fuzzy logic and neural networks, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Controle Systems, Theory and Applications*, IEEE Press., article 6.3, pp. 403–416.

Gupta, M. M. and Rao, D. H. (1993a). Dynamic neural units with applications to the control of unknown nonlinear systems, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 5.6, pp. 352–371. Reprinted from Journal of Intelligent and Fuzzy Systems, 1(1), pp. 73-92, 1993.

Gupta, M. M. and Rao, D. H. (1993b). Neuro-control systems: A tutorial, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, pp. 1–43.

Gutiérrez, L. B., Lewis, F. L. and Lowe, J. A. (1998). Implementation of a neural network tracking controller for a single flexible link: Comparision with PD and PID controllers, *IEEE Transactions on Industrial Electronics* **45**(2): 307–318.

Haykin, S. (1999). *Neural Networks A Comprehensive Foundation*, 2^{nd} edn, Prentice Hall, New Jersey, USA.

Henriques, J., Victor, J., Pereira, C. and Dourado, A. (1998). Experimental on-line learning for a benchmark process, *3rd Portuguese Conference on Automatic Control*, Coimbra, Portugal.
URL: *http://control.dei.uc.pt/pdf/JH091998b.pdf*

Hornik, K., Stinchcombe, M. and White, H. (1993). Multilayer feedforward networks are universal approximators, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 4.3, pp. 265–270. Reprinted with permission from Neural Networks, vol 2, pp. 359-366, Pergamon Press, Oxford, England.

Hüpi, R. and Gruener, G. (2001). *Software Documentation for the Scara Robot Inter*, Institute of Robotics (IfR), Swiss Federal Institute of Technology (ETH), Zurich, Swiss.

Hunt, K. J. and Sbarbaro, D. (1993). Neural networks for internal model control, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 5.4, pp. 321–327. Reprinted with permission from IEE Proceedings-D, 138(5), pp. 431-438, September, 1991.

Hunt, K. J., Sbarbaro, D., Zbikowski, R. and Gawthrop, P. J. (1993). Neural networks for control systems – a survey, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 3.1, pp. 171–200. Reprinted with permission from Automatica, 28(6), pp. 1083-1112, 1992.

Jagannathan, S. (1996). Adaptive control of unknown feedback linearizable systems in discrete-time using neural networks, *IEEE International Conference on Robotics and Automation*, Vol. 1, Minneapolis, Minnesota, USA, pp. 258–263.

Jeon, D. and Tomizuka, M. (1993). Learning hybrid force position control of robot manipulators, *IEEE Transactions on Robotics and Automation* **9**(4): 423–431.

Jondarr, C. G. H. (1996). Back Propagation Family Album, *Technical report, C/TR96-05*, Dept. of Computing, Macquaire University, NSW 2109, England.

Katič, D. and Vukobratovič, M. (1996). Connectionist based robot control: an overview, 13^{th} *IFAC*, Vol. 1b-05 6, San Francisco, USA, pp. 169–174.

Keller, R. (1999). CSCI 152 neural networks course, *Lecture slides*, Harvey Mudd College, Computer Science Dept., Claremont, California, USA.
URL: *http://www.cs.hmc.edu/claremont/keller/152-slides/index.html*

Kiguchi, K. and Fukuda, T. (1996). Fuzzy neural friction compensation method of robot manipulation during position/force control, *IEEE International Conference on Robotics and Automation*, Vol. 1, Minneapolis, Minesota, USA, pp. 372–377.

Kiguchi, K. and Fukuda, T. (1997). Intelligent position/force controller for industrial robot manipulators – application of fuzzy neural networks, *IEEE Transactions on Industrial Electronics* **44**(6): 753–761.

Kim, Y. H. and Lewis, F. L. (1998). Reinforcement adaptative learning neural network based friction compensantion for high speed and precision, *IEEE Conference on Decision & Control*, number 4 in *WM17*, Tampa, Florida USA, pp. 1064–1069.

Kim, Y. H. and Lewis, F. L. (1999). Neural network output feedback control of robot manipulators, *IEEE Transaction on Robotics and Automation* **15**(2): 301–309.

Kim, Y. H. and Lewis, F. L. (2000). Reinforcement adaptative learning neural-net-based friction compensantion control for high speed and precision, *IEEE Transaction on Control Systems Technology* **8**(1): 118–126.

Knight, K. (1990). Connectionist, ideas and algorithms, *Communications of the ACM* **33**(11): 59–74.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, New Jersey, USA.

Kuo, R. J. (1997). A robotic die polishing system through fuzzy neural networks, *Computers in Industry* **32**: 273–280.

Kwan, C. M. (1998). Robuts adaptive force/motion control of constrained robots, *IEE Proc. - Control Theory Appl.* **143**(1): 104–109.

Kwok, D. P., Tan, P., Li, C. K. and Wang, P. (1990). Linguistic PID controllers, *11th IFAC*, Vol. 7, Tallin, Estonia, USSR, pp. 192–197.

Lawrence, J. (1992). *Introduction to Neural Networks and Expert Systems*, California Scientific Software, Nevada City, California, USA.

Lewis, F. L., Abdallah, C. T. and Dawson, D. M. (1993). *Control of Robot Manipulators*, McMillan, New York, USA.

Lewis, F. L., Liu, K. and Yesildirek, A. (1995). Neural net robot controller with guaranteed tracking performance, *IEEE Transactions on Neural Networks* **6**(3): 703–715.

Li, Y. (1997). Hybrid control approach to the peg-in-hole problem, *IEEE Robotics & Automation Magazine* **4**(2): 52–60.

Lightbody, G. and Irwin, G. W. (1997). Nonlinear control structures based on embedded neural systems models, *IEEE Transactions on Neural Networks* **8**(3): 553–567.

Lin, C.-J. and Lin, C.-T. (1997). An ART-based fuzzy adapatative learning control network, *IEEE Transactions on Fuzzy Systems* **5**(4): 477–496.

Lin, S.-T. and Huang, A.-K. (1998). Hierarchical fuzzy force control for industrial robots, *IEEE Transactions on Industrial Electronics* **45**(4): 646–653.

Lippman, R. P. (1987). An introduction to computing with neural nets, *IEEE ASSP Mag.* pp. 4–21.

Liu, M. H. (1995). Force-controlled fuzzy-logic-based robotic deburring, *Control Eng. Practice* **3**(2): 189–201.

Lucibello, P. (1993). A learning algorithm for hybrid force control of robot arms, *Proc. IEEE International Journal on Robotics Automation*, pp. 654–658.

Maren, A. J. (1990). Multilayer Feedforward Neural Networks I: Delta Rule Learning, *in* C. T. Harston and R. M. Pap (eds), *Handbook of Neural Computing Applications*, Academic Pres, San Diego, California, USA, cap. 7, pp. 85–105.

Mason, M. (1981). Compliance and force control for computer controlled manipulators, *IEEE Transactions on Sytems, Man and Cybernetics* **SMC-11**(6): 418–432.

McCarragher, B. J., Hovland, G., Sikka, P., Aigner, P. and Austin, D. (1997). Hybrid dynamic modelling and control of constrained manipulation systems, *IEEE Robotics & Automation Magazine* **4**(2): 27–44.

McClamroch, N. H. (1993). Force and impedance control, *in* M. W. Spong, F. L. Lewis and C. J. Abdallah (eds), *Robot Control: Dynamics, Motion Planning and Analysis*, IEEE Pres Marketing, Piscataway, New Jersey, USA, part 7, pp. 273–276.

McClamroch, N. H. and Wang, D. (1988). Feedback stabilization and tracking of constrained robots, *IEEE Transactions on Automatic Control* **33**(5): 419–426.

Morris, A. S. and Khemaissia, S. (1996). Artificial neural network based intelligent robot dynamic control, *in* A. M. S. Zalzala and A. S. Morris (eds), *Neural Networks for Robotic Control – Theory and Applications*, Ellis Horwood, Great Britain, chapter 2, pp. 26–63.

Murphy, S. H. and Wen, J. T. (1991). Stability analysis of position and force control for robotic arms, *IEEE Transactions on Automatic Control* **36**(3): 365–371.

Narendra, K. S. (1997). Neural networks for real-time control, *36^{th} IEEE Conference on Decision and Control - CDC'97*, San Diego, California, USA, pp. 1026–1031.

Narendra, K. S. and Mukhopadhyay, S. (1993). Intelligent control usign neural networks, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 1.5, pp. 90–97. Reprinted from IEEE Control Systems Mag., 12(2), pp. 11-18, April, 1992.

Narendra, K. S. and Parthasarathy, K. (1993). Identification and control of dynamical systems using neural networks, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, pp. 444–467. Reprinted from IEEE Transactions on Neural Networks, 1(1), pp. 4-27, March, 1990.

Nascimento Jr., C. L. and Yoneyama, T. (2000). *Inteligência Artificial em Controle e Automação*, Editora Edgard Blücher.

Nguyen, D. H. and Widrow, B. (1990). The truck backer-upper: An example of self-learning in neural network, *IEEE Contr. Sys. Mag.* **10**(3): 18–23.

Nise, N. S. (2000). *Control Systems Engineering*, 3^{th} edn, John Wiley & Sons, New York.

Noriega, J. R. and Wang, H. (1998). A direct adaptive neural-network control for unknown nonlinear systems and its application, *IEEE Transactions on Neural Networks* **9**(1): 27–34.

Orr, M. (1998). Optimizing the widths of radial basic functions, 5^{th} *Brazilian Sympsium on Neural Networks*, Belo Horizonte, Brasil.
URL: *http://www.anc.ed.ac.uk/∼mjo/papers/bsnn98.ps.gz*

Orr, M. J. L. (1996). Introduction to radial basis fucntion networks, *Technical report*, Centre for Cognitive Science, University of Edinburgh.
URL: *www.anc.ed.ac.uk/∼mjo/papers/intro.ps*

Orr, M. J. L. (1999). Recent advances in radial basis function networks, *Technical reports*, Institue for Adaptive and Neural Computation, Division of Informatics, University of Edinburgh.
URL: *www.anc.ed.ac.uk/∼mjo/papers/recad.ps*

Ozaki, T., Suzuki, T., Furuhashi, T., Okuma, S. and Uchikawa, Y. (1991). Trajectory control of robotic manipulators using neural networks, *IEEE Transactions on Industrial Electronics* **38**(3): 195–202.

Pandian, S. R. and Kawamura, S. (1996). Hybrid force/position control for robot manipulators based on a d-type learning law, *Robotica* **14**: 51–59.

Passino, K. M. (1995). Inteligent control for autonomous systems, *IEEE Spectrum* **32**(6): 55–62.

Qin, S. J. (1994). Auto-tuned fuzzy logic control, *American Control Conference*, Saltimore, Maryland, USA, pp. 2465–2469.

Raibert, M. H. and Craig, J. J. (1981). Hybrid position/force control of manipulators, *Journal of Dynamic Systems, Measurement, and Control - Transactions of the ASME* . **102**: 126–133.

Ramírez, A. R. G., De Pieri, E. R. and Guenther, R. (2000). O controle de força e posição à estrutura variável aplicado em um robô industrial, *CBA'2000 – XII Congressso Brasileiro de Automática*, Florianópolis.

Rao, D. H. and Gupta, M. M. (1993). Dynamical neural units and function approximation, *in* M. M. Gupta and D. H. Rao (eds), *Neuro-Control Systems, Theory and Applications*, IEEE Pres, Piscataway, New Jersey, USA, article 4.6, pp. 289–294. Reprinted from IEEE Conf. on Neural Networks, San Francisco, CA, PP. 743-748, March 28–April 1, 1993.

Riedmiller, M. (1994a). Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms, *Neural Networks* **5**.
URL: *http://i11www.ira.uka.de/∼riedml/riedml_csi94_ps.Z*

Riedmiller, M. (1994b). Rprop – description and implementation details, *Technical report*, Centre for Cognitive Science, University of Edinburgh.
URL: *http://i11www.ira.uka.de/∼riedml/rprop_details.ps.Z*

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *IEEE International Conference on Neural Networks*, San Francisco, California, USA.
URL: *http://i11www.ira.uka.de/∼riedml/riedml_icnn93_ps.Z*

Sarle, W. S. (2002). Neural network FAQ, *Posting*, Periodic posting to the Usenet newsgroup, comp.ai.neural-nets. 7 parts, montly 28th updated, 03 Dez 2002.
URL: *ftp:ftp.sas.com/pub/neural/FAQ.html*

Sciavicco, L. and Siciliano, B. (1996). *Modeling and Control of Robot Manipulators*, McGraw-Hill.

Seborg, D. E., Edgar, T. F. and Mellichamp, D. A. (1989). *Process Dynamics and Control*, John Wiley & Sons.

Seraji, H. (1987). Adaptive force and position control of manipulators, *Journal of Robotics Systems* **4**(4): 421–434.

Seraji, H. (1994). Adaptative admittance control: An approach to explicit force control in compliant motion, *IEEE International Conference on Robotics and Automation* pp. 2705–2712.

Shan, Z., man Kim, H. and Wang, F.-Y. (1996). Plant identification and performance optimization for neuro-fuzzy networks, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, Baijing, China, pp. 2607–2612.

Shin, J.-H. and Lee, J.-J. (1997). Robust adaptive control of underactuated robot manipulators in cartesian space, *IEEE International Conference on Intelligent Robots and Systems, IROS'97*, Vol. 1, Grenoble, France, pp. 491–497.

Siegwart, R. Y., Büchi, R. and Bühler, P. (1998). Mechatronics education at ETH zurich based on 'Hands on Experience', 6^{th} *UK Mechatronics Forum Internat. Conf. Mechatronics'98*, Skövde, Sweden.

Sinh, S. K. and Popa, D. O. (1995). An analysis of some fundamental problems in adaptive control of force and impedance behaviour: Theory and experiments, *IEEE Transactions on Robotics and Automation* **11**(6): 912–921.

Sio, K. C. and Lee, C. K. (1998). Stability of neural network controllers for a class of plants, *IJCNN'98 International Joint Conference on Neural Neteorks and WCCI'98 IEEE World Congress on Computational Intelligende*, Anchorage, Alaska, pp. 954–957.

Slotine, J. J. E. (1984). Sliding controller design for nonlinear systems, *International Journal of Control* **40**(2): 421–434.

Song, K.-T. and Chu, T.-S. (1998). Reinforcement learning and its application to force control of an industrial robot, *Control Engineering Practice* **6**: 37–44.

Song, Y. D. (1997). Neuro-adaptative control with application to robotic systems, *Journal of Robotic Systems* **14**(6): 433–447.

Soucek, J. (1989). *Neural and Concurrent Real-Time Systems (The Sixth Generation Computer)*, John Wiley & Sons, New York, USA.

Spong, M. W. and Vidyasagar, M. (1989). *Robot Dynamics and Control*, John Wiley & Sons, New York, USA.

Sun, F., Sun, Z. and Woo, P.-Y. (1998). Stable neural-networks-based adaptive control for sampled-data nonlinear systems, *IEEE Transactions on Neural Networks* **9**(5): 956–968.

Sundareshan, M. K. and Askew, C. (1997). Neural network-assisted variable structure scheme for control of a flexible manipulator arm, *Automatica* **33**(9): 1699–1710.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA.
URL: *http://www-anw.cs.umass.edu/∼rich/book/the-book.html*

Tang, Y., Tomizuka, M. and Guerrero, G. (1997). Robust control of rigid robots, *CDC'97 - 36th Conference on Decision and Control*, Vol. WM07-1, San Diego, California, USA, pp. 791–796.

Tveter, D. R. (1998). *Neural Network FAQ*, SAS Institute, North Carolina, MIT FAQ Collection, drt@chistianliving.net.
URL: *ftp //ftp.sas.com/pub/neural/FAQ.html*

Verma, B. (1997). Fast training of multilayer perceptrons, *IEEE Transactions on Nerual Networks* **8**(6): 1314–1320.

Vicent, D. A. (1999). Robots help manufacturing, *Report ?*, Robotics Industry Directory, USA. ?

Villani, L., De Wit, C. C. and Brogliato, B. (1996). An exponentially stable adaptive force/position control for robot manipulators, *IFAC'96 - 13th Trienial World Congress*, Vol. 1b-01 2, San Francisco, California, USA, pp. 7–12.

Volpe, R. A. (1990). *Real and Artificial Forces in the Control of Manipulators: Theory and Experiments*, Phd thesis, Department of Physics, The Robotics Institute, Carnegie Mellon University, Pitssburgh, PA.
URL: *citeseer.nj.nec.com/volpe90real.html*

Volpe, R. and Khosla, P. (1993). A theoretical and experimental investigation of explicit force control strategies for manipulators, *IEEE Transactions on Automatic Control* **38**(11): 1634–1650.
URL: *citeseer.nj.nec.com/volpe93theoretical.html*

Vukobratović, M. and Katić, D. (1996). Stabilizing position/force control or robots interacting with environment by learning connectionist structures, *Automatica* **32**(12): 1733–1739.

Vukobratovič, M. and Stojić, R. (1995). Historical perspective of hybrid control in robotics: Beginnings, evolutions, criticism and trends, *Mechanism and Machine Theory* **30**(4): 519–532.

Warwick, K. (1996). An overview of neural networks in control applications, *in* A. M. S. Zalzala and A. S. Morris (eds), *Neural Networks for Robotic Control – Theory and Applications*, Ellis Horwood, Great Britain, chapter 1, pp. 1–25.

Watanabe, K. (1996). Intelligent control for robotic and mechatronic systems, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 1, Beinjing, China, pp. 322–327.

Weihmann, L. (1999). *Descrição, instalação, programação e funcionamento de um robô manipulador do tipo scara*, Master's thesis, Pós-Graduação em Engenharia Mecânica - Universidade Federal de Santa Catarina, Florianópolis - SC.

Whitney, D. E. (1987). Historical perspective and state of the art in robot force control, *International Journal of Robotics Research* **6**(1): 3–14.

Widrow, B. and Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptrons, madaline, and backpropagation, *Proceedings of the IEEE* **78**(9): 1415–1442.

Widrow, B. and Plett, G. L. (1997). Nonlinear adaptive inverse control, *36^{th} IEEE Congress on Decision and Control - CDC'97*, Vol. WM15-2, San Diego, California, USA, pp. 1032–1037.

Xiao, D., Ghosh, B. K., Xi, N. and Tarn, T. J. (2000). Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment, *IEEE Transactions on Control Systems Technology* **8**(84): 635–645.

Yabuta, T. (1992). Nonlienar basic stability concept of the hybrid position/force control scheme for robot manipulators, *IEEE Transactions on Automatic Control* **8**(5): 663–670.

Yabuta, T., Kamada, T., Tusjimura, T. and Saksta, H. (1988). Force control of servomechanismo using adaptative control, *IEEE International Journal of Robotics and Automation* **4**(2): 223–228.

Yager, R. R. and Filev, D. P. (1994). *Essentials of Fuzzy Modeling and Control*, John Wiley & Sons.

Yilditrim, S. and Sukkar, M. F. (1996). Internal model control of a robot using neural netwworks, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, Beijing, China, pp. 3095–3100.

Zadeh, L. A. (1996). Fuzzy control: Issues, contentions and perspectives, 13^{th} *IFAC*, pp. 35–38.

Zalzala, A. M. S. (1996). Model-based adaptive neural structures for robotic control, *in* A. M. S. Zalzala and A. S. Morris (eds), *Neural Networks for Robotic Control – Theory and Applications*, Ellis Horwood, Great Britain, chapter 4, pp. 80–105.

Zell, A. e. a. (1995). SNNS: Stuttgart neural network simulator, *User manual, version 4.1*, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart. 12 Dez 2002.
URL: *http://www-ra.informatik.uni-tuebingen.de/SNNS/UserManual/node1.html*

Zeng, G. and Hemani, A. (1997). An overview of robot force control, *Robotica* **15**: 473–482.

Zhihong, M., Wu, H. R. and Palaniswami, M. (1998). An adaptive tracking controller using neural networks for a class of nonlinear systems, *IEEE Transactions on Neural Networks* **9**(5): 947–955.