SÍNTESE DE CONTROLADORES PARA O PROBLEMA DE BALANCEAMENTO DE CARGA EM CLUSTERS HETEROGÊNEOS

João Marcos Meirelles da Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA

Aprovada por:

_____
Prof. Eugenius Kaszkurewicz, D.Sc.

_____
Prof. Amit Bhaya, Ph.D

_____
Prof. Paulo Cesar Pellanda, Dr.

_____
Prof. Fernando Cesar Lizarralde, D.Sc.

_____
Prof. Eugene Francis Vinod Rebello, Ph.D

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 2008

SILVA, JOÃO MARCOS MEIRELLES DA

Síntese de Controladores para o Problema de Balanceamento de Carga em Clusters Heterogêneos [Rio de Janeiro] 2008

XV, 197p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2008)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. balanceamento de carga
2. clusters de processadores
3. inequações matriciais lineares
4. redes neurais com atraso
5. controle ótimo
6. controle de custo garantido

I. COPPE/UFRJ II. Título (série)

O sincronismo do universo desorienta-me a tal ponto que não posso acreditar que um relógio possa existir sem um relojoeiro.

Voltaire

Agradecimentos

Em primeiro lugar à Deus, por simplesmente ter me dado o bem maior que é a vida. Em segundo aos meus pais, Aluizio Ferreira da Silva (*in memorian*) e Silvina Meirelles da Silva, que sempre nunca mediram esforços para dar o sustento necessário à família e que ensinaram-me os grandes valores como a honestidade e o respeito aos outros. Aos meus irmãos e irmã, por terem, junto aos meus pais, criado um ambiente familiar extremamente harmonioso e cheio de amor, valores estes fundamentais na formação do bom caráter de um cidadão. À minha esposa Gabriela, pelas suas broncas chamando-me ao dever de concluir este trabalho e pelo seu infindável apoio emocional nos momentos de dúvida quanto concluir ou não. À Marinha do Brasil, por ceder várias das minhas horas de trabalho em prol desta tese e, principalmente, ao Capitão-De-Fragata Engenheiro Naval Nelson Luiz de Paula Menezes **MONNERAT**, por ter me confiado a missão de montar, operar e gerenciar o cluster de computadores do Instituto de Pesquisas da Marinha (IPqM). Aos meus superiores, Capitão-de-Fragata Engenheiro Naval Orlando Ribeiro **VITA** Júnior pela compreensão em relação aos meus horários e pelo apoio, e ao Capitão-de-Mar-e-Guerra Engenheiro Naval Luiz Alberto Lisboa da Silva **CARDOSO** pelo incentivo em relação à tese e pela concessão de inúmeras licenças para tratar deste trabalho. Ao amigo Dr. Leonardo Valente Ferreira pela revisão do material e pelas inúmeras dúvidas retiradas. Ao meu orientador, Prof. Dr. Eugenius Kaszkurewicz da COPPE/UFRJ que mostrou extrema paciência na sua orientação, preocupando-se junto comigo nas horas de maior necessidade e corrigindo a minha rota, mesmo quando eu insistia em não enxergá-la. Ao mestre, com carinho, meu muitíssimo obrigado e eterna gratidão. O que foi feito não tem preço.

Por fim, dedico esta tese ao meu filho, Miguel Antônio, por simplesmente existir e dar sentido a tudo isto.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

SÍNTESE DE CONTROLADORES PARA O PROBLEMA DE BALANCEAMENTO DE CARGA EM CLUSTERS HETEROGÊNEOS

João Marcos Meirelles da Silva

Fevereiro/2008

Orientador: Eugenius Kaszkurewicz

Programa: Engenharia Elétrica

O problema do balanceamento de carga em clusters de processadores heterogêneos é analisado utilizando-se teoria de redes neurais artificiais com atrasos, teoria de controle ótimo e *Inequações Matriciais Lineares* (LMI).

Partindo-se de um modelo matemático que inclui atrasos e processadores com velocidade de processamento diferentes, o modelo é transformado em um caso especial de uma rede neural conhecida como *Modelo de Hopfield-Tank com atrasos*.

Propondo uma nova função de energia para este caso especial de rede neural com atrasos, garante-se as condições de convergência com base no uso de LMIs.

Critérios de performance sujeitos às condições de estabilidade para a versão não-linear são analisados, e um novo método sistemático de síntese de controladores para balanceamento de carga é proposto, utilizando-se duas LMIs acopladas - uma garantindo convergência global e a outra garantindo performance em uma região linear de operação.

Simulações e experimentos comprovam a eficácia desta abordagem, reduzindo-se os tempos de balanceamento de carga com custo computacional viável para clusters com grande número de processadores.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

CONTROLLER SYNTHESIS FOR THE LOAD BALANCING PROBLEM ON HETEROGENEOUS CLUSTERS

João Marcos Meirelles da Silva

February/2008

Advisor: Eugenius Kaszkurewicz

Department: Electrical Engineering

The load balancing problem in clusters of heterogeneous processors is analyzed using delayed artificial neural networks theory, optimal control theory and *Linear Matrix Inequalities* (LMI).

Starting with a mathematical model that includes delays and processors with different processing velocity, this model is transformed into a special case of a neural network model know as *Delayed Hopfield-Tank Neural Networks* model.

A new energy function is proposed to this delayed neural network special case, assuring convergency conditions through the use of LMIs.

Some performance criteria subject to stability conditions to the non-linear model version are analyzed, and a new load balancing controller systematic method of synthesis is proposed, using two coupled LMIs - one guaranteeing global convergence and the other guaranteeing performance in a linear region of operation.

Simulations and experiments proves the efficiency of this approach, reducing load balancing time with a viable computacional cost for clusters with high number of processors.

Sumário

Lista de Figuras **ix**

Lista de Tabelas **xiii**

1 Introdução **1**
1.1 Apresentação do Problema de Balanceamento de Carga 3
1.2 Revisão da Literatura 4
1.3 Motivação e Objetivo do Trabalho 5
1.4 Estrutura da Tese 8

2 Modelos Matemáticos e Métodos de Solução Existentes **10**
2.1 Modelo Hidrodinâmico 11
2.2 Modelo de Chiasson *et al.* 13
2.3 Modelo de Hopfield-Tank de Redes Neurais 27
2.4 Analogia Entre Modelos 29
2.5 Resumo do Capítulo 32

3 Formulações do Modelo Sob a Forma de Redes Neurais **34**
3.1 Versões do Modelo sem Atraso (Linear e Não-Linear) 37
3.2 Versões do Modelo com Atraso (Linear e Não-Linear) 43
3.3 Outra Forma de Representação do Modelo de Chiasson *et al.* 47
3.4 Resumo do Capítulo 49

4 Resultados de Estabilidade Para os Diferentes Modelos **50**
4.1 Versões do Modelo sem Atraso (Linear e Não-Linear) 50
4.2 Versões do Modelo com Atraso (Linear e Não-Linear) 58
4.3 Resumo do Capítulo 67

5 O Problema de Controle Objetivando Desempenho **71**
5.1 Leis de Controle 73
5.2 Sistemática para Análise dos Resultados das Simulações 76
5.3 Síntese de Controladores 79
5.4 Metodologia Proposta para a Síntese 135
5.5 Resumo do Capítulo 136

6 Simulações e Experimentos em Clusters Homogêneos e Heterogêneos 138
6.1 Arquitetura de Implementação 139
6.2 Setup Experimental e Parâmetros do Modelo de Rede Neural 143
6.3 Experimentos em um Cluster Homogêneo 147
6.4 Experimentos em um Cluster Heterogêneo 154
6.5 Análise dos Resultados 159
6.6 Resumo do Capítulo 160

7 Conclusões e Trabalhos Futuros 161
7.1 Trabalhos Futuros 162

Apêndices 164

A Prova de Conservação do Número Total de Tarefas 164

B Algoritmos Genéticos 175
B.1 População Inicial 178
B.2 Mecanismo de Codificação 178
B.3 Função de Adequação 179
B.4 Mecanismo de Seleção 179
B.5 Mecanismo de Recombinação 181
B.6 Mecanismo de Mutação 183
B.7 Critério de Terminação 183
B.8 Descrição Genérica de um Algoritmo Genético 184

C Inequações Matriciais Lineares 186

D Fluxograma Simplificado do Algoritmo de Balanceamento de Carga 189

Referências 191

Lista de Figuras

1.1 Network Of Workstations. 1
1.2 Processadores Quad-Core Xeon equipando módulo de processamento. . . . 3
1.3 Cluster do IPqM com 11 nós Pentium 4HT 2.6 GHz com 11 Gb de RAM . 6
1.4 Metodologia a ser empregada 8

2.1 Esquema de balanceamento de carga do modelo hidrodinâmico 12
2.2 Curva característica da função não-linear $uhsat(y_i(t))$ empregada no modelo (2.1). 15
2.3 Curva característica da função não-linear $\mu(x_i(t))$ empregada no modelo (2.1). 16
2.4 Simulação do modelo (2.1) com atrasos ($\tau_{ij} = 200\mu s$ e $h_{ij} = 400\mu s$) 18
2.5 Simulação do modelo (2.1) sem atraso de comunicação ($\tau_{ij} = 0\mu s$ e $h_{ij} = 400\mu s$) 20
2.6 Simulação do modelo (2.1) sem atrasos ($\tau_{ij} = 0\mu s$ e $h_{ij} = 0\mu s$) 21
2.7 Diagrama de tempo do balanceamento de carga síncrono. 26
2.8 Curva característica da função de ativação $\phi_i(x_i(t))$ (2.31) 29
2.9 A analogia entre os modelos publicados por HUI *et al.* e CHIASSON *et al.*. 30
2.10 Relacionamento entre os modelos estudados 31

3.1 Desenvolvimento a partir do Modelo Base 35
3.2 Diagrama em blocos da versão linear na variável x sem atraso 40
3.3 Diagrama em blocos da versão linear na variável y sem atraso 41
3.4 Diagrama em blocos da versão não-linear na variável x sem atraso 42
3.5 Diagrama em blocos da versão não-linear na variável y sem atraso 42
3.6 Diagrama em blocos da versão linear na variável x com atraso 44
3.7 Diagrama em blocos da versão linear na variável y com atraso 45
3.8 Diagrama em blocos da versão não-linear na variável x com atraso 46
3.9 Diagrama em blocos do modelo de rede neural com atrasos do ponto de vista da Teoria de Controle 47

4.1 Determinação da matriz P para demonstrar a estabilidade segundo Lyapunov 69
4.2 $\dot{V}(y_1, y_2)$ sujeita à $y_1 = -y_2$ é a parabola formada pela intersecção do plano com a calha 70

5.1 Convergência do Controlador Proposto 74
5.2 Função $v_i(y_i(t))$ para diferentes valores de ganho 74
5.3 Funções não-lineares analisadas na simulação [09]. 78

5.4 Pulsos de entrada $\lambda_p = [7 \quad 5 \quad 3]^T$ e $t_\lambda = 100ms$ 78
5.5 Diagrama em blocos do modelo versão linear na variável y sem atraso . . . 79
5.6 Simulação [01]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[c]}$ . . 86
5.7 Simulação [02]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[02]}$ . . 87
5.8 Simulação [01]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[c]}$ (5.27). O gráfico (a) apresenta o sinal de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 88
5.9 Simulação [02]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[02]}$ (5.29). O gráfico (a) apresenta o sinal de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 89
5.10 Simulação [02]: Resultado da simulação para a matriz de ganhos $K_{[01]}$ com uma nova condição inicial $x(0) = [0,6 \quad 0,4 \quad 0,21]^T$ 90
5.11 Simulação [03]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[03]}$ (5.39). 94
5.12 Simulação [04]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[04]}$ (5.61).102
5.13 Simulação [04]: Comparação entre os sinais de erro das simulações $[01]$ e $[04]$. O gráfico (a) apresenta os sinais de erro com os ganhos dados por $K_{[c]}$ (5.27), e o gráfico (b) apresenta os sinais de erro com os ganhos dados por $K_{[04]}$ (5.61). 104
5.14 Simulação [04]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[c]}$ (5.27). O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 105
5.15 Simulação [04]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[04]}$ (5.61), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 106
5.16 Custo computacional da síntese de controladores pela Definição 5.3 em função do número de nós n de um cluster. 108
5.17 Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^1_{[05]}$ (5.65). 110
5.18 Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^2_{[05]}$ (5.66). 111
5.19 Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^1_{[05]}$ (5.66), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 112
5.20 Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[04]}$ (5.61), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7). 113
5.21 Simulação [06]: Resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz $K_{[c]}$ (5.27). 116
5.22 Simulação [06]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[04]}$ (5.61).116
5.23 Simulação [06]: O gráfico (a) mostra a resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz de ganhos $K_{[c]}$ (5.27). O gráfico (b) mostra a atividade dos controladores conforme definido em (5.7). 117
5.24 Simulação [06]: O gráfico (a) mostra a resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz de ganhos $K_{[04]}$ (5.61). O gráfico (b) mostra a atividade dos controladores conforme definido em (5.7). 118

5.25 Simulação [07]: Resultado da simulação para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 1,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$. 120
5.26 Simulação [07]: Exemplo de atividade das funções não-lineares $\phi(y_i(t))$ e $\phi(y_i(t-h))$ para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 1,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$. 122
5.27 Simulação [08]: Resultado da simulação para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 0,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$. 124
5.28 Simulação [08]: Exemplo de atividade das funções não-lineares $\phi(y_i(t))$ e $\phi(y_i(t-h))$ para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 0,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$. 125
5.29 Simulação [09]: Comparação de desempenho para as funções não-lineares de primeiro e primeiro-terceiro quadrantes (uhsat, sat, uhsign, sign) para a matriz de ganhos $K_{[09]}$ (5.72). 127
5.30 Simulação [09]: Comparação de desempenho para as funções não-lineares de primeiro e primeiro-terceiro quadrante (uhtanh e tanh) para a matriz de ganhos $K_{[09]}$ (5.72). 128
5.31 Simulação [09]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[09]}$ (5.72) e função $sign(y(t))$. 129
5.32 Simulação [10]: Resposta do modelo (2.1) a um atraso significativo em relação ao tempo da resposta transitória. O gráfico (f) destaca a resposta periódica onde o *burst* acontece. 131
5.33 Simulação [11]: O gráfico (a) apresenta a resposta do modelo (2.1) para as matrizes de ganhos $K^1_{[11]}$, $K^2_{[11]}$ e $K^3_{[11]}$. O gráfico (b) destaca a resposta do nó 2, mostrando que os elementos fora da diagonal principal da matriz de ganhos contribui para a redução do efeito de bursting criado pelo atraso. . . 134

6.1 Arquitetura de Implementação 139
6.2 Atraso de transferência h em função do número de bytes a serem transmitidos. 145
6.3 Experiência [01]: O gráfico (a) apresenta o tempo de espera estimado para $K_z = 0,9$ e o gráfico (b) apresenta a respectiva simulação com h médio de $532\mu s$ e Δ_t médio de 0,35ms. 150
6.4 Experiência [01]: Ampliação de parte da Figura 6.3 detalhando a troca simulada de tarefas entre os nós 1 e 2 para $K_z = 0,9$. 151
6.5 Experiência [01]: Resposta do modelo (2.1) - Tempo de espera estimado para $K_z = 1,0$. 151
6.6 Experiência [02]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação para $K_{[04]}$ com h médio de $481\mu s$. 153
6.7 Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $K^1_{[03]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,82ms. 156

6.8 Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,28ms. 157
6.9 Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,28ms. 158

A.1 Diagrama do balanceamento de carga entre dois nós com atraso 166

D.1 Fluxograma simplificado do algoritmo de balanceamento de carga 189

Lista de Tabelas

5.1 Parâmetros para a simulação [01] 84
5.2 Parâmetros do Algoritmo Genético para a simulação [02] 86
5.3 Parâmetros de desempenho entre as simulações $[01]$ e $[02]$ 87
5.4 Custo computacional para a solução do problema dado pela Definição 5.1. . 91
5.5 Parâmetros para a simulação [03] 92
5.6 Custo computacional para a solução do problema dado pela Definição 5.2. . 95
5.7 Parâmetros para a simulação [04] 101
5.8 Parâmetros de desempenho entre as simulações $[01]$ e $[04]$ 103
5.9 Custo computacional para a solução do problema dado pela Definição 5.3. . 107
5.10 Parâmetros para a simulação [05] 109
5.11 Parâmetros de desempenho entre duas matrizes de ganhos $K^1_{[05]}$ (5.65) e $K^2_{[05]}$ (5.66) para a simulação [05]. 110
5.12 Parâmetros para a simulação [06] 114
5.13 Parâmetros para a simulação [07] 119
5.14 Parâmetros para a simulação [08] 123
5.15 Desempenho do nó 2 do modelo (2.1) para as três matrizes de ganho apresentadas: $K^1_{[11]}$, $K^2_{[11]}$ e $K^3_{[11]}$, mostrando o efeito dos valores fora da diagonal principal sobre o bursting e sobre o settling time. 133
5.16 Tabela comparativa para as três diferentes metodologias de síntese de controladores. Δt_{sol} é o tempo necessário utilizado para encontrar uma solução, e os respectivos settling time encontrados nas simulações [02],[03,[04]. . . 137

6.1 Atraso de transferência h em função do número de bytes transferidos 144
6.2 Definição de normas de uma matriz quadrada Π qualquer. 146
6.3 Parâmetros de desempenho para o experimento [01]. 148

Lista de Símbolos

A - Matriz de estados de um sistema linear invariante no tempo.

A_{mf} - Matriz A em malha fechada.

h - Atraso de transferência de tarefas.

I - Matriz identidade.

K_d - Matriz de ganhos diagonal.

K_z - Ganho escalar discreto.

$K^j_{[i]}$ - A j-ésima matriz de ganhos da i-ésima simulação.

$\overline{K}_{[i]}$ - Matriz de ganhos normalizada da i-ésima experiência.

M_p - *Maximum peak overshoot*.

M_t - *Maximum overshoot instant*.

n - Número de nós que compõem um cluster.

p_{ij} - Fração da carga a ser transferida do j-ésimo nó para o i-ésimo nó.

q_i - Quantidade de tarefas na i-ésima fila.

$r_i(t)$ - Número de tarefas acima ou abaixo da média local estimada do tamanho das filas pelo i-ésimo nó.

t_{p_i} - Tempo de processamento utilizado pelo i-ésimo processador para uma determinada tarefa.

t_s - *Settling time* ou tempo de assentamento.

T - Matriz de conversão de tempo de espera entre os nós.

W - Matriz de realimentação no modelo de redes neurais com atraso.

W_h - Matriz de realimentação atrasada no modelo de redes neurais com atraso.

$x_i(t)$ - Tempo de espera estimado por uma tarefa para que seu processamento seja iniciado.

y_{thr}	-	Valor de *threshold* para a variável $y(t)$.
y_{max}	-	Valor de saturação da função $uhsat(y(t))$ que está relacionado à capacidade do canal de comunicação da rede.
Z_1, Z_2, Z_3	-	Variáveis matriciais para o problema de síntese de controladores via LMI.
Δ_{t_i}	-	Intervalo de tempo para a execução de uma iteração do algoritmo de balanceamento de carga.
$\Delta_{t_{sol}}$	-	Intervalo de tempo necessário para encontrar uma solução numérica.
ε	-	Constante positiva suficientemente pequena.
λ_i	-	Taxa de geração de tarefas para o i-ésimo nó.
λ_p	-	Taxa de geração de tarefas para o i-ésimo nó na forma de um pulso não-unitário.
$\mu(x_i(t))$	-	Taxa de consumo de tarefas para o i-ésimo nó.
$\phi(\cdot)$	-	Função não linear.
τ	-	Atraso de comunicação.

Capítulo 1

Introdução

O rápido progresso da tecnologia VLSI e das tecnologias de computação em rede vêm tornando o campo do processamento paralelo e distribuído economicamente atrativo para diversas aplicações que demandam uma capacidade de processamento muito alta, tais como: Aerodinâmica, meio-ambiente, meteorologia, astronomia, química, segurança (quebra de chaves criptográficas), projeto de fármacos, indústria de armamentos e processamento de sinais de alta velocidade em tempo real.

Os clusters são um conjunto de máquinas interligadas que executam uma ou várias tarefas para obtenção de um resultado [1] - Figura 1.1.

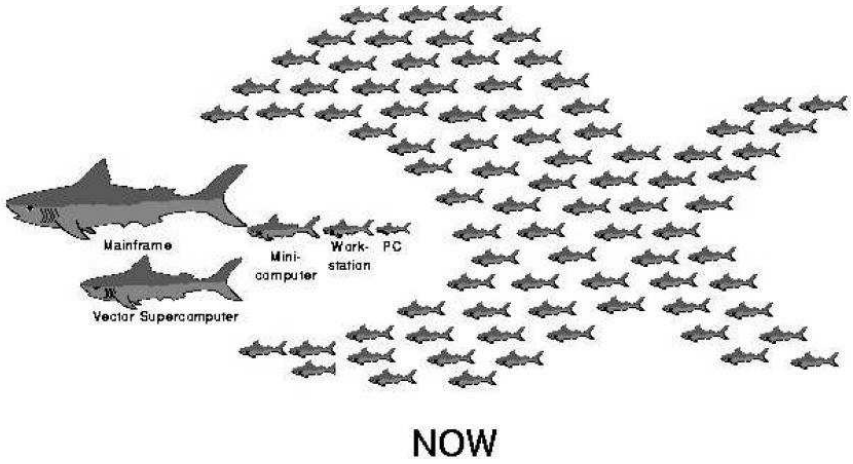

Figura 1.1: Network Of Workstations.

A idéia de juntar diferentes máquinas e dividir tarefas entre elas, para melhor aproveitar a capacidade de processamento individual resultando em uma capacidade maior do que a capacidade de um mainframe ou minicomputador, não é nova (1958). Mas faltava tecnologia

naquela época[1] - Figura 1.1.

Foi apenas na década de 80 que apareceram as primeiras máquinas paralelas, dando origem à Computação Paralela.

Em 1994, a NASA (National Aeronautics and Space Administration) criou um projeto chamado **Beowulf** cuja finalidade foi a de juntar diversas máquinas com um desempenho total de 1GFlop/s a um custo baixo ($\sim$ US$ 50.000,00) para o processamento de grandes quantidades de dados oriundas de satélites. Nesta época, as primeiras distribuições LINUX já estavam disponíveis, bem como a popularização de redes e a capacidade de processamento do processsador INTEL 80486 DX2 66MHz.

A maior desvantagem dos clusters, no entanto, é que o seu desempenho depende muito do tipo de aplicação a ser executada, pois o maior gargalo encontra-se justamente na rede de comunicação. O excesso de comunicação entre os nós e um balanceamento de carga deficiente podem atrasar em muito o processamento das tarefas.

Outro fator que pode degradar o desempenho de um cluster é o desbalanceamento de carga entre os nós, pois alguma tarefas podem desdobrar-se em outras tarefas, significando acúmulo destas nas filas, principalmente em clusters heterogêneos.

Os desafios principais para os clusters hoje podem ser listados como

1. Aplicações e Algoritmos: Busca-se criar novos algoritmos que sejam assíncronos, porém estáveis, e que dependam pouco de comunicação entre os nós;

2. Administração e Gerenciamento: Busca-se tratar de forma mais eficientes os recursos disponíveis como memória, espaço em disco, manutenção, segurança, balanceamento de carga, etc...

3. Redes e Protocolos: Busca-se o desenvolvimento de redes de alta velocidade e protocolos com o mínimo de *overhead*, a fim de aliviar o gargalo que representa a rede.

Os clusters são, de uma forma em geral, um compromisso entre custo e benefício.

No entanto, a evolução da necessidade de processamento em conjunto para prover maior poder computacional, como é o caso dos clusters, tem levado os conceitos de um cluster de processadores para dentro dos próprios microprocessadores.

[1] Figura extraída do site The Berkeley NOW Project. http://now.cs.berkeley.edu

Processadores com dois ou mais núcleos são comuns em uma única pastilha VLSI, como o $Quad - Core\ Intel^{®}\ Xeon^{®}\ series\ 5300$ - Figura 1.2[2].

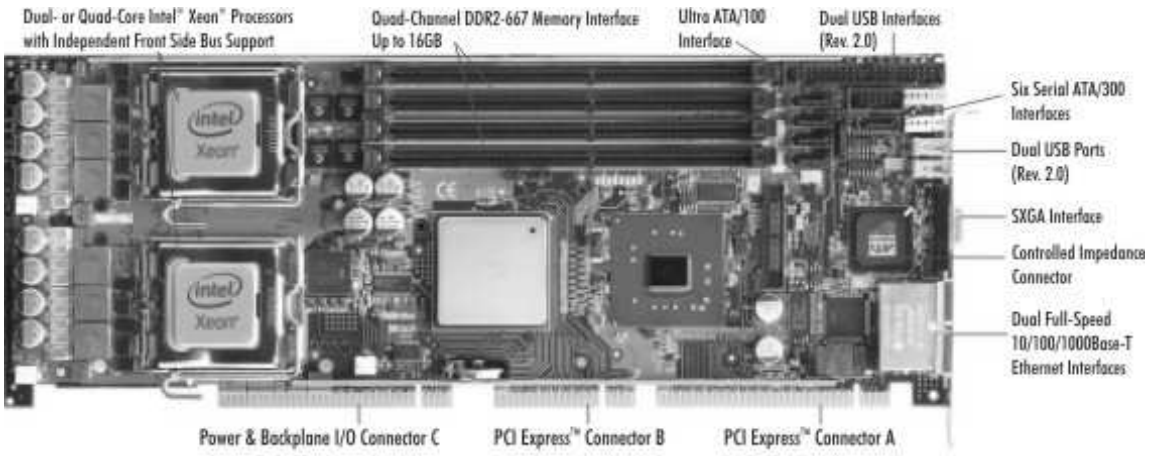

Figura 1.2: Processadores Quad-Core Xeon equipando módulo de processamento.

Tais processadores demandam novas ferramentas de programação paralela e/ou distribuída, gerenciamento do fluxo de informações, balanceamento de carga, entre outras, para permitir o uso eficiente do hardware.

1.1 Apresentação do Problema de Balanceamento de Carga

Os sistemas em clusters constituem-se em um conjunto de computadores autônomos conectados por uma rede (homogênea ou heterogênea). As principais vantagens destes sistemas são o alto desempenho, disponibilidade e extensibilidade a um custo reduzido. Para melhorar o desempenho destes sistemas, é essencial manter a carga de trabalho equilibrada entre todos os processadores [2].

O objetivo maior do balanceamento de carga em um sistema distribuído, é alocar tarefas entre os processadores para maximizar a utilização destes, e minimizar a média do tempo de resposta.

Existem várias taxonomias para o problema de balanceamento de carga [3]. Métodos diretos examinam a distribuição global de carga entre os processadores e atribui porções de carga para os processadores antes do processo começar. Métodos iterativos examinam o progresso computacional a utilização esperada dos recursos, e ajusta as atribuições de carga periodicamente conforme o processamento vai ocorrendo.

[2]Figura extraída do site http://www.signalogic.com

A atribuição de carga pode ser determinística, como no método de difusão [4] e métodos de gradiente. Uma comparação entre os diversos métodos determinísticos pode ser encontrado em [5].

Sendo conhecido como um problema do tipo **NP-HARD**, os problemas de balanceamento de carga são normalmente resolvidos utilizando-se heurísticas para obter-se uma solução ótima, ou quase-ótima, porque problemas encontrados em aplicações práticas não podem ser resolvidos, em muitos casos, de forma ótima apenas com modelos formais [6]. Além do mais, muitas soluções estudadas sobre este problema, eram dependentes da topologia do sistema empregado: Redes em anel, torus, hipercubo, etc...

1.2 Revisão da Literatura

Através dos anos, um número de métodos têm sido aplicados para resolver o problema de balanceamento de carga utilizando técnicas de *Branch-and-Bound* [7], teoria dos grafos [8], programação matemática [9] ou enumeração completa [10].

De acordo com SEREDYNSKI [11], a maioria destes métodos é capaz de encontrar uma solução ótima, mas somente em diversos casos restritos. Para lidarmos com o caso mais geral possível, um número de heurísticas foi introduzido. Eles não garantem uma solução ótima, mas oferecem uma solução quase ótima na grande maioria dos casos.

Nos últimos anos, alguns métodos foram propostos baseados em técnicas de *Simulated Annealing* [12], Algoritmos Genéticos [13–15], Agentes Inteligentes [16–19], Ant Colony [20], Teoria de Jogos [21–23] e redes neurais [24, 25].

Ao longo da década de 90 e no início do século XXI, iniciaram-se as abordagens ao problema de balanceamento de carga e agendamento via redes neurais artificiais como em [24, 26–29].

Com a interligação de clusters através da Internet (*grids*), ou seja, separados por distâncias geográficas curtas até muito grandes, o problema do balanceamento de carga têm-se tornado mais complexo [30], pois novas características do problema que antes procurava-se evitar, agora são exacerbadas. Dentre algumas das características temos: falhas na disponibilidade de processadores [17], capacidade de processamento altamente variável, custo financeiro de comunicação [31] e atrasos nos links [32].

Dentre as características citadas acima, o estudo do atraso (que normalmente são aleató-

rios) tem sido um dos mais importantes. Muitas vezes, o tempo necessário para migrar-se uma tarefa para um processador distante é maior do que o tempo necessário para esta mesma tarefa ser executada localmente. Os novos algoritmos devem tomar decisões sobre o custo da comunicação em ambientes diversos.

Em 2004, CHIASSON *et al.* [33] apresentaram os resultados de uma avaliação experimental enfatizando os atrasos de comunicação entre os processadores em uma rede local e em uma WAN. Seus resultados mostram os efeitos do atraso e suas variâncias sobre o tempo de processamento total.

A direção que as pesquisas recentemente publicadas apontam é a do uso cada vez maior de técnicas e heurísticas distribuídas e inteligentes. Apesar da natureza paralela das redes neurais, o que facilita a sua implementação, muito pouco tem sido encontrado na literatura sobre o uso delas associadas ao problema de balanceamento de carga [24] e ao problema de agendamento [28, 29, 34].

O estudo de redes neurais com atraso aplicadas ao problema de balanceamento de carga está apenas no início.

1.3 Motivação e Objetivo do Trabalho

Há apenas alguns anos, a maior fabricante mundial de microprocessadores da atualidade, a Intel Corporation, desistiu de explorar o aumento progressivo do clock em seus chips devido à barreira física existente para a integração cada vez maior de componentes [35]. Sua meta tornou-se explorar novas arquiteturas e a inclusão de vários processadores no mesmo sistema, conforme a estratégia da sua maior concorrente - a AMD Corporation [36].

Tal estratégia mostra claramente a tendência dos sistemas futuros em utilizarem processamento paralelo e distribuído como uma das formas mais baratas de aumentar o poder de processamento de um sistema computacional.

Um outro agente motivador é que muitas organizações privadas e governamentais possuem hoje uma rede local (LAN) conectando centenas e até milhares de microcomputadores pessoais tais como PCs e workstations - Figura 1.3 . O poder computacional em uma rede local pode, facilmente, exceder o poder de um supercomputador. Por outro lado, a maioria dos computadores são subutilizados a maior parte do tempo, mesmo durante períodos de pico de utilização. Então, a distribuição de tarefas entre os processadores em uma rede

torna-se bastante interessante a fim de melhorar o desempenho do parque computacional como um todo. Com a disponibilidade de redes de alta velocidade tais como: Fast Ethernet, Gigabit Ethernet, ATM e FDDI, as quais reduziram substancialmente o custo da comunicação entre processadores, o balanceamento de carga pode aumentar imensamente o fluxo e o processamento de informações sem nenhum hardware adicional. O hardware existente é simplesmente utilizado de uma forma mais efetiva.

Figura 1.3: Cluster do IPqM com 11 nós Pentium 4HT 2.6 GHz com 11 Gb de RAM

Devido a este ganho potencial, o problema de balanceamento de carga vem sendo estudado intensivamente nestes últimos anos [19, 22, 25, 37–41].

Dentre os inúmeros trabalhos já publicados, o foco desta tese foi direcionado para aqueles que tratam dos efeitos do atraso, sempre presente na comunicação entre os nós, e para aqueles que tratam de processadores com capacidade diferentes entre si.

Nesta linha, os trabalhos de HUI *et al.* [40, 42, 43], e os publicados por CHIASSON *et al.* [30, 32, 33] foram os que mais se destacaram pela sua clareza, e por motivos de conterem dados experimentais.

Os trabalhos publicados por HUI *et al.* fazem uma analogia do problema de balanceamento de carga entre processadores heterogêneos com o problema de equilíbrio hidroestático entre cilindros de líquido (conhecido como problema dos *vasos comunicantes*), onde cada processador é representado por um cilindro líquido, onde o diâmetro do cilindro é diretamente proporcional à capacidade de processamento de um nó, e a altura do cilindro à quantidade de trabalho a ser realizada (carga do processador). Os autores demonstram que

o equilíbrio hidroestático entre os cilindros corresponde à uma distribuição ótima de carga, com energia mínima.

No entanto, HUI *et al.* não tratam dos efeitos de atraso por uma abordagem direta, assumindo que a troca de tarefas é instantânea. Além disso, eles não modelaram o problema matematicamente, deixando apenas registrado um pseudo-algoritmo sobre como o balanceamento de carga deveria ser tratado tendo em mente a troca de líquidos (tarefas) entre os vasos (processadores).

Em [44], o autor baseia-se nos trabalhos de HUI *et al.* para criar um modelo matemático e determinístico, levando ainda em consideração o efeito dos atrasos de transferência (troca de tarefas) e de comunicação (troca de informações de estado entre os nós), dando uma grande contribuição para o trabalho de HUI *et al.*.

Em seus trabalhos publicados, CHIASSON *et al.* apresentam um controlador não-linear para promover a troca de tarefas entre os nós. O controlador, presente em cada um dos nós, baseia-se no estado completo do sistema (na quantidade de tarefas de seu próprio nó mais a informação de quantidade de tarefas presentes nos outros nós). Desta forma, o controlador do i-ésimo nó estima o quanto de tarefas ele possui acima ou abaixo da média da quantidade de tarefas em todos os nós, gerando uma informação que servirá de variável de decisão para a sua ação: doar ou receber tarefas e em que quantidade.

Porém, CHIASSON *et al.* não apresenta uma metodologia de síntese de controladores, do ponto de vista da teoria de controle, para obter os ganhos destes controladores. A determinação dos ganhos é baseado nos limites de banda passante da rede de comunicação entre os nós e através de avaliações experimentais, onde ele decide quais serão os ganhos de forma subjetiva.

Nesta tese é proposta uma forma de síntese de controladores para o modelo de (2.1) baseada na teoria de convergência de redes neurais com atraso e LMI (Inequações Matriciais Lineares), buscando diminuir o tempo ocioso dos processadores e contribuir também com o desenvolvimento deste modelo enquadrando-o sob o ponto de vista da teoria de controle - Figura 1.4.

O objetivo principal desta tese deverá ser alcançado primeiro através de um re-arranjo do modelo (2.1) e mostrando uma certa similaridade com o modelo de redes neurais de Hopfield-Tank com atraso, porém não exatamente igual. Uma vez que esta similaridade tenha sido evidenciada, trabalhos recentemente publicados sobre redes neurais com atraso e

LMI servirão de base para o estudo da estabilidade, convergência e síntese para os controladores não-lineares.

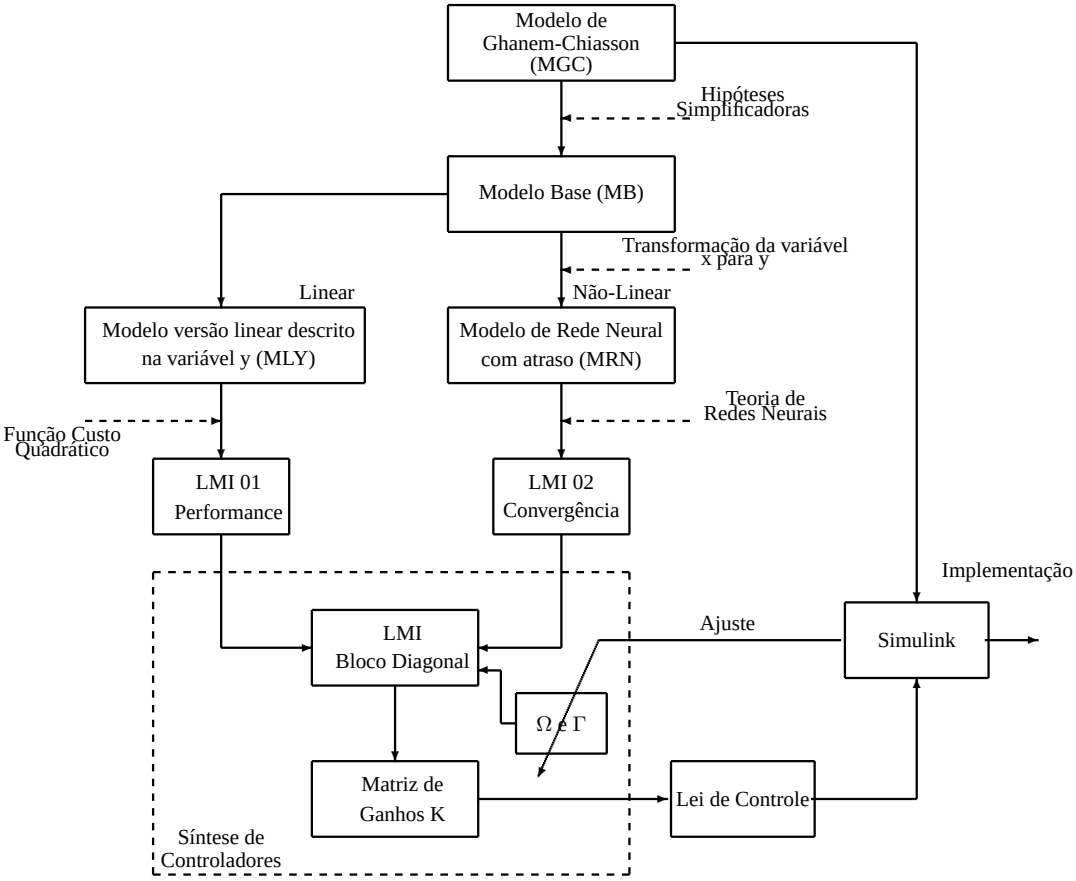

Figura 1.4: Metodologia a ser empregada

Simulações e experimentos em um cluster real mostram a validade desta abordagem.

1.4 Estrutura da Tese

O capítulo 2 aborda alguns modelos que se destacaram dentro da literatura em relação ao que se pretende nesta tese. Os modelos são explanados, ressaltando-se as similaridades e diferenças entre eles.

O capítulo 3 aborda as representações, ou aproximações, do problema de balanceamento de carga através de redes neurais, estudando desde o modelo mais simples linearizado até o modelo de interesse deste trabalho. A ordem de desenvolvimento destes modelos também é explicada e justificada neste capítulo.

No capítulo 4, a estabilidade para os modelos desenvolvidos no capítulo anterior é abordada, seguindo a mesma seqüência proposta.

No capítulo 5 abordamos a síntese de controladores para um dos modelos apresentados no capítulo 4, enfatizando a diferença entre os controladores do modelo utilizado como base e dos controladores propostos nesta tese, bem como diversas simulações.

No capítulo 6 simulações e experimentos baseados nos parâmetros medidos do *cluster* do IPqM são apresentados para validar a metodologia apresentada nesta tese.

E, finalmente, no capítulo 7 são feitas as conclusões sobre o trabalho e o problema de balanceamento de carga, bem como sobre o efeito do atraso. Ao final do capítulo, sugestões de desenvolvimento acerca do tema são deixadas para pesquisas futuras.

Capítulo 2

Modelos Matemáticos e Métodos de Solução Existentes

Apesar de diversos modelos diferentes terem sido publicados nos últimos anos, não existe ainda um modelo único e bem estabelecido de balanceamento de carga entre microprocessadores que sirva de referência para se estabelecer mecanismos de solução. Em vez disso, a grande maioria dos autores trabalham propondo seus próprios modelos, muitos baseados em algum outro modelo já publicado. Tais modelos normalmente focam em uma ou duas características do problema devido à grande complexidade de atacar-se todas as características simultaneamente.

Em relação às diversas técnicas para realizar o balanceamento de carga, destacou-se um trabalho de 1993 bastante citado até hoje que mostra a comparação entre alguns métodos [5].

Dentre os diversos modelos analisados na literatura disponível, buscou-se uma concentração apenas naqueles que mais se aproximavam das necessidades de um cluster atualmente, como processadores heterogêneos e atraso nos links de comunicação.

Nesta busca, dois modelos destacaram-se: o modelo publicado por HUI *et al.* [40] e o modelo publicado por CHIASSON *et al.* [33].

O primeiro modelo é interessante pois trata-se de uma analogia entre o problema de distribuição de carga entre processadores e o problema dos vasos comunicantes, onde características de um cilindro de líquido representam algumas características de um processador, tais como quantidade de carga sendo processada e capacidade de processamento. O objetivo é redistribuir o volume de líquido entre todos os cilindros, de forma que a altura da coluna

de líquido seja a mesma em todos eles. Tal objetivo leva à prova de que o sistema atinge um nível de energia mínimo, o que, por sua vez, corresponde a um nível ótimo de balanceamento de tarefas. No entanto, este modelo não aborda explicitamente o problema de atraso e trata a comunicação e a transferência de carga entre os nós como sendo instantâneas.

O segundo modelo inclui os chamados *atraso de comunicação* e *atraso de transferência de tarefas* entre os processadores, investigando seus efeitos sobre o custo e o tempo de balanceamento e propondo um controlador. Entende-se como custo alto a excessiva troca de tarefas entre os nós, resultando em respostas oscilatórias nas filas.

O atraso de comunicação, definido como τ_{ij}, é o tempo transcorrido para um nó j enviar uma informação, geralmente com poucos bytes (tamanho da fila, status do nó, sincronismo), para o nó i.

O atraso de transferência de tarefa, ou simplesmente atraso de transferência, definido como h_{ij}, é o tempo transcorrido para um nó j transferir uma determinada tarefa para o nó i. O atraso de transferência é, normalmente, muito maior que o atraso de comunicação.

Ambos os modelos também tratam de processadores heterogêneos, o que é interessante para aplicações em clusters que tenham sido expandidos após algum tempo de uso, situação esta onde normalmente os nós adicionais possuem capacidade de processamento maior que os anteriores.

Ao final do capítulo, serão mostradas algumas similaridades entre estes dois modelos e o modelo de rede neural do tipo Hopfield-Tank com atraso. A partir daí, desenvolveremos a idéia de que é possível utilizarmos o que foi publicado na área de redes neurais com atraso a fim de buscarmos uma classe de controladores que melhorem o desempenho.

2.1 Modelo Hidrodinâmico

Nos trabalhos apresentados em [40, 42, 43], cada processador é visto como um cilindro líquido onde a área transversa corresponde à capacidade do processador, os links de comunicação são modelados como canais de líquido entre os cilindros e o algoritmo de balanceamento gerencia o fluxo deste líquido. O objetivo é alcançar o estado onde a altura das colunas de líquido sejam a mesma em todos os cilindros. O sistema de computação é modelado como um grafo sem direção $G = (N, E)$, onde N é o conjunto de processadores e E representa a topologia da rede. Os autores propõem um quadro hidrodinâmico geral para

redistribuir a carga entre os processadores de forma que cada processador obtém uma fatia da carga de forma proporcional à sua capacidade de processamento. Eles definem uma função de energia potencial para G de forma que o seu valor mínimo corresponda ao estado de equilíbrio do sistema onde a altura das colunas de líquido seja a mesma em todos os cilindros. A técnica de vizinho-mais-próximo é utilizada para migrar as tarefas entre os processadores. A Figura 2.1 mostra o esquema de funcionamento.

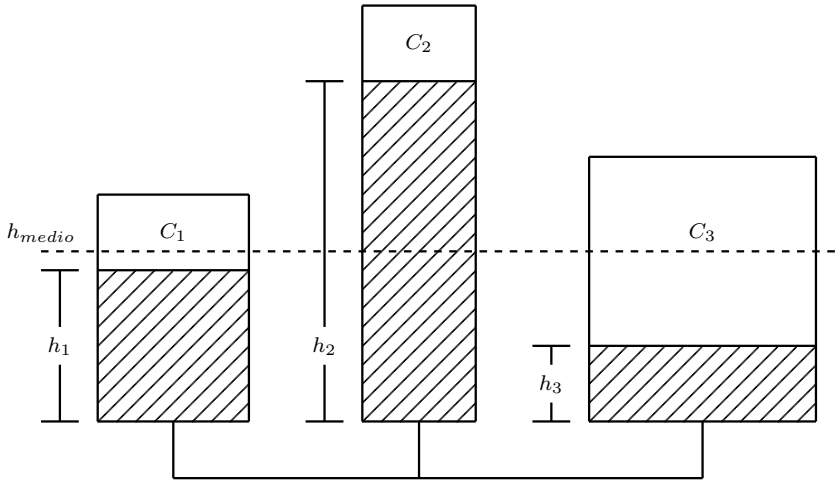

Figura 2.1: Esquema de balanceamento de carga do modelo hidrodinâmico

Cada processador n_i é associado a um cilindro de líquido cuja área transversa corresponde a $c_i > 0$. Para todo $n_i \in N$ a energia potencial da coluna de líquido em n_i é definida como $PE(n_i) = \frac{c_i h_i^2}{2}$, onde h_i é a altura da coluna de líquido em n_i. A energia potencial total de G é definida como a soma da energia potencial em todos os nós. Os autores consideram um canal líquido infinitamente fino entre a base de dois cilindros se existir uma conexão entre os dois processadores correspondentes em G. O equilíbrio global é alcançado quando a altura de todas as colunas de líquido nos cilindros forem a mesma. Eles mostram que este estado de equilíbrio corresponde à energia potencial global mínima. A quantidade de carga transferida do nó n_i para o nó n_j é dada por $\gamma \frac{c_i c_j}{c_i + c_j}(h_i - h_j)$ onde γ é definida como o *fator de balanceamento* o qual encontra-se no intervalo $(0, 1)$ e é utilizado para controlar a quantidade do fluxo de tarefas. É assumido que os canais de comunicação possuem um tempo de atraso fixo tal que a atividade de balanceamento de carga é completada em um intervalo finito de tempo B. Cada passo do balanceamento de carga possui dois passos: Troca de informações e Migração. Os autores mostram que, seguindo este procedimento, a energia potencial

total converge geometricamente para o estado ótimo. Eles aplicaram este modelo de balanceamento em oito topologias de rede diferentes: árvore binária, completa, hipercubo, linear, malha, anel, estrela e torus, e encontram um resultado favorecendo as topologias hipercubo e torus que apresentaram o menor tempo de balanceamento de carga.

2.2 Modelo de Chiasson *et al.*

Em [33], um novo modelo contínuo, dinâmico, não-linear e determinístico com atrasos foi desenvolvido para caracterizar os efeitos deste último, presente em links de comunicação, no problema de balanceamento de carga em clusters de processadores. O modelo é mostrado como sendo auto-consistente, já que o tamanho das filas não pode ser negativo e o número total de tarefas presentes nas filas e trafegando na rede é conservado.

Tal modelo foi concebido para estudar o efeito dos atrasos em relação a diferentes políticas de balanceamento em bancos de dados paralelos. No caso, o NDIS (National DNA Index System) e o CODIS (Combined DNA Index System).

Um nó raiz comunica-se com k grupos de redes de computadores. Cada um desses grupos é composto de n nós (*hosts*) contendo cópias idênticas de parte do banco de dados. Qualquer par de grupos corresponde a bases de dados diferentes, as quais não são necessariamente disjuntas. Um registro específico (ou perfil de DNA no caso) é geralmente armazenado em dois grupos por questões de redundância contra problemas de falha.

As bases de dados são implementadas como um conjunto de filas com mecanismos de buscas (*threads*), de forma que haja uma thread de busca por nó. As requisições de busca (*queries* no jargão de banco de dados) são criadas não somente pelos clientes da base de dados, mas também pelos próprios processos de busca já que a árvore de índices é descendente para qualquer requisição. Dentro deste contexto, uma tarefa é interpretada como uma query e o conjunto de queries forma uma fila. Todas as tarefas que devem ser processadas são chamadas de carga.

Requisições de busca que esperam por seu processamento podem ser inseridas em qualquer fila associada a um mecanismo de busca, e o conteúdo dessas filas podem ser movidas arbitrariamente entre nós de processamento de um grupo de forma a alcançar um balanceamento [33].

O modelo utilizado em [33] é dado por:

$$\frac{dx_i(t)}{dt} = \lambda_i - \mu_i(x_i(t)) + v_i(y_i(t)) - \sum_{j=1}^{n} p_{ij} \frac{t_{p_i}}{t_{p_j}} v_j(y_j(t - h_{ij})) \tag{2.1}$$

$$v_i(y_i(t)) = -K_i \phi(y_i(t)) \tag{2.2}$$

$$\phi(y_i(t)) = uhsat(y_i(t)) \tag{2.3}$$

$$y_i(t) = x_i(t) - \frac{\sum_{j=1}^{n} x_j(t - \tau_{ij})}{n} \tag{2.4}$$

onde

$$p_{ij} \triangleq \frac{sat(x_j^{avg} - x_i(t - \tau_{ji}))}{\sum sat(x_j^{avg} - x_i(t - \tau_{ji})}; p_{ij} \geq 0, p_{jj} = 0, \sum_{i=1}^{n} p_{ij} = 1 \tag{2.5}$$

$$uhsat(y_i(t)) = \begin{cases} y_{max} & \text{se} \quad y_i(t) > y_{max} \\ y_i(t) & \text{se} \quad 0 \leq y_i(t) \leq y_{max} \\ 0 & \text{se} \quad y_i(t) < 0 \end{cases} \tag{2.6}$$

Neste modelo, temos os seguintes parâmetros:

- n é o número de nós;
- $x_i(t)$ é o *tempo de espera estimado* experimentado por uma tarefa inserida em uma *fila* do i-ésimo nó. Sendo $q_i(t)$ o número de tarefas no i-ésimo nó e t_{p_i} o tempo médio necessário para processar uma tarefa no i-ésimo nó. O *tempo de espera estimado* médio é dado por $x_i(t) = q_i(t)t_{p_i}$. Note que $x_j/t_{p_j} = q_j$ é o número de tarefas na fila do nó j. Se estas tarefas forem transferidas para o nó i, então o tempo de espera estimado transferido é $q_j t_{p_i} = x_j t_{p_i}/t_{p_j}$, logo, a fração t_{p_i}/t_{p_j} converte tempo de espera no nó j para tempo de espera no nó i;
- $\lambda_i \geq$ é a taxa de crescimento do tempo de espera no i-ésimo nó causado pela adição de tarefas (taxa de crescimento de x_i);
- $\mu_i \geq 0$ é a taxa de redução no tempo de espera causado pela execução das tarefas no i-ésimo nó e é dado por $\mu_i \equiv (1 \times t_{p_i})/t_{p_i} = 1$ para todo i se $x_i(t) > 0$, enquanto se $x_i(t) = 0$, então $\mu_i \triangleq 0$, ou seja, se não há tarefas na fila, então o tamanho da fila não pode decrescer;

- $v_i(t)$ é taxa de remoção (transferência) das tarefas do nó i no tempo t pelo algoritmo de balanceamento de carga no nó i. Notar que $v_i(t) \leq 0$;
- $p_{ij}v_j(t)$ é a taxa a qual o nó j envia tempo de espera para o nó i no tempo t onde $p_{ij} \geq 0, \sum_{i=1}^{n} p_{ij} = 1$ e $p_{jj} = 0$. Ou seja, a transferência do tempo de espera estimado $\int_{t_1}^{t_2} v_j(t)dt$ do nó j, no intervalo de tempo $[t_1, t_2]$ para os outros nós é realizado com o i-ésimo nó recebendo a fração $p_{ij}(t_{p_i}/t_{p_j}) \int_{t_1}^{t_2} v_j(t)dt$, onde a razão t_{p_i}/t_{p_j} converte o tempo de espera no nó j para o tempo de espera no nó i. Como $\sum_{i=1}^{n}(p_{ij} \int_{t_1}^{t_2} v_j(t)dt) = \int_{t_1}^{t_2} v_j(t)dt$, isto resulta na remoção total do tempo de espera $\int_{t_1}^{t_2} v_j(t)dt$ do nó j. A quantidade $-p_{ij}v_j(t - h_{ij})$ é a taxa de crescimento (taxa de transferência) do tempo de espera estimado (tarefas) no tempo t do nó j para o nó i, onde $h_{ij}(h_{ii} = 0)$ é o tempo de atraso para a transferência do nó j para o nó i;
- Os parâmetros $\tau_{ij}(\tau_{ii} = 0)$ representam os tempos de atraso para a comunicação dos tempos de espera estimados do nó j para o nó i. O parâmetro $x_i^{avg} = (\sum_{j=1}^{n} x_j(t - \tau_{ij}))/n$ é a **estimativa** (devido a atrasos) do i-ésimo nó do tempo de espera estimado médio da rede e é chamado de *média local* (estimativa local da média).

As Figuras 2.2 e 2.3 mostram as curvas características da funções $uhsat(y_i(t))$ e $\mu(x_i(t))$, respectivamente, utilizadas no modelo (|refMGC).

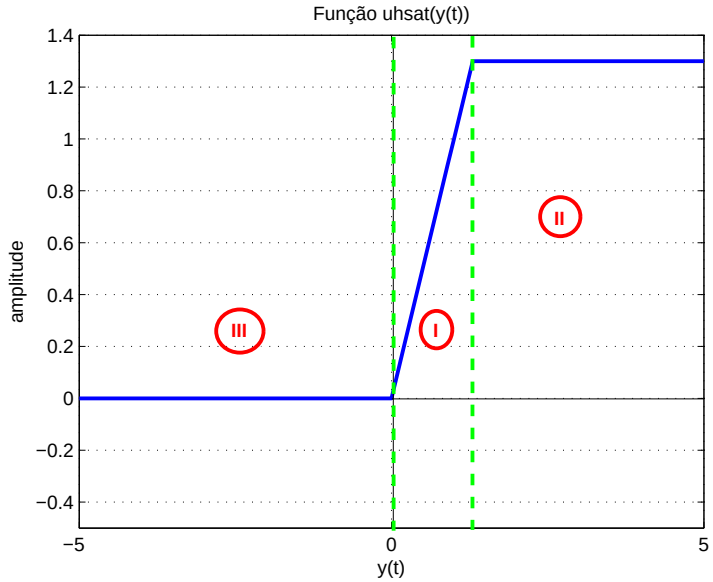

Figura 2.2: Curva característica da função não-linear $uhsat(y_i(t))$ empregada no modelo (2.1).

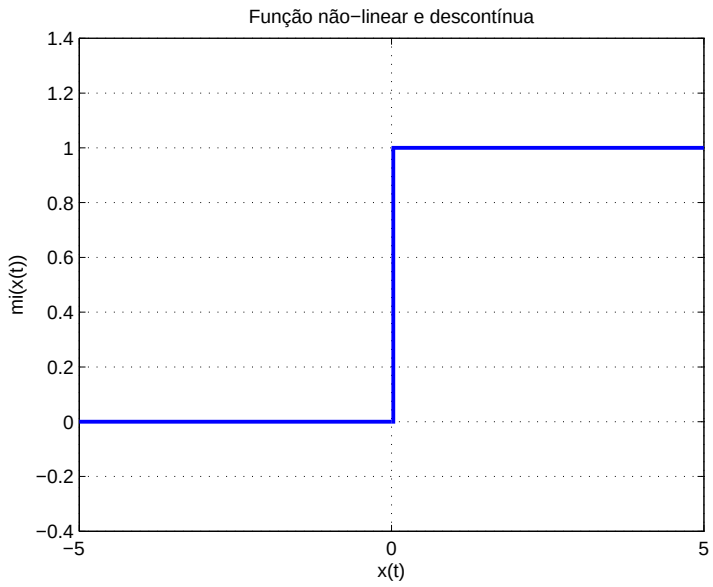

Figura 2.3: Curva característica da função não-linear $\mu(x_i(t))$ empregada no modelo (2.1).

Neste modelo, todas as taxas estão em unidades da *taxa de mudança* do tempo de espera estimado, ou tempo/tempo que é adimensional. Os detalhes podem ser encontrados em [30].

Podemos notar que, como $v_i(t) \leq 0$, o nó i só pode enviar tarefas para os outros nós e não pode iniciar nenhuma outra transferência de um outro nó para ele. A lei de controle $v_i(t) = -K_i\phi(y_i(t))$ nos diz que: Se a saída do i-ésimo nó $x_i(t)$ estiver acima da média local $(\sum_{j=1}^{n} x_j(t-\tau_{ij})/n)$, então ele enviará tarefas aos outros nós, enquanto se ela for menor que a média local, então nada será enviado. O j-ésimo nó recebe a fração $\int_{t_1}^{t_2} p_{ij}(t_{p_i}/t_{p_j})v_i(t)dt$ do tempo de espera transferido $\int_{t_1}^{t_2} v_i(t)dt$ atrasado de h_{ij}.

As figuras (2.2) e (2.3) representam as funções não lineares do modelo, sendo que a função $\mu(x(t))$ apresenta uma descontinuidade e é responsável pelo consumo de tarefas no modelo.

Após o estudo dos trabalhos dos autores relacionados a este tema [30, 32, 33, 45], podemos dizer que as hipóteses assumidas pelos autores sobre este modelo são:

1. Inicialmente, o modelo foi proposto de forma a ocorrer consumo de tarefas simultaneamente ao balanceamento delas [44–46]. Isto foi modificado em um trabalhos posterior [30];

2. Considera-se que o número de tarefas seja suficientemente alto de forma que o tamanho das filas possa ser aproximado por uma variável contínua, daí o modelo ser

contínuo;

3. O modelo considera que todos os nós conseguem trocar informações diretamente, como em uma topologia full-mesh ou barramento[1]. Mas para as redes construídas com topologia hipercubo, rede binária, etc..., isto pode não se aplicar muito bem [47];

4. As funções não-lineares $\phi(y_i(t)) = uhsat(y_i(t))$ são as mesmas.

Algumas características que o modelo (2.1) não trata:

1. O modelo considera o atraso de comunicação τ_{ij} e o atraso de transferência h_{ij} entre dois nós i e j. No entanto, seu controlador dado pela equação (2.2) não considera o custo da troca;

2. Apesar do modelo considerar um tempo médio de atraso de comunicação e atraso de transferência, este último é diretamente proporcional à quantidade de tarefas a ser transferida, pois quanto maior a carga, maior o tempo necessário para que ela chegue completamente ao(s) nó(s) de destino.

3. Os autores não apresentam uma forma de determinar o ganho à luz da teoria de controle, tornando a escolha destes subjetiva.

Para ilustramos o funcionamento do modelo (2.1) vamos apresentar um exemplo extraído de [33].

Exemplo 2.2.1. *[33] Seja um cluster homogêneo composto de três nós n_1, n_2 e n_3 com as seguintes capacidades de processamento respectivamente: $t_{p_1} = t_{p_2} = t_{p_3} = 10\mu s$. O atraso de comunicação (τ_{ij}) é de $200\mu s$ e o atraso de transferência (h_{ij}) é de $400\mu s$. Se as condições iniciais das filas são dadas por $x_1(t) = 0,6$, $x_2(t) = 0,4$ e $x_3(t) = 0,2$, então a Figura 2.4 apresenta o resultado da simulação do modelo para a seguinte matriz de ganhos K:*

$$K = \begin{bmatrix} 6666,7 & 0 & 0 \\ 0 & 4166,7 & 0 \\ 0 & 0 & 5000,0 \end{bmatrix} \tag{2.7}$$

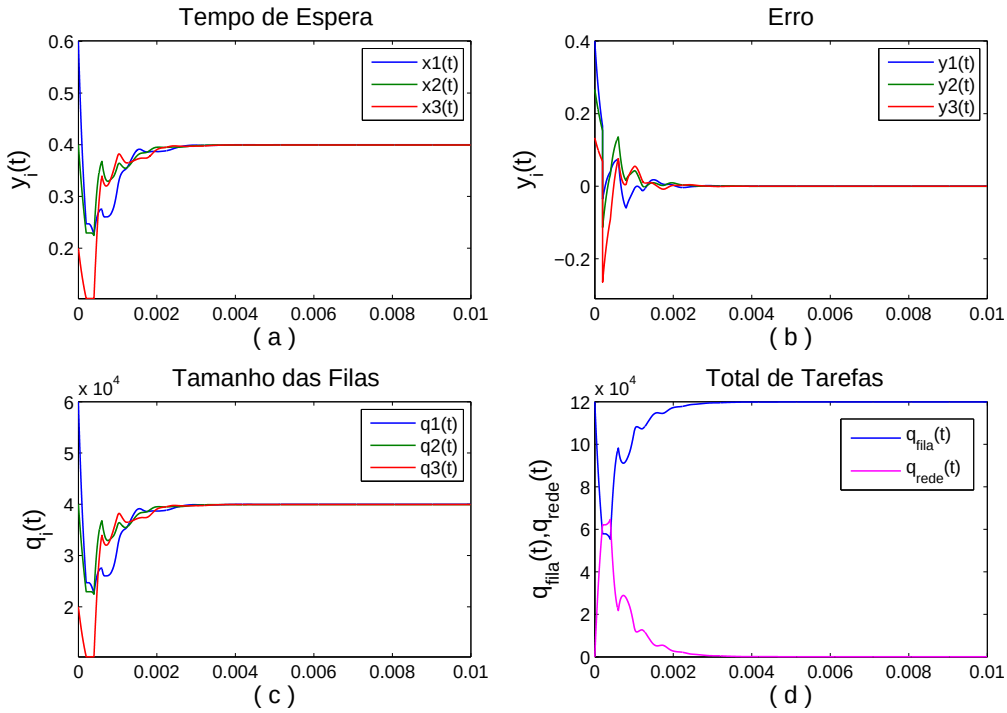

Figura 2.4: Simulação do modelo (2.1) com atrasos ($\tau_{ij} = 200\mu s$ e $h_{ij} = 400\mu s$)

A Figura 2.4a nos mostra o resultado da simulação para o exemplo (2.2.1), onde podemos verificar o efeito oscilatório provocado pelos atrasos de comunicação e transferência. Como $y(t) = [y_1 \quad y_2 \quad \dots \quad y_n]^T$ é na verdade o erro entre o tempo de espera estimado do i-ésimo nó e a média dos tempos de espera estimado de todos os n nós do sistema, então, em condição de equilíbrio de carga, $y(t) \rightarrow \mathbf{0}$ quando $t \rightarrow \infty$ - Figura 2.4a. A Figura (2.4c) nos mostra a evolução do número de tarefas em cada fila durante a simulação. A Figura 2.4d nos mostra o total de tarefas presentes nas filas (em azul) e o total de tarefas trafegando na rede (em magenta) ao longo da simulação. Podemos notar que o número total de tarefas nas filas cai durante os primeiros instantes da simulação e depois retorna ao mesmo valor inicial (em azul), enquanto o número total de tarefas presentes na rede aumenta (em magenta) e depois diminui. A soma, em cada instante de tempo, do valor numérico da curva em azul com o seu respectivo valor da curva em magenta é constante e igual ao número total de tarefas presentes nas filas no início da simulação. De certa forma, a rede pode ser vista como se fosse mais uma fila e temporária.

[1]Mesmo que haja roteadores e switches no caminho entre duas máquinas, estes equipamentos não participam efetivamente da troca de tarefas, ou seja, eles não tem filas onde se possa armazenar e processá-las. Eles contribuem apenas para aumentar o atraso.

Se desprezarmos o atraso de comunicação, que normalmente é muito menor que o atraso de transferência, e fizermos a aproximação $p_{ij} = p = 1/(n-1)$, utilizada em [33], podemos reescrever (2.1) e colocá-lo na forma matricial, já fazendo a substituição de (2.2):

$$\dot{x}(t) = \lambda - \mu(x(t)) - K_d\phi(y(t)) + TK_d\phi(y(t-h)) \tag{2.8}$$

onde $\dot{x}(t) = [x_1(t) \quad x_2(t) \quad \cdots \quad x_n(t)]^T$ é o vetor de estados, $\phi(y(t-h)) = [\phi_1(y_1(t-h)) \quad \phi_2(y_2(t-h)) \quad \cdots \quad \phi_n(y_n(t-h))]^T$ é o vetor de funções não-lineares, $\lambda = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_n]^T$ é vetor de geração de tarefas, $K_d = diag(K_1 \quad K_2 \quad \cdots \quad K_n)$ é a matriz diagonal de ganhos e

$$T = p \begin{bmatrix} 0 & \frac{t_{p_1}}{t_{p_2}} & \cdots & \frac{t_{p_1}}{t_{p_n}} \\ \frac{t_{p_2}}{t_{p_1}} & 0 & \cdots & \frac{t_{p_2}}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{t_{p_n}}{t_{p_1}} & \frac{t_{p_n}}{t_{p_2}} & \cdots & 0 \end{bmatrix}_{n \times n} \tag{2.9}$$

é a matriz de *conversão de tempo de espera* entre os nós. Este é um modelo aproximado, na forma matricial, do modelo (2.1).

O próximo exemplo segue os mesmos parâmetros do exemplo anterior. A única diferença é a ausência do atraso de comunicação ($\tau_{ij} = 0$).

Podemos notar na Figura 2.5 que a resposta é menos oscilatória que a resposta mostrada na Figura 2.4.

A Figura 2.6 nos mostra a mesma simulação caso desprezássemos o atraso de comunicação ($\tau_{ij} = 0$) e de transferência ($h_{ij} = 0$). Pode-se notar que a troca de tarefas ocorre de maneira muito mais suave do que nas outras duas simulações. Podemos notar também que a quantidade de tarefas nas filas permanece constante por todo o período da simulação.

As três simulações anteriores foram apresentadas para validar o modelo em Simulink[2], e formar uma base de comparação para as simulações futuras, que poderão incluir ou não o atraso de transferência já que a ausência do atraso de comunicação será uma das hipóteses assumidas neste trabalho.

[2]Pacote gráfico para simulações da Mathworks Inc.

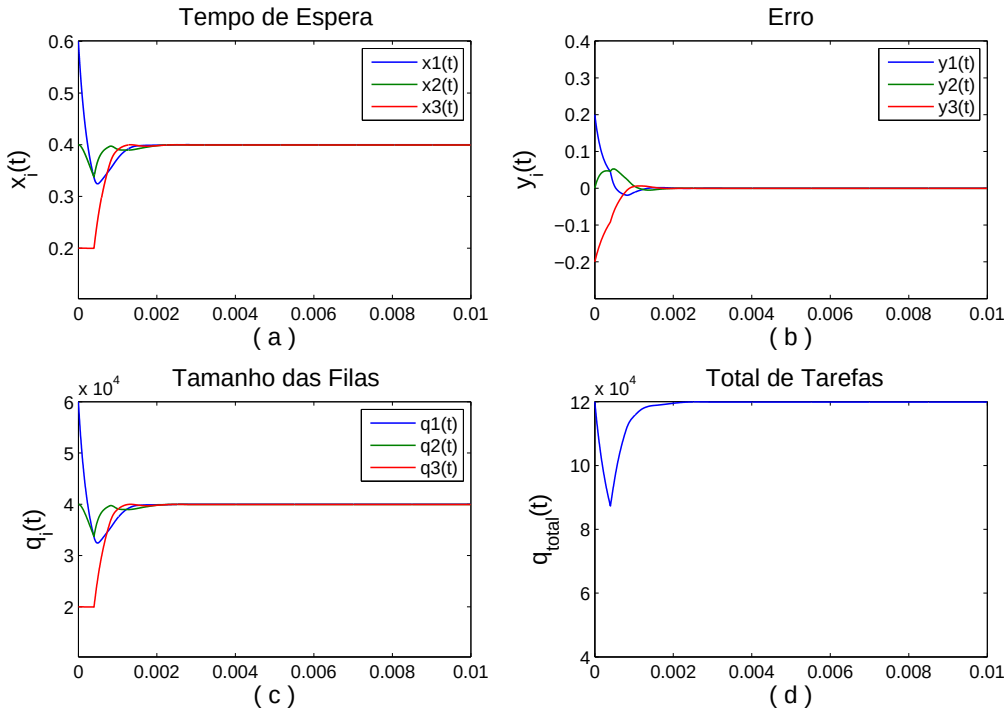

Figura 2.5: Simulação do modelo (2.1) sem atraso de comunicação ($\tau_{ij} = 0\mu s$ e $h_{ij} = 400\mu s$)

2.2.1 Análise de Conservação do Número de Tarefas

Uma das mais importantes características do **MGC** é a conservação do total de tarefas em todas as filas e na rede de comunicação durante o balanceamento de carga [33].

Somando-se a equação:

$$\frac{d}{dt}\left(\frac{x_i(t)}{t_{p_i}}\right) = \frac{\lambda_i - \mu_i}{t_{p_i}} + \frac{v_i(t)}{t_{p_i}} - \sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t - h_{ij}) \tag{2.10}$$

para $i = 1, 2, \ldots, n$, obtemos então:

$$\frac{d}{dt}\left(\sum_{i=1}^{n} q_i(t)\right) = \sum_{i=1}^{n}\left(\frac{\lambda_i - \mu_i}{t_{p_i}}\right) + \sum_{i=1}^{n}\frac{v_i(t)}{t_{p_i}} - \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}} v_j(t - h_{ij}) \tag{2.11}$$

o que representa a taxa de de mudança do comprimento total em todos os nós. No entanto, a própria rede de comunicação contém tarefas em trânsito entre os nós, que também devem ser computadas. O modelo dinâmico do tamanho das filas na rede é dado por

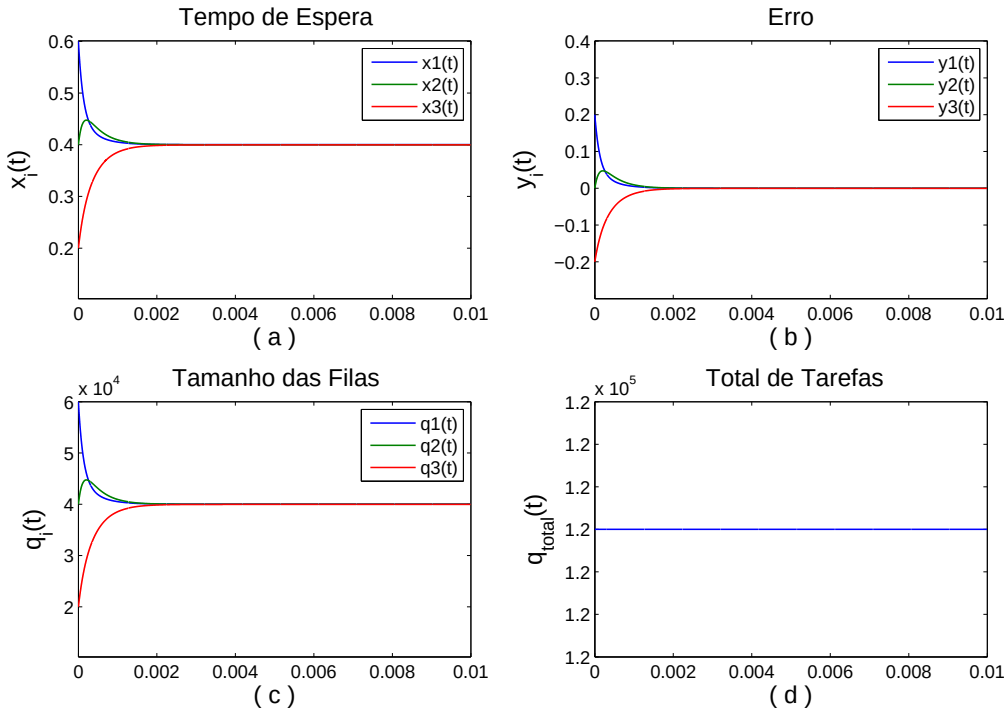

Figura 2.6: Simulação do modelo (2.1) sem atrasos ($\tau_{ij} = 0\mu s$ e $h_{ij} = 0\mu s$)

$$\frac{d}{dt}q_{rede_i}(t) = \sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t - h_{ij}) - \sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t) \tag{2.12}$$

onde q_{rede_i} é o número de tarefas postas na rede que estão sendo enviadas ao nó i. Esta equação simplesmente diz que o j-ésimo nó está colocando tarefas na rede a fim de enviá-las para o i-ésimo nó à uma taxa de $\frac{p_{ij}}{t_{p_j}} u_j(t)$, enquanto o i-ésimo nó está recebendo estas tarefas do nó j pela rede à uma taxa de $\frac{p_{ij}}{t_{p_j}} u_j(t - h_{ij})$. Somando as equações (2.12) para todos os nós, teremos

$$\begin{aligned} \frac{d}{dt}\left(\sum_{i=1}^{n} q_{rede_i}(t)\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t - h_{ij}) - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} v_j(t - h_{ij}) - \sum_{j=1}^{n} \frac{v_j(t)}{t_{p_j}} \end{aligned} \tag{2.13}$$

Adicionando-se as equações (2.11) e (2.13) podemos comprovar a conservação do número total de tarefas através de

$$\frac{d}{dt}\sum_{i=1}^{n}\left(q_i(t)+q_{rede_i}(t)\right)=\sum_{i=1}^{n}\left(\frac{\lambda_i-\mu_i}{t_{pi}}\right) \tag{2.14}$$

Em outras palavras, o número total de tarefas que estão no sistema (filas + rede) pode aumentar somente pela taxa de chegada de tarefas $\sum_{i=1}^{n}\lambda_i/t_{p_i}$ em todos os nós, ou de forma similar, decrescer pela taxa de processamento de tarefas $\sum_{i=1}^{n}\mu_i/t_{p_i}$ em todos os nós. O balanceamento de carga não pode, por si só, incrementar ou decrementar o número total de tarefas presentes nas filas.

Neste trabalho, a matriz de ganhos K não se limitará apenas à uma matriz diagonal, pois um dos objetivos será o de flexibilizar a ação de controle.

No entanto, a proposição de uma matriz de ganhos deste tipo necessita de uma demonstração de que a conservação do número total de tarefas será mantida.

No apêndice A é dada a prova de conservação do número total de tarefas para uma matriz de ganhos não-diagonal. Neste mesmo apêndice, ainda investigaremos a possibilidade do uso de duas matrizes de ganhos diferentes: K (matriz de realimentação não-linear) e K^{τ} (matriz de realimentação não-linear com atraso).

A justificativa para esta investigação decorre do fato de que a equação (3.30) do modelo de rede neural com atraso proposto pode ser escrito como:

$$\dot{y}(t)=-RK\phi(y(t))+RTK^{\tau}\phi(y(t-h))+R\lambda \tag{2.15}$$

onde a matriz de ganhos K deixa de ser única nos dois termos. Isto pode permitir uma flexibilidade maior da ação de controle.

2.2.2 Análise de Estabilidade

O controlador do modelo (2.1) é dado por

$$v_i(t)=-K_i uhsat(y_i(t)) \tag{2.16}$$

onde o ganho $K_i>0$ deverá ser especificado. Fisicamente, estes ganhos estão limitados pelas restrições de banda da rede. De acordo com [33], o termo p_{ij} pode ser visto também como parâmetros de controle a ser especificado em função das restrições de banda.

O modelo (2.1) é assintoticamente estável no sentido de Lyapunov para qualquer con-

junto de ganhos $K_i > 0$ e qualquer conjunto $p_{ij} \geq 0$ com $\sum_{i=1}^{n} p_{ij} = 1$. Especificamente, temos o seguinte teorema:

Teorema 2.1. *[33] Dado o sistema descrito pelas equações (2.1)-(2.4) e (2.12) com* $\lambda_i = 0$ *para* $i = 1, 2, \ldots, n$ *e condições iniciais* $x_i(0) \geq 0$*, então* $(q_i(t), q_{rede_i}(t)) \to 0$ *conforme* $t \to \infty$*.*

Prova: Primeiro pode-se notar que q_{rede_i} é não-negativa através da equação (2.12). Segue então que

$$q_{rede_i}(t) = -\sum_{j=1}^{n} \frac{p_{ij}}{t_{p_j}} \left(\int_{t-h_{ij}}^{t} v_j(\tau) d\tau \right) \geq 0 \tag{2.17}$$

Sob as condições do teorema (2.1), a equação (2.14) torna-se

$$\frac{d}{dt} \sum_{i=1}^{n} (q_i(t) + q_{rede_i}(t)) = -\sum_{i=1}^{n} \frac{\mu_i(q_i)}{t_{p_i}} \tag{2.18}$$

Seja $V(t) \triangleq \sum_{i=1}^{n} (q_i(t) + q_{rede_i}(t))$ e, como $q_i(t)$ e $q_{rede_i}(t)$ são não-negativas, $V(t) \geq 0$ e é igual a zero se, e somente se, $q_i(t) = q_{rede_i}(t) = 0$ para todo i. Ainda, como $\mu_i(q_i(t)) = 1$ para $q_i(t) > 0$ e $\mu_i(q_i(t)) = 0$ se, e somente se, $q_i(t) = 0$, segue então que $dV/dt = -\sum_{i=1}^{n} \mu_i(q_i(t))/t_{p_i} \leq 0$. Isto implica em

$$V(t) = V(0) - \int_0^t \sum_{i=1}^{n} \frac{\mu_i(q_i(t))}{t_{p_i}} dt \geq 0 \tag{2.19}$$

é monotonicamente decrescente. Como $V(t)$ é limitada inferiormente, temos $V(t) \downarrow V_f \geq 0$, ou

$$\lim_{t \to \infty} \int_0^t \sum_{i=1}^{n} \frac{\mu_i}{t_{p_i}} dt = V(0) - V_f \geq 0 \tag{2.20}$$

A quantidade $\mu_i(q_i(t))$ é 1 ou 0 dependendo de quando $q_i(t)$ é positiva ou nula, então $\mu_i(q_i(t))$ pode ser visto como um conjunto de pulsos de altura unitária e largura variável. A integral $\int_0^\infty \mu_i(q_i(t)) dt$ é finita por (2.20) o que implica que as larguras dos pulsos unitários que compõem $\mu_i(q_i(t))$ devem tender a zero conforme $t \to \infty$. Então, mesmo que uma fila de tamanho $q_i(t) = x_i(t)/t_{p_i}$ continue a chavear entre zero e valores positivos, os intervalos de tempo para o qual ela é não nula deve tender a zero conforme $t \to \infty$.

Ou seja, o tamanho da i-ésima fila $q_i(t)$ é não negativa, contínua, limitada pela função não negativa monotonicamente decrescente $V(t)$, e os intervalos onde $q_i(t)$ é não negativa tende a zero conforme $t \to \infty$. De forma mais precisa, seja $I_{t,h} = [t-h, t]$ e se definirmos $E_{t,h} = \{s \in I_{t,h} : q_i(s) > 0\}$, então a medida de Lebesgue de $E_{t,h}$, representada por $m(E_{t,h})$, converge para zero conforme $t \to \infty$ para todo h. Ainda mais, conforme $u_i(t) \leq 0$ é sempre verdadeiro e

$$\begin{aligned} u_i(t) &= -K_i uhsat\left(x_i(t) - \frac{\sum_{j=1}^{n} x_j(t-\tau_{ij})}{n}\right) \\ &= -K_i uhsat\left(t_{p_i} q_i(t) - \frac{\sum_{j=1}^{n}\left(\frac{t_{p_i}}{t_{p_j}}\right) q_j(t-h_{ij})}{n}\right) \end{aligned} \tag{2.21}$$

Segue então que os intervalos de tempo para os quais as funções limitadas $v_i(t)$ sejam não nulos deve convergir para zero conforme $t \to \infty$. A partir de observações, segue que $v_i(t) < 0$ necessariamente implica em $q_i(t) > 0$. Logo, a integral (2.17) pode ser limitada superiormente por $\int_{[t-h_{ij},t] \cap E_{t,h_{ij}}} |K_i| y_{max} d\tau = |K_i| y_{max} m(E_{t,h_{ij}})$ (Relembre que $E_{t,h} \subset I_{t,h}$), que converge para zero conforme $t \to \infty$. Conseqüentemente, pela equação (2.17), $q_{rede_i}(t) \to 0$ conforme $t \to \infty$.

Agora, mostraremos que a função decrescente monotonicamente $V(t)$ deve convergir para zero, ou seja, $\lim_{t\to\infty} V(t) = V_f = 0$. Suponhamos que não, de forma que $V_f > 0$. Conforme $q_{rede_i}(t) \to 0$, escolhamos t_1 grande o suficiente de forma que $0 \leq \sum_{i=1}^{n} q_{rede_i}(t) < \epsilon V_f$ para $t > t_1$, onde $0 < \epsilon < 1$. Como

$$V(t) = \sum_{i=1}^{n} (q_i(t) + q_{rede_i}(t)) \geq V_f, \quad \forall t \tag{2.22}$$

temos

$$\sum_{i=1}^{n} q_i(t) \geq (1-\epsilon) V_f > 0, \quad \text{para} \quad t > t_1 \tag{2.23}$$

Para cada $t > t_1$, existe pelo menos um i (o qual depende de t) para cada $q_i(t) > 0$. Então, $\sum_{i=1}^{n} \mu_i(q_i(t))/t_{p_i} dt \geq \min\{1/t_{p_i}\}$ para todo $t > t_1$. Pela equação (2.18) temos então que

$$V(t) = V(t_1) - \int_{t_1}^{t} \sum_{i=1}^{n} \frac{\mu_i(q_i(t))}{t_{p_i}} dt \leq V(t_1) - \int_{t_1}^{t} \min\left\{\frac{1}{t_{p_i}}\right\} dt \qquad (2.24)$$

Conforme o lado direito da equação (2.24) eventualmente torna-se negativo, temos uma contradição e então $V_f = 0$. Como já foi demonstrado que $q_{rede_i}(t) \rightarrow 0$ para todo i, $V(t) \rightarrow 0$ então implica que $q_i(t) \rightarrow 0$ para todo i. Isto completa a prova do teorema.

2.2.3 Determinação de Ganhos

Para explicar a conexão existente entre o ganho de controle K_i e a implementação do modelo (2.1), devemos relembrar que o ganho representa a taxa de redução do tempo de espera por segundo no modelo (2.1), e que $y_i(t) = (q_i(t) - n^{-1} \sum_{j=1}^{n} q_j(t - \tau_{ij}))t_{p_i} = r_i(t)t_{p_i}$, onde $r_i(t)$ é o número de tarefas acima da média estimada (local) do número de tarefas em todos os nós.

A implementação do algoritmo de balanceamento deve executar a lei de controle repetidamente com um tempo de intervalo (possivelmente aleatório) entre as ações de balanceamento. Sendo Δt_i o tempo de intervalo entre execuções sucessivas do algoritmo de balanceamento no i-ésimo nó, e a lei de controle contínua $u(t) = -K_i y_i(t)$, uma lei de controle discreta é definida de forma a remover uma fração da fila de tarefas $K_z r_i(t)$, $(0 < K_z < 1)$ em um tempo Δt_i. A taxa de redução do tempo de espera é $-K_z r_i(t) t_{p_i} / \Delta t_i = -K_z y_i(t) / \Delta t_i$, de forma que uma lei de controle contínua equivalente, dado o ganho K_z de tempo discreto e o intervalo de controle Δt_i, é dada por:

$$u(t) = -\frac{K_z y_i(t)}{\Delta t_i} \Rightarrow K_i = \frac{K_z}{\Delta t_i} \qquad (2.25)$$

A lei de controle acima nos mostra o quão rápido o algoritmo de balanceamento de carga pode ser processado e quanto de carga ele consegue transferir.

A Figura 2.7 representa uma política de balanceamento de carga síncrona, onde os nós começam uma iteração de balanceamento simultaneamente e terminam em tempos diferentes (t_{b_i}, $i = 1, 2, \ldots, n$). No entanto, os nós que possuem pouca carga a ser transferida terminam primeiro e acabam esperando pelos nós que possuem mais carga. Como no decorrer do tempo a quantidade de carga vai se equilibrando entre os nós, os tempos t_{b_i} vão aproximando-se um dos outros.

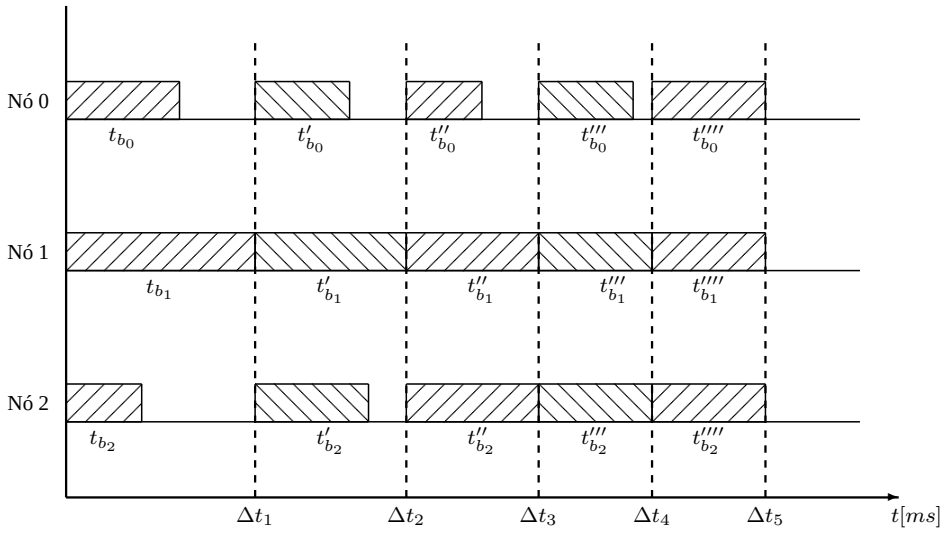

Figura 2.7: Diagrama de tempo do balanceamento de carga síncrono.

Esta política de balanceamento de carga foi a utilizada neste trabalho e é a mais fácil de ser implementada, utilizando-se a função MPI *MPI_Barrier* [48], e difere da política implementada por GHANEM [45], que é totalmente assíncrona.

Uma outra política de implementação, quanto à tempo de processamento do próprio algoritmo de balanceamento de carga, refere-se ao fato de ser possível executar todo o trabalho de balanceamento de uma única vez (*One-Shot Load Balancing*) ou em pequenas iterações repetidas [45], onde alterna-se o algoritmo de balanceamento com o processamento das tarefas. Na implementação deste trabalho, optou-se pela política do One-Shot Load Balancing também por ser de maior facilidade de implementação.

As diferenças de políticas de implementação entre GHANEM-CHIASSON [33] e a deste trabalho não chegam a constituir uma barreira no que tange a avaliação, pois os ganhos independem da forma de implementação. Além do mais, tanto os ganhos encontrados pela metodologia CHIASSON [33] como pela proposta nesta tese são implementados no mesmo algoritmo, permitindo uma comparação direta.

Nos experimentos realizados pelos autores, o intervalo Δt_i se modificava cada vez que o algoritmo de balanceamento de carga era executado. Desta forma, o valor utilizado para Δt era um valor médio.

Em um dos seus experimentos, o valor médio dos ganhos (para $K_z = 0,5$) foram $K_1 = 0,5/75\mu s = 6667$, $K_2 = 0,5/120\mu s = 4167$ e $K_3 = 0,5/100\mu s = 5000$.

Outros ganhos foram calculados para $K_z = 0,1$ a $K_z = 0,6$ também e, visualmente, os

ganhos que deram a melhor resposta foram os escolhidos.

2.3 Modelo de Hopfield-Tank de Redes Neurais

Através de trabalhos publicados e que alcançaram grande repercussão, J.J. Hopfield e D.W. Tank introduziram um modelo dinâmico, hoje conhecido como **modelo de Hopfield-Tank** para representar redes neurais artificiais [49–51]. Este sistema dinâmico possui uma interpretação física com um circuito elétrico contendo amplificadores não-lineares interligados por uma rede de ganhos lineares. Tal circuito pode ser comparado a uma *rede neural artificial*, pois cada amplificador não-linear pode ser visto como um modelo de uma função de ativação neural, e a rede de interconexão, como as sinapses.

A forma geral com que este circuito pode ser representado é dado por [52]:

$$C_i \frac{dx_i(t)}{dt} = -\frac{x_i(t)}{R_i} + \sum_{j=1}^{n} t_{ij} v_j(t) + I_i \tag{2.26}$$

$$v_j(t) = \phi_j(x_j(t)) \tag{2.27}$$

e, na forma matricial

$$C\dot{x}(t) = -Gx(t) + Tv(t) + I \tag{2.28}$$

onde $C > 0$ é uma matriz diagonal de capacitâncias de entrada do amplificador neural; $G > 0$ é a matriz de admitâncias dos amplificadores; $T = t_{ij}$ é a matriz $n \times n$ real e constante de interconexão da rede; I é o vetor de correntes externas e constantes de entrada; $x = [x_1(t), \ldots, x_n(t)]^T$ é o vetor das tensões neurais e o vetor $v(t) = [v_1(t), \ldots, v_n(t)]^T$ são as tensões de saída dos neurônios.

As funções de ativação do neurônio $\phi_i(\cdot)$ possuem as seguintes características [53]:

- $\phi_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, n$;
- $\phi_i(\cdot)$ é limitada;
- $\phi_i(\cdot)$ é não-decrescente;
- $\phi_i(\cdot)$ é uma função contínua por partes em $\mathbb{R}$.

Isto significa que $\phi_i(\cdot)$ pode possuir pontos de descontinuidade de salto, onde existem limites finitos à direita e à esquerda, e que $\phi_i(\cdot)$ possui um número finito de descontinuidades em um intervalo compacto de $\mathbb{R}$.

A importância deste modelo de rede neural para a solução de problemas de agendamento em tempo real foi discutida em [28]. Dentro deste contexto, faz-se necessário que a rede neural tenha um único ponto de equilíbrio, estável e globalmente atrativo, de forma a evitar as chamadas soluções espúrias (mínimos locais).

No entanto, na prática, verificou-se que o modelo (2.28) apresentava comportamento instável em muitas ocasiões, devido ao atraso de tempo enfrentado pelo sinal na realimentação.

O modelo matemático passou então a contemplar os efeitos do atraso e novas provas de convergência vem sendo apresentadas para este novo modelo, muitas empregando *Inequações Matriciais Lineares* (LMI), onde há uma vasta literatura sobre este assunto, como por exemplo o modelo *Hopfield Delayed Neural Network* - **HDNN** - apresentado em [54]:

$$\dot{x}_i(t) = -x_i + \sum_{j=1}^{n} w_{ij}\phi_j(x_j(t)) + \sum_{j=1}^{n} w_{h_{ij}}\phi_j(x_j(t-h)) + u_i, \quad i = 1, 2, \ldots, n \tag{2.29}$$

que, posto na forma matricial, fica:

$$\dot{x}(t) = -x(t) + W\phi(x(t)) + W_h\phi(x(t-h)) + u \tag{2.30}$$

onde $x(t) = [x_1(t), \cdots, x_n(t)]^T$ é o vetor de estados, $\phi(x(t)) = [\phi_1(x_1(t)), \cdots, \phi_n(x_n(t))]^T$ é o vetor de saída, $W = \{w_{ij}\}$ é a matriz de realimentação, $W_h = \{w_{h_{ij}}\}$ é a matriz de realimentação atrasada, $u = [u_1, \cdots, u_n]^T$ é o vetor constante e h é o atraso. A função de ativação $\phi_i(x_i(t))$ é dada por

$$\phi_i(x_i(t)) = 0.5(|x_i(t) + 1| - |x_i(t) - 1|), \quad i = 1, \ldots, n \tag{2.31}$$

e seu gráfico correspondente é apresentado na Figura 2.8.

Na próxima seção, será mostrado que o modelo matemático publicado por HUI *et al.* possui uma similaridade com o modelo publicado por CHIASSON *et al.*, e que este último possui uma similaridade com o modelo de Hopfield-Tank com atraso. Como conseqüência, uma função de Lyapunov correspondente é construída baseada em um dos trabalhos sobre

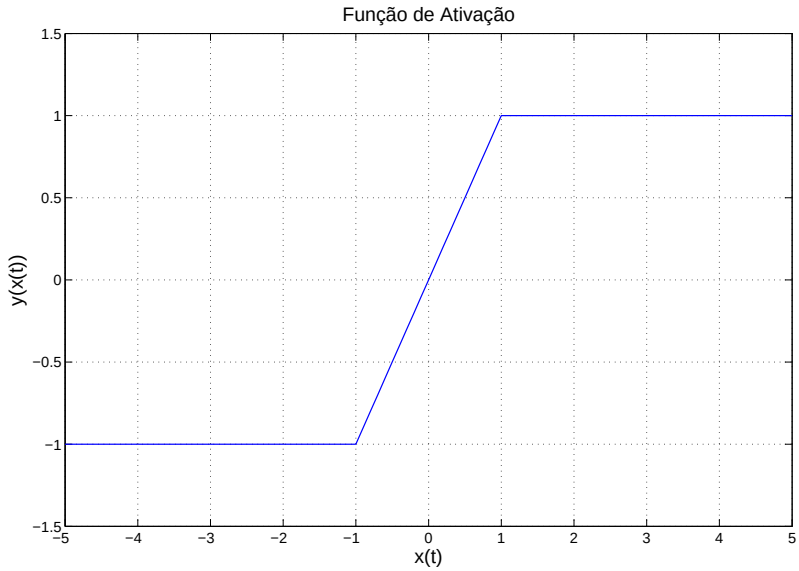

Figura 2.8: Curva característica da função de ativação $\phi_i(x_i(t))$ (2.31)

convergência de redes neurais artificiais com atraso de SINGH [54]. Desta forma, a prova de existência, unicidade e estabilidade global assintótica do ponto de equilíbrio nos modelos é assegurada através da função de Lyapunov expressa na forma de uma LMI.

2.4 Analogia Entre Modelos

Os estudos preliminares dos modelos supra-citados nos permitem realizar uma analogia entre o modelo publicado por HUI *et al.* e o modelo publicado por CHIASSON *et al.* (2.1), a fim de mostrar que estes possuem semelhanças.

Vamos recordar que:

- No modelo de HUI *et al.*

$$v_i(t) = h_i(t)C_i \tag{2.32}$$

onde v_i é a quantidade de tarefas destinadas ao i-ésimo processador (volume), h_i é a altura da coluna de líquido e C_i é a capacidade de processamento deste i-ésimo processador.

Quanto mais rápido for o i-ésimo processador, maior será o termo C_i e, quanto maior a carga de trabalho a ser realizada por este mesmo processador, maior será h_i, ou seja, maior a quantidade total de tarefas v_i.

- No modelo publicado por CHIASSON *et al.*

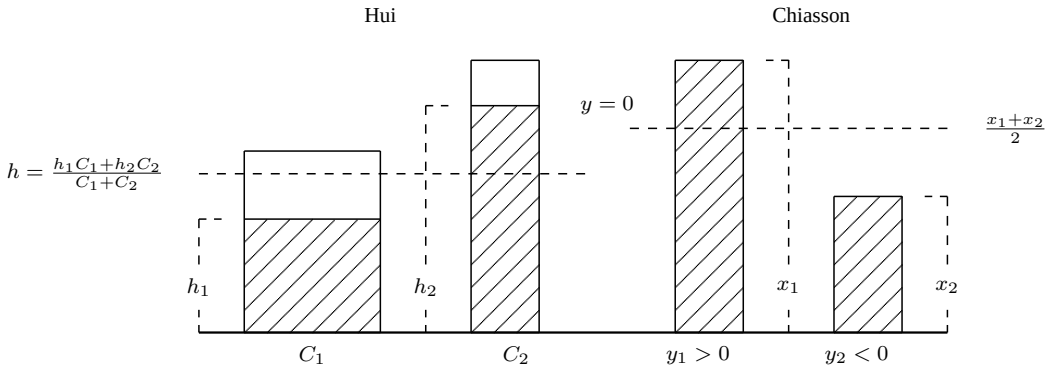

Figura 2.9: A analogia entre os modelos publicados por HUI *et al.* e CHIASSON *et al.*.

$$x_i(t) = q_i(t)t_{p_i} \tag{2.33}$$

onde x_i é o tempo de espera estimado para uma tarefa ser executada no i-ésimo processador, q_i é a quantidade de tarefas na fila i alocadas para este i-ésimo processador e t_{p_i} é o tempo de processamento na i-ésima fila.

Quanto maior o termo t_{p_i}, significa que mais lentamente o processador está consumindo tarefas, e quanto maior a quantidade q_i de tarefas na i-ésima fila, maior ser o tempo de espera x_i.

Estabelecendo a relação

Modelo de Hui *et al.*	Modelo de Chiasson *et al.*
C_i	$\frac{1}{t_{p_i}}$
$v_i(t)$	$q_i(t)$
$h_i(t)$	$x_i(t)$

Podemos adaptar a equação (2.1) para o modelo de Hui, pois como dito anteriormente, não existe uma forma matemática fechada em seu trabalho.

$$\frac{dv_i(t)}{dt} = u(y_i(t)) - \sum_{j=1}^{n} p_{ij}\frac{C_j}{C_i}u(y_j(t)) \tag{2.34}$$

$$u(y_i(t)) = -K_i uhsat(y_i(t)) \tag{2.35}$$

$$y_i(t) = v_i(t) - \frac{\sum_{j=1}^{n} v_j(t)}{n} \tag{2.36}$$

Podemos agora comparar os modelos (2.1), o modelo de Hopfield-Tank com atraso (2.30), com o modelo de HUI *et al.* dado acima para verificar que eles são análogos, mas não estritamente equivalentes.

Olhando cada termo isoladamente dos três modelos na tabela abaixo, podemos apontar algumas diferenças.

Modelo	1^o **Termo**	2^o **Termo**	3^o **Termo**	4^o **Termo**
Hui	$\times$	$(-I+C)K_d\phi(y(v(t)))$	$\times$	$\times$
Chiasson	$-\mu(x(t))$	$-K_d\phi(y(x(t)))$	$TK_d\phi(y(x(t-h)))$	λ
Hopfield	$-x(t)$	$W\phi(x(t))$	$W^h\phi(x(t-h))$	u

O modelo de HUI *et al.* não possui o 1^o termo que é uma realimentação linear, o 3^o termo que é uma realimentação não-linear atrasada, nem o 4^o termo que é o bias[3].

Uma diferença entre o modelo (2.1) e o de Hopfield é que o 1^o termo no modelo (2.1) é uma função não-linear e descontínua de $x(t)$ e não a própria variável de estado $x(t)$. A segunda é que o 2^o e o 3^o termos no modelo (2.1) são funções compostas do tipo $\phi \circ y \circ x$, enquanto no modelo de Hopfield a função composta é do tipo $y \circ x$. Será necessário trabalharmos sobre esta diferença, o que será feito no capítulo 3.

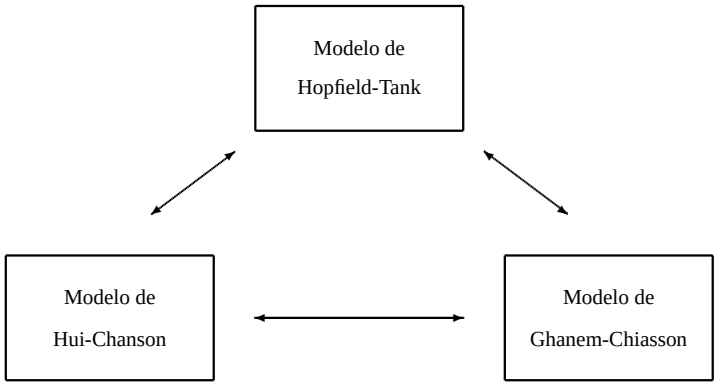

Figura 2.10: Relacionamento entre os modelos estudados

A Figura 2.9 apresenta uma comparação visual da analogia dos modelos de balanceamento apresentados e a Figura 2.10 apresenta o esquemático de relacionamento entre eles e o modelo de redes neurais.

[3]O termo *bias* é o correspondente na linguagem de redes neurais à geração de tarefas no modelo (2.1), ou seja, ele é um caso particular deste último

O termo $\mu(x(t)$ pode ser desconsiderado pois em [33], os autores demonstraram que não há sentido em falarmos de consumo de tarefas simultaneamente ao balanceamento de carga pois, ou o processador está utilizando o seu esforço para consumir tarefas ou para processar o balanceamento de carga. Sendo assim, omitiremos este termo neste trabalho a partir do capítulo 3 em diante.

2.5 Resumo do Capítulo

Neste capítulo, verificamos três modelos conhecidos para o problema de balanceamento de carga e descobrimos algumas similaridades entre eles. Destacamos também as diferenças entre o modelo (2.1) e o modelo de rede neural de Hopfield-Tank com atraso.

Apresentamos uma análise aprofundada do modelo (2.1), já que ele fornece um modelo matemático de seu esquema de balanceamento, enquanto o modelo publicado por HUI *et al.* não o faz, ficando apenas na descrição de um pseudo-algoritmo para implementá-lo. No entanto, sua abordagem facilita o entendimento do funcionamento do modelo (2.1).

Diante das idéias publicadas em [30, 42], podemos realizar um extrato dos pontos de cada modelo e implementar a tabela abaixo nos mostrando o que cada modelo aborda e o que não aborda.

Modelo	**Proc. heterog.**	**Atraso**	**Modelo Formal**	**Cálculo de Ganhos**
Hui	SIM	NÃO	NÃO	NÃO
Chiasson	SIM	SIM	SIM	NÃO
Hopfield	SIM	SIM	SIM	NÃO

Apesar do modelo de Hopfield-Tank não ser especificamente um modelo de balanceamento de carga entre processadores, é possível dizer que ele se aplica ao problema, e com processadores heterogêneos, conforme será demonstrado no próximo capítulo.

A principal contribuição desta tese será na última coluna da tabela acima, pois estabeleceremos princípios para o cálculo dos ganhos à luz da Teoria de Controle, expandindo assim um pouco mais o modelo (2.1).

Para permitir uma comparação justa, o problema a ser tratado nesta tese continuará no domínio contínuo, deixando a abordagem por meios discretos para trabalhos futuros.

Um dos objetivos a ser alcançado é o de reduzir o tempo de espera estimado (variável $x_i(t)$), porém, sem desconsiderar que a própria complexidade na execução das leis de

controle consome tempo de processamento. Tal tempo de processamento também deve ser minimizado.

Capítulo 3

Formulações do Modelo Sob a Forma de Redes Neurais

A idéia de utilizar redes neurais para promover o balanceamento de carga entre microprocessadores advém da similaridade mostrada no capítulo 2 e de algumas características inerentes a elas que são desejáveis em um algoritmo de balanceamento de carga [24, 26, 27, 29]. Eis algumas:

- Velocidade de processamento e custo computacional razoáveis;
- Natureza paralela intrínseca;
- Possibilidade de convergência assintótica quando submetida a atrasos.

Neste capítulo, diversas versões de modelos serão desenvolvidas e estudadas até alcançarmos a versão com a complexidade desejada, onde reforçaremos a similaridade de um modelo aproximado de (2.1) com um caso particular de uma rede neural conhecida como *Hopfield Delayed Neural Network* (**HDNN**). Tal forma particular trata-se na verdade da extensão de um sistema *Persidskii* [55] no qual estaremos incluindo atrasos.

Isto permite abrir um leque de oportunidades de estudo dos efeitos do atraso sobre a convergência de um problema de balanceamento de carga entre microprocessadores, baseando-se no que há de mais recente na literatura e em pesquisas sobre redes neurais artificiais com atraso.

A organização do estudo será feita conforme visto na Figura 3.1.

A tabela a seguir indica as subseções onde cada modelo pode ser encontrado.

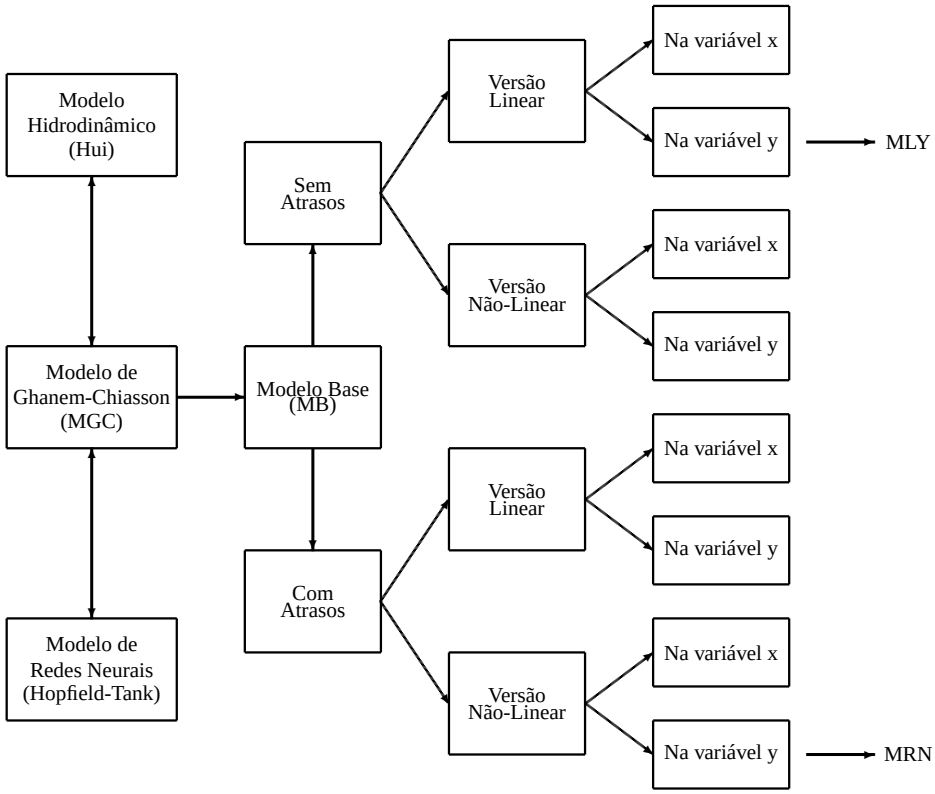

Figura 3.1: Desenvolvimento a partir do Modelo Base

	Versão Linear		**Versão Não-Linear**	
Variável	**x**	**y**	**x**	**y**
Sem Atrasos	seção (3.1.1.1)	seção (3.1.1.2)	seção (3.1.2.1)	seção (3.1.2.2)
Com Atrasos	seção (3.2.1.1)	seção (3.2.1.2)	seção (3.2.2.1)	seção (3.2.2.2)

Tabela (3) - Seqüência de desenvolvimento e suas subseções.

O estudo destes modelos intermediários simplificados possui um papel preponderante no entendimento mais aprofundado do problema, permitindo a análise e comparação dos comportamentos do sistema não-linear e sua versão linear visando estabelecer uma estratégia de controle para o sistema não-linear.

A partir desta premissa, faremos as seguintes hipóteses em relação ao modelo (2.1):

$\mathbf{h_1}$ O atraso de comunicação nos links, por ser geralmente muito menor que o atraso de transferência em um cluster, será desconsiderado;

$\mathbf{h_2}$ O atraso de transferência será considerado único, ou homogêneo;

$\mathbf{h_3}$ O termo correspondente ao consumo de tarefas $\mu(x)$ não será considerado pelo motivo já exposto no capítulo 2, no último parágrafo da seção (2.4);

$\mathbf{h_4}$ O termo p_{ij} será assumido como constante e definido por $p_{ij} = p = 1/(n-1)$, conforme [30];

$\mathbf{h_5}$ Será considerado que não há falha ou desconexão de processadores, conseqüentemente, o número n de processadores do sistema será constante;

$\mathbf{h_6}$ Apesar dos sistemas operacionais modernos atuarem com diversas filas (cada qual com uma prioridade diferente), trabalharemos com todas as filas como sendo uma única pois estaremos considerando o tempo de espera estimado médio $x_i(t)$ para uma tarefa ser executada no i-ésimo processador.

Estas serão as hipóteses assumidas para o modelo que será utilizado no início do estudo. Doravante chamaremos este modelo de **Modelo Base**:

$$\dot{x}_i(t) = v_i(y_i(t)) - \sum_{j=1}^{n} p\frac{t_{p_i}}{t_{p_j}} v_j(y_j(t-h)) + \lambda_i \tag{3.1}$$

onde

$$v_i(y(t)) = -K_i\phi(y_i(t)) \tag{3.2}$$

$$\phi(y_i(t)) = uhsat(y_i(t)) \tag{3.3}$$

$$y_i(t) = x_i(t) - \frac{\sum_{j=1}^{n} x_j(t)}{n} \tag{3.4}$$

$$p = \frac{1}{n-1} \tag{3.5}$$

ou, na forma compacta:

$$\dot{x}(t) = v(y(t)) - Tv(y(t-h)) + \lambda \tag{3.6}$$

onde $v(y(t)) = -K_d\phi(y(t))$, $K_d = diag(K_1, \ldots, K_n)$, $\phi(y(t)) = [\phi(y_1(t)), \ldots, \phi(y_n(t))]^T$.

Nas próximas seções, apresentaremos as versões de modelos derivados do Modelo Base na forma compacta (3.6). No capítulo 4 será abordado o estudo das condições de estabilidade para cada uma das versões geradas e, finalmente no capítulo 5, a síntese de controladores será apresentada.

3.1 Versões do Modelo sem Atraso (Linear e Não-Linear)

Devido ao efeito instabilizante do atraso [30], iniciaremos o estudo das versões dos modelos desprezando-o, e verificando assim os resultados do balanceamento de carga caso as transferências de tarefas e a comunicação fossem instantâneas entre os diversos nós.

Após as versões sem atrasos terem sido entendidas, passaremos ao estudo dos mesmos modelos acrescidos de atrasos e verificaremos os efeitos destes sobre cada um dos modelos.

3.1.1 Versão Linear sem Atraso

A fim de estudarmos os modelos do mais simplificado ao mais complexo, o que propiciará um melhor entendimento de todos os casos particulares do **Modelo Base** (3.6), vamos iniciar com uma versão de modelo que seja uma versão linear deste.

Seja o sistema linear invariante no tempo dado abaixo

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ z(t) = Cx(t) \end{cases} \tag{3.7}$$

onde $y(t) = [y_1(t), \ldots, y_n(t)]^T$ é o vetor de estados formado por n variáveis de estado, $u(t) = [u_1(t), \ldots, u_p(t)]^T$ é o vetor de entrada formado por p entradas como seus elementos, $z(t) = [z_1(t), \ldots, z_m(t)]^T$ é o vetor de saída formado por m saídas como seus elementos, $A \in \mathbb{R}^{n \times n}$ é a matriz de estados, $B \in \mathbb{R}^{n \times p}$ é a matriz de coeficientes de entrada e $C \in \mathbb{R}^{m \times n}$ é a matriz de coeficientes de saída.

Considerando a lei de controle dada pela realimentação de estados

$$u(t) = Kx(t) \tag{3.8}$$

então temos o seguinte sistema em malha fechada com realimentação de estados (*state feedback control*):

$$\begin{cases} \dot{x}(t) = (A + BK)x(t) \\ z(t) = Cx(t) \end{cases} \tag{3.9}$$

A realimentação de estados pressupõe que todas as variáveis de estado estejam disponíveis para a realimentação e que o sistema seja controlável. Na prática, isto pode não ser verdade porque as variáveis de estado podem não estar disponíveis para medição direta ou a medição pode ser cara demais.

Nos casos práticos, a medição da saída é normalmente mais fácil, levando à realimentação de saída. No entanto, o sistema deve ser observável para que seja possível a construção de observadores de estado, os quais geram uma estimativa deste [56].

No caso do Modelo Base (3.1), como uma variável de estado $x_i(t)$ representa o tempo de espera estimado experimentado por uma tarefa inserida em uma fila do i-ésimo nó, ela pode ser obtida através da equação $x_i(t) = q_i(t)t_{p_i}$ - seção (2.2).

Considerando a lei de controle dada pela realimentação de saída

$$u(t) = Kz(t) \tag{3.10}$$

então temos o seguinte sistema em malha fechada com realimentação de saída (*output feedback control*):

$$\begin{cases} \dot{x}(t) = (A + BKC)x(t) \\ z(t) = Cx(t) \end{cases} \tag{3.11}$$

Vamos mostrar que a versão linear sem atraso descrito na variável x do Modelo Base (3.6) é um caso particular do modelo em malha fechada com realimentação de saída (3.11), e que podemos colocá-lo na forma de um modelo em malha fechada com realimentação de estados. Esta mudança será útil quando analisarmos a versão não linear com atraso descrita na variável y.

Vamos assumir as seguintes hipóteses para o Modelo Base (3.6):

- A matriz de ganhos não mais será do tipo diagonal;
- O termo não-linear $v(y(t)) = -K_d\phi(y(t))$ será substituído por $-Ky(t)$;
- O termo não-linear $v(y(t-h)) = -K_d\phi(y(t-h))$ será substituído por $-Ky(t)$.

Assumindo estas hipóteses, é possível expressar o modelo na variável x ou na variável y. Esta última, se mostrará útil quando analisarmos a versão do modelo não-linear com atraso.

3.1.1.1 Versão Linear Sem Atraso Descrito na Variável x

Considerando as hipóteses feitas e após as devidas substituições, podemos apresentar o modelo versão linear sem atraso descrito na variável x, ou seja, partindo do modelo:

$$\dot{x}(t) = -Ky(t) + TKy(t) + \lambda \tag{3.12}$$

e da equação (2.4), podemos extrair a seguinte relação entre as variáveis x e y:

$$y(t) = Rx(t) \tag{3.13}$$

onde

$$R = \frac{1}{n}\begin{bmatrix} (n-1) & -1 & \cdots & -1 \\ -1 & (n-1) & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & (n-1) \end{bmatrix}_{n\times n} \tag{3.14}$$

$R \in \mathbb{R}^{n\times n}$ é estruturalmente singular.

Substituindo (3.13) em (3.12), teremos:

$$\dot{x}(t) = -KRx(t) + TKRx(t) + \lambda \tag{3.15}$$

e colocando $KRx(t)$ em evidência, teremos:

$$\begin{cases} \dot{x}(t) = (-I + T)KRx(t) + \lambda \\ y(t) = Rx(t) \end{cases} \tag{3.16}$$

onde I é a matriz identidade $n \times n$.

A Figura 3.2 mostra o diagrama em blocos da versão linear na variável x sem atraso para a forma (3.16).

Comparando o modelo (3.16) com (3.11), podemos verificar que a matriz A em (3.16) é nula (todos os pólos na origem), a matriz B de (3.16) é da da por $B = (-I + T)$, a matriz

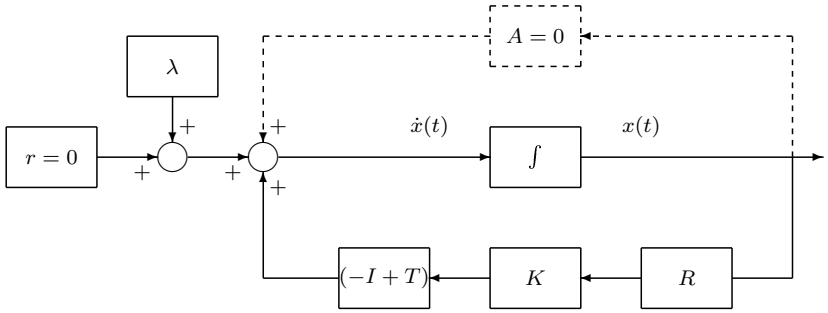

Figura 3.2: Diagrama em blocos da versão linear na variável x sem atraso

$C = R$ e $y(t) = z(t)$. Este modelo em malha fechada (3.16) é resultado da seguinte lei de controle por realimentação de saída:

$$u(t) = KRx(t) \tag{3.17}$$

Podemos colocar o modelo (3.16) na forma apresentada em (3.11) se multiplicarmos (3.12) à esquerda pela matriz R, resultando na versão linear na variável y, descrita a seguir.

3.1.1.2 Versão Linear Sem Atraso Descrito na Variável y

Multiplicando-se a equação (3.12) pela matriz R à esquerda em ambos os lados da equação (3.18), teremos:

$$R\dot{x}(t) = -RKy(t) + RTKy(t) + R\lambda \tag{3.18}$$

Como $\dot{y}(t) = R\dot{x}(t)$, a equação fica da seguinte forma:

$$\dot{y}(t) = -RKy(t) + RTKy(t) + R\lambda \tag{3.19}$$

Colocando $y(t)$ em evidência:

$$\begin{cases} \dot{y}(t) = R(-I + T)Ky(t) + R\lambda \\ z(t) = Cy(t) \end{cases} \tag{3.20}$$

onde a matriz C é a identidade, neste caso. A equação (3.20) descreve um sistema linear invariante no tempo em malha fechada na variável y, onde a matriz A de (3.20) é nula, a matriz B de (3.20) é dada por $B = R(-I + T)$, e a lei de controle por realimentação de

estados é dada por:

$$u(t) = Ky(t) \tag{3.21}$$

Neste caso, a matriz B será singular e o estado deste modelo, dado por $y(t) = [y_1(t), \ldots, y_n(t)]^T$, representa a diferença entre o tempo médio esperado e a média local estimada - ver equação (3.13) - que, sob leis de controle adequadas (ganhos K), deverá convergir para zero (*zeroing the output control*).

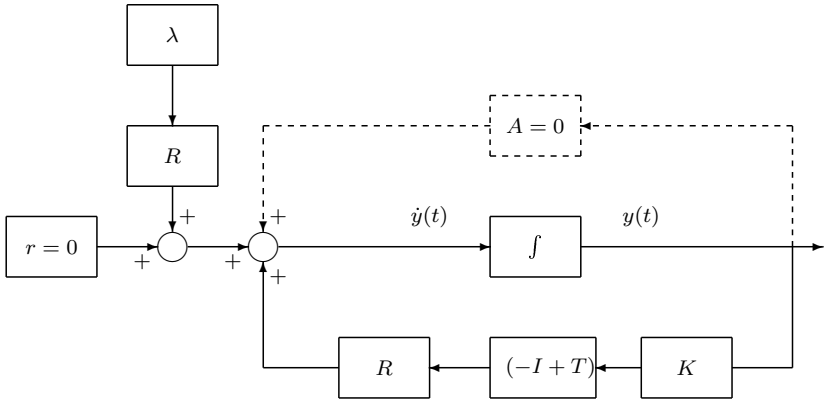

Figura 3.3: Diagrama em blocos da versão linear na variável y sem atraso

O estudo do modelo versão linear na variável y sem atraso é importante, pois ele fornecerá subsídios para o desempenho do modelo em malha fechada na síntese de controladores na forma de redes neurais com atraso, como será visto na seção (5.3).

A Figura 3.3 mostra o diagrama em blocos da versão linear na variável y sem atraso.

3.1.2 Versão Não-Linear sem Atraso

Vamos considerar a função não-linear $\phi(y(t)) = uhsat(y(t))$ e descrever os modelos correspondentes nas variáveis x e y.

3.1.2.1 Versão Não-Linear Sem Atraso Descrito na Variável x

Retomando a equação (3.6), assumindo o atraso $h = 0$, e substituindo o vetor $v(y(t)) = -K\phi(y(t))$, teremos:

$$\dot{x}(t) = -K\phi(y(t)) + TK\phi(y(t)) + \lambda \tag{3.22}$$

Colocando os termos $K\phi(y(t))$ em evidência, teremos:

$$\dot{x}(t) = (-I + T)K\phi(y(t)) + \lambda \tag{3.23}$$

Este modelo é o que, na verdade, mais se aproxima do modelo hidrodinâmico de Hui pois não considera atraso de transferência nem atraso de comunicação.

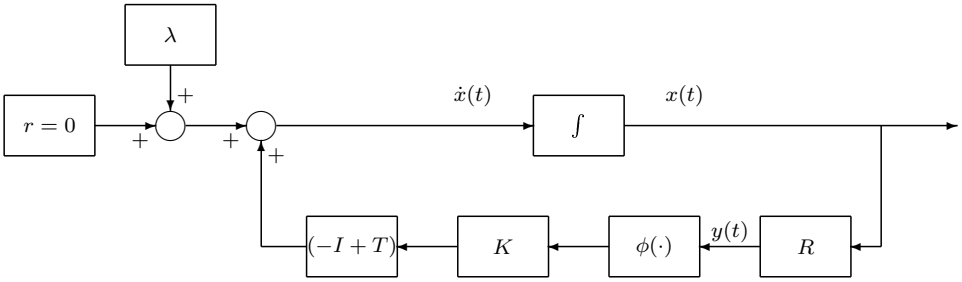

Figura 3.4: Diagrama em blocos da versão não-linear na variável x sem atraso

A Figura 3.4 apresenta o diagrama em blocos da versão não-linear na variável x sem atraso.

3.1.2.2 Versão Não-Linear Sem Atraso Descrito na Variável y

Multiplicando-se a equação (3.23) pela matriz R pela esquerda, teremos:

$$\dot{y}(t) = R(-I + T)K\phi(y(t)) + R\lambda \tag{3.24}$$

a qual acaba na forma do modelo de rede neural de Hopfield.

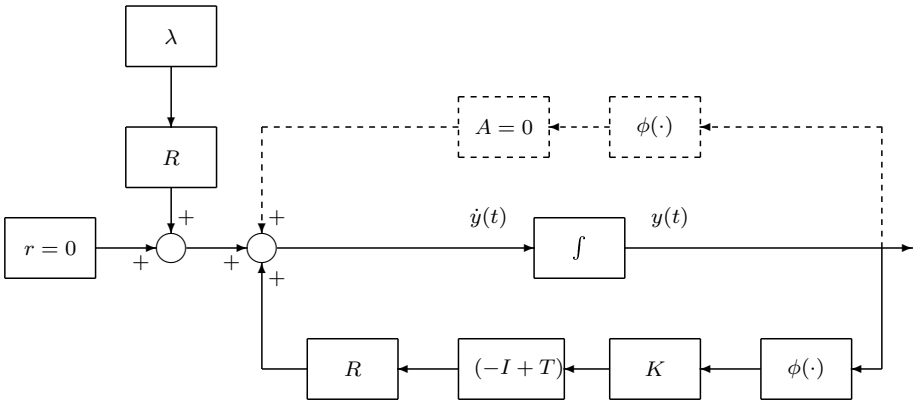

Figura 3.5: Diagrama em blocos da versão não-linear na variável y sem atraso

Se desconsiderarmos o termo $R\lambda$, ou seja sem a entrada de tarefas, então a equação (3.24) assume a forma de um sistema Persidskii, definido por [52, 55]:

Definição 3.1 (Sistemas Persidskii)**.** *Seja um sistema dinâmico* $\dot{y} = \phi(y)$, $y \in \mathbb{R}^n, \phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ *é dito ser do tipo Persidskii se ele for da forma*

$$\dot{y}_i(t) = \sum_{j=1}^{n} a_{ij}\phi_j(y_j(t)), \quad i = 1, 2, \ldots, n \tag{3.25}$$

onde para cada cada $i, j, \phi_j : \mathbb{R} \mapsto \mathbb{R}$ *pertence à classe de funções de setor infinito:*

- $\psi\phi_j(\psi) \geq 0$ *e* $\phi_j(0) = 0$*;*
- $\int_0^{y_j} \phi_j(\tau)d\tau \mapsto \infty$*, quando* $|y_j| \mapsto \infty$*;*
- ϕ_j *é Lipschitz contínua.*

O sistema dinâmico (3.25) também pode ser escrito na forma compacta como se segue: $\dot{y}(t) = A\phi(y)$, onde $A = (a_{ij})$ e $\phi(y) = [\phi_1(y_1), \ldots, \phi_n(y_n)]^T$.

Como será mostrado no capítulo 4, na literatura existem condições de estabilidade global assintótica com base nas características da matriz A para sistemas desta classe.

A Figura 3.5 apresenta o diagrama em blocos da versão não-linear em y sem atraso.

3.2 Versões do Modelo com Atraso (Linear e Não-Linear)

Após termos descrito as versões de modelos sem atraso, passaremos a representá-los considerando-se um atraso homogêneo (atraso único) nos modelos.

3.2.1 Versão Linear com Atraso

Conforme a seção anterior e suas subseções, vamos continuar seguindo a mesma seqüência apresentada de modelos, conforme a Figura 3.1.

3.2.1.1 Versão Linear com Atraso Descrito na Variável x

Partindo da equação (3.6) e assumindo as hipóteses dadas anterioremente para a versão linear, teremos:

$$\dot{x}(t) = -Ky(t) + TKy(t-h) + \lambda \tag{3.26}$$

Como $y(t) = Rx(t)$ - equação (3.13), podemos substituir esta relação na equação acima, obtendo

$$\dot{x}(t) = -KRx(t) + TKRx(t-h) + \lambda \tag{3.27}$$

Temos então um sistema linear do tipo $\dot{x}(t) = Wx(t) + W_h x(t-h)$, onde $W = -KR$ e $W_h = TKR$. As propriedades destes sistemas foram estudadas em [57, 58], e com atraso variável em [59, 60].

Podemos notar que a matriz de conversão de tempo de espera entre os nós - matriz T (2.9) multiplica apenas o segundo termo do lado direito da equação (3.27).

Esta matriz T é formada pela razão dos tempos de espera em cada fila (t_{p_i}/t_{p_j}). Estes tempos de espera, por sua vez, dependem de forma inversamente proporcional à velocidade do processador de um determinado nó. Quanto maior a diferença de capacidade entre os processadores, o número de condicionamento da matriz T torna-se maior. Isto acontece porque para um valor t_{p_i} alto e um valor t_{p_j} baixo, o elemento $T_{ij} = t_{p_i}/t_{p_j}$ da matriz T será alto, enquanto o elemento $T_{ji} = t_{p_j}/t_{p_i}$ será baixo.

Observa-se que no caso das versões de modelos sem atraso, ocorre uma soma das matrizes I e T, e a diagonal principal da matriz T, que originalmente era nula conforme a equação (2.9), passa a ser negativa e unitária.

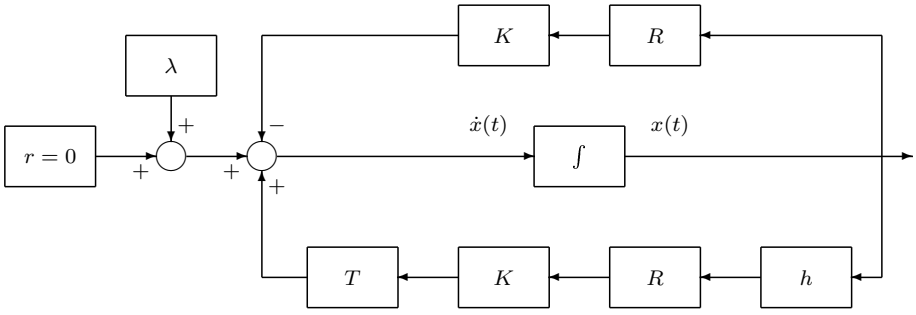

Figura 3.6: Diagrama em blocos da versão linear na variável x com atraso

A Figura 3.6 apresenta o diagrama de blocos da versão linear na variável x com atraso.

3.2.1.2 Versão Linear com Atraso Descrito na Variável y

O modelo da versão linear na variável y com atraso é importante, pois ele fornece subsídios para a síntese de controladores na forma de redes neurais com atraso conforme capítulo (5), além de esclarecer a seqüência lógica do desenvolvimento realizado neste trabalho.

Partindo da equação (3.26), multiplicando-se os dois membros pela matriz R pela esquerda, teremos:

$$\dot{y}(t) = -RKy(t) + RTKy(t-h) + R\lambda \tag{3.28}$$

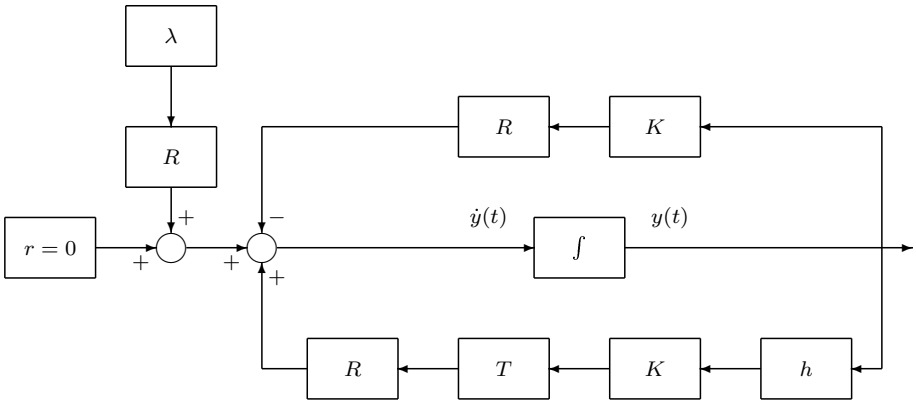

Figura 3.7: Diagrama em blocos da versão linear na variável y com atraso

A Figura 3.7 apresenta o diagrama de blocos da versão linear na variável y com atraso.

3.2.2 Versão Não-Linear com Atraso

Vamos assumir agora as funções $\phi(y(t)) = uhsat(y(t))$ e apresentar as versões dos modelos não-lineares com atrasos descritos nas variáveis x e y.

3.2.2.1 Versão Não-Linear com Atraso Descrito na Variável x

A versão do modelo não-linear com atraso, descrito na variável x, constitui-se no próprio Modelo Base (3.6), com as funções $v(y(t))$ e $v(y(t-h))$ substituídas. A equação, desta forma pode ser apresentada como:

$$\dot{x}(t) = -K\phi(y(t)) + TK\phi(y(t-h)) + \lambda \tag{3.29}$$

Podemos notar que esta equação é similar ao modelo do Hopfield-Tank com atrasos homogêneos, com a ressalva das diferenças já citadas na seção Analogia entre os modelos do capítulo 2.

De forma a eliminar as diferenças entre os termos de realimentação linear e não-linear entre o Modelo Base (3.6) e o modelo de Hopfield-Tank com atraso, poderemos multiplicar a equação acima pela matriz R à esquerda e reescrevê-la na variável y.

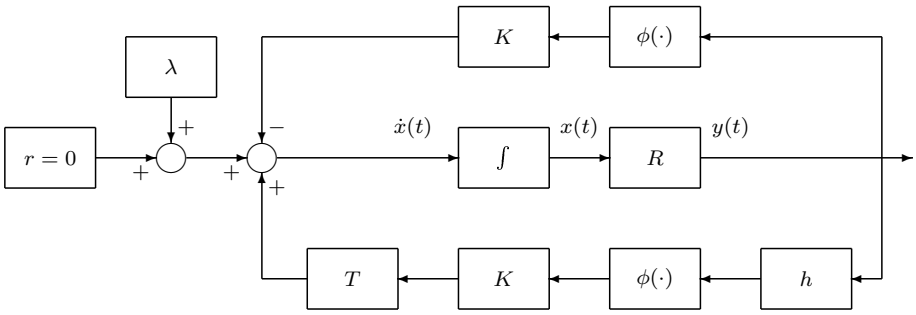

Figura 3.8: Diagrama em blocos da versão não-linear na variável x com atraso

A Figura 3.8 apresenta o diagrama de blocos da versão linear na variável x com atraso.

3.2.2.2 Versão Não-Linear com Atraso Descrito na Variável y

Como feito nos casos anteriores, a partir da versão não-linear com atraso, descrita na variável x, podemos multiplicar a equação (3.29) pela matriz R pela esquerda em ambos os membros, e obter uma versão não-linear com atraso descrita na variável y. Desta forma, colocamos nosso Modelo Base (3.6) em uma forma particular de rede neural de Hopfield-Tank, que é o modelo final com o qual trabalharemos visando obter as leis de controle adequadas. Doravante chamaremos este modelo de **Modelo de Rede Neural** ou **MRN**.

$$\dot{y}(t) = -RK\phi(y(t)) + RTK\phi(y(t-h)) + R\lambda \tag{3.30}$$

Observa-se que o Modelo de Rede Neural que representa o problema de balanceamento de carga não possui a realimentação linear presente no modelo de Hopfield-Tank com atrasos dado em (2.30).

A ausência do termo de realimentação linear dificulta a ação estabilizante dos controladores, já que o termo linear com realimentação negativa contribui para a estabilização do sistema, enquanto os atrasos contribuem para a instabilização.

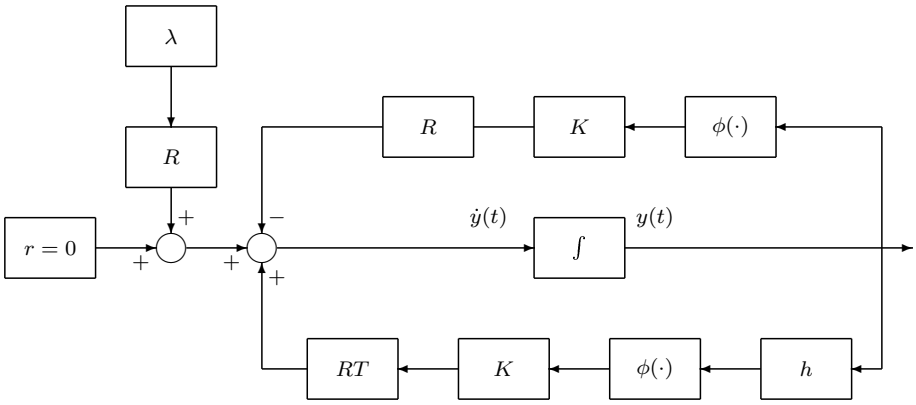

Figura 3.9: Diagrama em blocos do modelo de rede neural com atrasos do ponto de vista da Teoria de Controle

A Figura 3.9 apresenta o diagrama de blocos do Modelo de Rede Neural que representa o problema de balanceamento de carga.

3.3 Outra Forma de Representação do Modelo de Chiasson *et al.*

Uma outra forma de reescrever o modelo (2.1) incluindo os parâmetros $\mu(x)$ e λ foi estudada no início de todo o trabalho e será apresentada nesta seção.

Esta forma de representação apresentava, em seu início, a vantagem de incluir o termo $\mu(x)$ (consumo de tarefas) em uma forma matricial semelhante à do modelo de Hopfield-Tank, porém mostrou-se desnecessária a partir de [30], onde mostrou-se que não há sentido em termos consumo de tarefas, representado pelo termo $\mu(x)$ simultaneamente ao balanço de carga pois ou o processador está processando alguma tarefa (consumindo) ou realizando o esforço necessário para executar o balanço de carga.

Vamos considerar um cluster apenas com dois processadores, ou seja, $n = 2$, inicialmente sem atraso. Partindo da equação (2.4), e permitindo um sistema com ordem estendida, temos:

$$\begin{cases} y_1(t) = x_1(t) - \frac{x_1(t)+x_2(t)}{2} \\ y_2(t) = x_2(t) - \frac{x_1(t)+x_2(t)}{2} \\ y_3(t) = x_1(t) \\ y_4(t) = x_2(t) \end{cases} \tag{3.31}$$

Derivando $y_i(t)$ em relação ao tempo, teremos que:

$$\begin{cases} \dot{y}_1(t) = \dot{x}_1(t) - \frac{\dot{x}_1(t)+\dot{x}_2(t)}{2} \\ \dot{y}_2(t) = \dot{x}_2(t) - \frac{\dot{x}_1(t)+\dot{x}_2(t)}{2} \\ \dot{y}_3(t) = \dot{x}_1(t) \\ \dot{y}_4(t) = \dot{x}_2(t) \end{cases} \tag{3.32}$$

Fazendo as substituições necessárias no sistema da equação (3.32) a partir da equação (2.1), e reescrevendo na forma matricial, teremos:

$$y(t) = \frac{1}{2}\begin{bmatrix} -1-p\frac{t_{p_2}}{t_{p_1}} & 1+p\frac{t_{p_2}}{t_{p_1}} & -2 & 2 \\ 1+p\frac{t_{p_2}}{t_{p_1}} & -1-p\frac{t_{p_2}}{t_{p_1}} & 2 & -2 \\ -2 & 2p\frac{t_{p_1}}{t_{p_2}} & -2 & 0 \\ 2p\frac{t_{p_2}}{t_{p_1}} & -2 & 0 & -2 \end{bmatrix}\begin{bmatrix} K_1 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \\ \mu(y_3(t)) \\ \mu(y_4(t)) \end{bmatrix} + \tag{3.33}$$

$$+\frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}$$

No entanto, esta representação (3.33) apresenta uma desvantagem para sistemas de grande porte, pois é necessário trabalharmos com matrizes de ordem até $2n \times 2n$.

Apesar da mesma forma de representação matricial também poder ser feita para o modelo (2.1) com atraso, a representação matricial dada pelo **Modelo Base** na forma compacta (3.6) torna-se mais vantajosa por proporcionar matrizes de ordem até $n \times n$, justificando assim a escolha por (3.6).

3.4 Resumo do Capítulo

Neste capítulo, partimos do Modelo Base (3.6) e apresentamos diversas versões deste modelo. Cada um dos modelos assumindo uma ou mais particularidades.

A comparação do modelo versão não-linear com atraso descrito na variável y (3.30) com o modelo de Hopfield-Tank com atrasos (2.30) evidencia que o modelo (3.30) é um caso particular de (2.30), uma vez que o modelo (3.30) difere deste último apenas por não apresentar o termo de realimentação linear negativa.

A descoberta da possibilidade de representarmos o modelo (2.1) como um caso particular do modelo de redes neurais com atraso **HDNN** (2.30), resultando no **MRN** (3.30), nos permitirá a busca de controladores, com a devida prova de convergência, através de trabalhos publicados na área de redes neurais com atraso, principalmente utilizando-se inequações matriciais lineares que, devido ao *Método dos Pontos Interiores*, nos permite solucionar problemas com um custo computacional atraente.

Este caminho, realizado através da apresentação destes modelos, teve como propósito estabelecer as equações que poderão reger os casos particulares, mesmo que alguns destes casos sejam difíceis de serem encontrados na prática, como por exemplo, um cluster realmente homogêneo com uma taxa média de processamento de tarefas igual para todos os nós. Ainda assim, estas versões permitiram um melhor entendimento do problema e ajudou na definição do caminho a ser seguido.

Capítulo 4

Resultados de Estabilidade Para os Diferentes Modelos

Neste capítulo, faremos um estudo de análise de estabilidade para alguns dos modelos desenvolvidos no capítulo 3, de forma a permitir um melhor entendimento de comportamentos particulares de um modelo mais geral, que é o da rede neural com atraso (**MRN**).

O estudo das condições de estabilidade, principalmente quando elas são apresentadas na forma de LMIs, auxiliará na determinação das matrizes de ganho de realimentação de uma forma computacionalmente eficiente através do *Método dos Pontos Interiores*.

4.1 Versões do Modelo sem Atraso (Linear e Não-Linear)

Nesta seção apresentaremos resultados de estabilidade considerando-se os modelos sem atrasos. Com isto, espera-se facilitar a captação do comportamento do modelo nesta situação hipotética, o que permitirá uma comparação para os mesmos sistemas com atraso quando estes forem incluídos mais adiante.

4.1.1 Versão Linear sem Atraso

A fim de facilitar o entendimento do comportamento do sistema, vamos explicitar as condições de estabilidade para as versões lineares dos sistemas sem atrasos.

4.1.1.1 Versão Linear sem Atraso Descrito na Variável x

Seja o sistema linear e invariante no tempo (SLIT) dado pela equação (3.7), reproduzida parcialmente aqui por conveniência:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.1}$$

considerando a lei de controle:

$$u(t) = KRx(t) \tag{4.2}$$

então, em malha fechada, o SLIT (3.7) possui a seguinte forma:

$$\dot{x}(t) = (\underbrace{A + BKR}_{A_{mf}})x(t) \tag{4.3}$$

Seja a equação da versão linear sem atraso descrita na variável x (3.16), reproduzida aqui por conveniência :

$$\dot{x}(t) = (\underbrace{(-I + T)KR}_{A_{mf}})x(t) + \lambda \tag{4.4}$$

temos que, neste caso, a matriz A_{mf} é dada por $A_{mf} = (-I + T)KR$. Comparando as equações (4.3) e (4.4), verificamos que A é nula e $B = (-I + T)$ nesta última. Vale observar ainda que, como não temos controle sob o termo λ no modelo, ele deve ser visto como uma perturbação e não uma entrada do sistema.

Devido ao fato da matriz A ser nula, técnicas de síntese de controladores clássicas, como por exemplo, posicionamento de pólos (fórmula de Ackermann) não são aplicáveis nesse caso, pois o modelo falha nos testes de controlabilidade e observabilidade.

Uma forma de contornar este problema é inserindo uma pequena perturbação dada por uma matriz $A = -\varepsilon I$, onde I é a matriz identidade e ε é uma constante positiva suficientemente pequena ($\varepsilon \approx 0$), de modo a mover os pólos da origem para o semi-plano lateral esquerdo do plano complexo.

Um sistema linear invariante no tempo será estável quando o teorema de Lyapunov, citado abaixo, for satisfeito:

Teorema 4.1 (Teorema de Lyapunov)**.** *[61]*

Uma condição necessária e suficiente para um sistema linear invariante no tempo $\dot{x}(t) = A_{mf}x(t)$ *ser estritamente estável é que, para qualquer matriz definida positiva* Q*, a única matriz* P *solução da equação de Lyapunov*

$$A_{mf}^T P + P A_{mf} = -Q \tag{4.5}$$

seja simétrica definida positiva.

Como exemplo, podemos verificar a estabilidade de (4.4) para os mesmos ganhos e parâmetros dados em [33] - $t_{p_1} = t_{p_2} = t_{p_3} = 10\mu s$, e matriz de ganhos K dada por:

$$K = \begin{bmatrix} 6667 & 0 & 0 \\ 0 & 4167 & 0 \\ 0 & 0 & 5000 \end{bmatrix} \tag{4.6}$$

Substituindo os valores das matrizes I, K, T e R em (4.4), teremos:

$$\dot{x}(t) = \underbrace{\begin{bmatrix} -5972,5 & 2778,0 & 3194,5 \\ 2778,0 & -4722,5 & 1944,5 \\ 3194,5 & 1944,5 & -5139,0 \end{bmatrix}}_{A_{mf}} x(t) + \lambda \tag{4.7}$$

Os autovalores da matriz A_{mf} são $[-9,01.10^3 \quad -6,81.10^3 \quad -1,12.10^{-12}]^T$.

Utilizando o script Matlab[1] *hdstab.m* [52] (chamado pelo script Lyapunov.m) para resolver a equação (4.5), é possível verificar que o sistema é estável de acordo com o Teorema de Lyapunov, conforme mostra a Figura 4.1, onde a matriz P encontrada é a própria identidade para este exemplo.

4.1.1.2 Versão Linear sem Atraso Descrito na Variável y

Vamos retomar a equação do sistema linear invariante no tempo (3.20), reproduzida aqui por conveniência:

$$\dot{y}(t) = R(-I + T)Ky(t) + R\lambda \tag{4.8}$$

[1] Software matemático da Mathworks Inc.

onde $B = R(-I + T)$.

Como a matriz R, que é estruturalmente singular, multiplica o termo $(-I + T)$, a matriz B será singular [62].

Para analisarmos a estabilidade do modelo (4.8), vamos considerar o termo $R\lambda$ sendo nulo e concentrar-nos no restante da equação, que representa a troca de carga.

Para satisfazer a condição de equilíbrio $\dot{y}(t) = 0$, temos então

$$BKy(t) = 0 \tag{4.9}$$

No entanto, se $y(t)$ estiver no espaço nulo da matriz BK, então a solução trivial de equilíbrio $\lim_{t\to\infty} y(t) = 0$ para o modelo (4.8) não será a única.

Unicidade do Ponto de Equilíbrio

Para provar a unicidade do ponto de equilíbrio do modelo (4.8), convém analisarmos a equação (3.13), reproduzida aqui por conveniência:

$$y(t) = Rx(t) \tag{4.10}$$

que nos fornece uma restrição sobre o modelo. As componentes do vetor $y(t)$ não podem assumir quaisquer valores.

Devido ao fato de uma das linhas da matriz R ser combinação linear das outras $(n-1)$ linhas restantes, então uma das componentes do vetor $y(t)$ será a combinação linear das $(n-1)$ componentes restantes, ou seja:

$$\sum_{i=1}^{n} y_i(t) = 0 \tag{4.11}$$

ou, na forma compacta:

$$c^T y(t) = 0 \tag{4.12}$$

onde $c^T = [1 \quad 1 \quad \cdots \quad 1]$.

Por exemplo, a equação (3.13) para $n = 2$:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \Rightarrow y_1 + y_2 = 0 \tag{4.13}$$

e para o caso $n = 3$:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \Rightarrow y_1 + y_2 + y_3 = 0 \tag{4.14}$$

Seja o sistema de equações:

$$\begin{bmatrix} B \\ c^T \end{bmatrix}_{(n+1)\times n} y(t) = 0 \tag{4.15}$$

Como a matriz B é singular, uma de suas linhas é linearmente dependente das outras $(n-1)$ linhas restantes. Logo, podemos substituir a linha linearmente dependente da matriz B pela linha dada pelo vetor c^T, reduzindo a ordem do sistema de equações (4.15) e obtendo:

$$Z = \begin{bmatrix} & B^* & \\ - & - & - \\ 1 & \cdots & 1 \end{bmatrix}_{n\times n} \tag{4.16}$$

onde B^* é a matriz B sem a sua linha linearmente dependente.

Utilizando o *toolbox de matemática simbólica* do Matlab, podemos chegar à seguinte expressão geral para o determinante da matriz Z:

$$det(Z) = \kappa\frac{(\sum_{i=1}^{n} t_{p_i})(\sum_{j=1}^{n}\prod_{i=1}^{n-1} t_{p_i})}{\prod_{i=1}^{n} t_{p_i}} \tag{4.17}$$

onde $\kappa > 0$. Como $t_{p_i} > 0, i = 1, \ldots, n$, temos que $det(Z) > 0$.

Logo,

$$\begin{bmatrix} B^* \\ c^T \end{bmatrix} y = 0, \text{ se, e somente se, } y = 0 \tag{4.18}$$

provando que o modelo (4.8) possui um único ponto de equilíbrio que localiza-se na origem.

Prova de Estabilidade

Realizada a prova da unicidade do ponto de equilíbrio de (4.8), vamos provar que este ponto de equilíbrio em $y^* = 0$ é estável.

Seja a função de Lyapunov para $n = 2$:

$$V(y_1, y_2) = y^T(t)Py(t) \tag{4.19}$$

onde $y(t) = [y_1(t), y_2(t)]^T$. A sua derivada no tempo

$$\dot{V}(y_1, y_2) = y^T(t)(A^T P + PA)y(t) \tag{4.20}$$

forma uma calha invertida no plano (y_1, y_2) - Figura 4.2. No entanto, o par (y_1, y_2) não pode assumir quaisquer valores in $\mathbb{R}$, pois a relação (3.13), quando aplicada neste caso:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \tag{4.21}$$

implica em $y_1(t) + y_2(t) = 0$. A intersecção do plano vertical na Figura 4.2 com a calha invertida, fornece a curva de $\dot{V}(y_1, y_2)$, que só se anula na origem. Ou seja, $\dot{V}(y_1, y_2) < 0, \forall (y_1, y_2) \neq 0, \quad y_1 + y_2 = 0$.

Esta análise pode ser facilmente estendida para n qualquer.

4.1.2 Versão Não-Linear sem Atraso

Vamos rever agora os mesmos modelos anteriores, porém introduzindo a não-linearidade $\phi(y_i(t)) = uhsat(y_i(t))$ e checar as condições de estabilidade para estes casos.

4.1.2.1 Versão Não-Linear sem Atraso Descrito na Variável x

Retomando a equação (3.23), reproduzida aqui por conveniência:

$$\dot{x}(t) = (-I + T)K\phi(y(t)) + \lambda \tag{4.22}$$

podemos notar que o mesmo assemelha-se a um sistema do tipo Persidskii, mas que ainda não está na sua forma original. No entanto, se multiplicarmos a matriz R por ambos os lados

da equação acima e pela esquerda, então poderemos colocar o sistema na forma Persidskii, o qual possui condição de estabilidade conhecida [52]. A próxima subseção tratará deste sistema no formato adequado para a classe de funções Persidskii.

4.1.2.2 Versão Não-Linear sem Atraso Descrito na Variável y

Retomando a equação (3.24), reproduzida aqui por conveniência:

$$\dot{y}(t) = R(-I + T)K\phi(y(t)) + R\lambda \tag{4.23}$$

temos que o sistema pertence à classe Persidskii com um entrada constante $R\lambda$, pois além da equação (4.23) estar na forma compacta de (3.25), o termo não-linear $\phi(y(t))$ de (4.23) satisfaz as condições necessárias previstas na definição da classe - Ver definição 3.1.

Unicidade do Ponto de Equilíbrio

Deslocando o ponto de equilíbrio $y^* = [y_1^*, \ldots, y_n^*]^T$ de (4.23) para a origem e usando a relação $z(\cdot) = y(\cdot) - y^*$, a equação (4.23) é transformada em

$$\dot{z}(t) = R(-I + T)K\tilde{\phi}(z(t)) \tag{4.24}$$

onde $z(\cdot) = [z_1(\cdot), \ldots, z_n(\cdot)]^T$ é o novo vetor de estados, $\tilde{\phi}(z(\cdot)) = [\tilde{\phi}_1(z_1(\cdot)), \ldots, \tilde{\phi}_n(z_n(\cdot))]^T$ representa o vetor de funções não-lineares do modelo transformado, e $\tilde{\phi}_i(z_i(\cdot)) = \phi_i(z_i(\cdot) + y_i^*) - \phi_i(y_i^*), i = 1, 2, \ldots, n$.

As funções $\tilde{\phi}_i(z_i(\cdot))$ também satisfazem:

$$\tilde{\phi}_i^2(z_i(\cdot)) \leq z_i(\cdot)\tilde{\phi}_i(z_i(\cdot)), \quad \tilde{\phi}_i(0) = 0, \quad i = 1, 2, \ldots, n. \tag{4.25}$$

Para provar a unicidade do ponto de equilíbrio do sistema (4.24), vamos lembrar que, uma vez que ele esteja em equilíbrio, então $\dot{z} = 0$, logo:

$$R(-I + T)K\tilde{\phi}(z(t)) = 0 \tag{4.26}$$

Se o termo $(-I + T)K\tilde{\phi}(z(t))$ da equação acima estiver no espaço nulo $\mathcal{N}(R)$ da matriz R, então a solução trivial $z = 0$ não será o único ponto de equilíbrio do sistema.

Vamos analisar o termo $(-I+T)K\tilde{\phi}(z(t))$ para $n=2$, já considerando uma matriz K não-diagonal:

$$\begin{bmatrix} -1 & \frac{t_{p_1}}{t_{p_2}} \\ \frac{t_{p_2}}{t_{p_1}} & -1 \end{bmatrix} \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} \tilde{\phi}_1(z(t)) \\ \tilde{\phi}_2(z(t)) \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.27}$$

Multiplicando as matrizes acima e, representando $\tilde{\phi}_j(z(t))$ como $\tilde{\phi}_j$ apenas para facilitar a notação, temos:

$$\begin{bmatrix} -\sum_{j=1}^{2} k_{1j}\tilde{\phi}_j + \frac{t_{p_1}}{t_{p_2}} \sum_{j=1}^{2} k_{2j}\tilde{\phi}_j \\ \frac{t_{p_2}}{t_{p_1}} \sum_{j=1}^{2} k_{1j}\tilde{\phi}_j - \sum_{j=1}^{2} k_{2j}\tilde{\phi}_j \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.28}$$

A equação matricial acima nos mostra que a primeira linha da matriz à esquerda deve ser igual a sua segunda linha. No entanto, para que isto seja verdadeiro, e para $\tilde{\phi}_j \neq 0$ (solução não-trivial), é necessário que:

$$\frac{t_{p_1}}{t_{p_2}} = -1 \quad \text{e} \quad \frac{t_{p_2}}{t_{p_1}} = -1 \tag{4.29}$$

o que viola a condição do modelo (2.1), que diz que $t_{p_i} > 0$. Logo, por contradição, prova-se que o ponto de equilíbrio é único. O mesmo raciocínio pode ser aplicado para n qaulquer, obtendo-se invariavelmente a mesma conclusão.

Estabilidade do Ponto de Equilíbrio

Sendo assim, poderemos aplicar o Teorema de Persidskii a fim de assegurar a estabilidade de uma solução para este modelo. Mas antes, vamos a algumas definições importantes:

Definição 4.1. *A classe de funções $\mathcal{S}_c$ é definida como*

$$\mathcal{S}_c = \{\phi(\cdot)|\phi : \mathbb{R} \mapsto \mathbb{R}; \xi\phi(\xi) > 0; \phi(0) = 0 \tag{4.30}$$

e

$$\int_0^y \phi(\tau)d\tau \to \infty \textit{ quando } |y| \to \infty\} \tag{4.31}$$

Esta classe também é conhecida como a classe de funções de não-linearidades positivas de setor infinito.

Definição 4.2. *A classe de funções $\mathcal{S}_c^l$ é a classe formada pelo produto Cartesiano dos fatores idênticos $\mathcal{S}_c$.*

Definição 4.3 (Matriz Hurwitz Diagonalmente Estável)**.** *Uma matriz A é dita Hurwitz Diagonalmente Estável se existir uma matriz diagonal P positiva definida cuja solução da equação:*

$$A^T P + PA \tag{4.32}$$

seja negativa definida.

A classe de matrizes Hurwitz Diagonalmente Estáveis é representada por $\mathcal{D}_c$.

Agora é possível enunciar o Teorema de Estabilidade de Persidskii [52]:

Teorema 4.2 (Teorema de Estabilidade de Persidskii)**.** *A solução $y^* = 0$ do sistema não-linear (3.25) é globalmente assintoticamente estável para todo $\phi(\cdot) \in \mathcal{S}_c^l$ se $A \in \mathcal{D}_c(P)$. De forma equivalente, para cada membro da classe de sistemas não-lineares $\{\dot{y} = A\phi(y), y(0) = y_0 : \phi \in \mathcal{S}_c^n\}$, a função definida positiva*

$$V(y) = 1/2 \sum_{i=1}^{n} p_i \int_0^{y_i} \phi_i(\tau) d\tau \tag{4.33}$$

é uma função de Lyapunov diagonal, estabelecendo estabilidade assintótica global da solução $y^* = 0$ do sistema correspondente.

O uso da formulação de Persidskii é justificado pelo fato de as funções de Lyapunov associadas aos sistemas Persidskii serem bastante conhecidas.

Resultados para a estabilidade de sistemas Persidiskii com funções não-lineares descontínuas podem ser encontradas em [53, 63, 64].

4.2 Versões do Modelo com Atraso (Linear e Não-Linear)

Iniciaremos com o estudo de algumas versões lineares do Modelo Base com atraso e depois estenderemos este assunto aos casos não-lineares. A última versão a ser estudada neste capítulo será justamente a versão do Modelo de Rede Neural com atraso, que é o caso de referência que nos interessa para o problema de balanceamento de carga (Modelo Base).

4.2.1 Versão Linear com Atrasos

A classe de sistemas lineares com atraso vêm sendo estudada há algum tempo, como por exemplo em [60, 65–67], dentre vários outros. Alguns trabalhos mais recentes para sistemas lineares com atrasos múltiplos podem ser encontrados em [58, 59, 68, 69]. Estes dois últimos já abordando LMI e o conceito de equações de Riccati.

O estudo de estabilidade da versão linear com atraso é importante pois ele se aproxima do modelo final de interesse, que é o Modelo de Rede Neural (3.30), uma vez que o **MRN** por vezes, opera na região linear da função $\phi_i(y_i(t)) = uhsat(y_i(t))$ - Ver Figura 2.2.

4.2.1.1 Versão Linear com Atraso Descrito na Variável x

Seja o modelo dado pela equação diferencial (3.27), reproduzida aqui por conveniência:

$$\dot{x}(t) = -KRx(t) + TKRx(t-h) + \lambda \tag{4.34}$$

Realizando a seguinte substituição de variáveis: $W = -KR$ e $W_h = TKR$, temos o seguinte sistema linear com atraso descrito na variável x:

$$\dot{x}(t) = Wx(t) + W_h x(t - h(t)) \tag{4.35}$$

De acordo com KOKAME e MORI [59], a função de Lyapunov candidata

$$V(t, x_t) = x^T(t)Px(t) + \int_{t-h(t)}^{t} x^T(\tau)Qx(\tau)d\tau \tag{4.36}$$

onde $P \succ 0$ e $Q \succ 0$, e a taxa de mudança no atraso $\dot{h}$ limitada pela seguinte relação: $\dot{h}(t) < \alpha$, $0 < \alpha < 1$, a estabilidade assintótica do modelo (4.35) é assegurada se existirem matrizes definidas positiva P e Q que satisfaçam a seguinte LMI:

$$\begin{bmatrix} PW + W^TP + Q & PW_h \\ W_h^TP & -(1-\alpha)Q \end{bmatrix} < 0 \tag{4.37}$$

Vamos apresentar o seguinte exemplo:

Exemplo 4.2.1. *Seja o modelo (4.35) com uma taxa de variação no atraso de* $\dot{h} = 0,4$, $\alpha = 0,7$ *e matrizes*

$$W = \begin{bmatrix} -3 & 2 \\ 2 & -6 \end{bmatrix}, \quad e \quad W_h = \begin{bmatrix} -0,3 & 0,3 \\ 0,3 & -0,8 \end{bmatrix} \tag{4.38}$$

onde $\lambda(W) = [-7 \quad -2]$ *e* $\lambda(W_h) = [-0,9405 \quad -0,1595]$ *são os autovalores das matrizes* W *e* W_h*, respectivamente.*

Utilizando a função ***feasp*** *da* ***toolbox de controle robusto*** *do Matlab para resolver a LMI (4.37), temos que a LMI é factível para as matrizes* W *e* W_h *(script Kokame_Mori_Exemplo.m):*

$$P = \begin{bmatrix} 0,8561 & 0,2907 \\ 0,2907 & 0,4184 \end{bmatrix}, \quad e \quad Q = \begin{bmatrix} 2,1435 & 0,0322 \\ 0,0322 & 2,0884 \end{bmatrix} \tag{4.39}$$

resultando em um sistema assintoticamente estável.

Os autovalores da matriz P são dados por $\lambda(P) = [0,2734 \quad 1,0011]$, e os autovalores da matriz Q são dados por $\lambda(Q) = [2,0736 \quad 2,1583]$.

Na solução numérica através da toolbox, as matrizes P e Q foram estruturalmente definidas como sendo do tipo *full-symmetric*. No entanto, outras soluções para as matrizes P e Q também são possíveis definindo a estrutura destas como sendo diagonal.

Exemplo 4.2.2. *Definindo estruturalmente as matrizes* P *e* Q *como sendo matrizes diagonais, e utilizando o mesmo exemplo anterior (4.2.1), temos a seguinte solução para (4.2.1):*

$$P = \begin{bmatrix} 23,8510 & 0 \\ 0 & 14,9900 \end{bmatrix}, \quad e \quad Q = \begin{bmatrix} 53,3989 & 0 \\ 0 & 67,1790 \end{bmatrix} \tag{4.40}$$

Nos exemplos (4.2.1) e (4.2.2) acima, as matrizes W e W_h eram matrizes de posto completo.

Vamos verificar o que acontece se introduzirmos uma alteração significativa no exemplo (4.2.1).

Exemplo 4.2.3. *Vamos definir a matriz* W *como sendo:*

$$W = \begin{bmatrix} -3 & 2 \\ 6 & -4 \end{bmatrix}, \quad e \quad W_h = \begin{bmatrix} -0,3 & 0,3 \\ 0,3 & -0,8 \end{bmatrix} \tag{4.41}$$

onde $\lambda(W) = [0 \quad -7]$ *e* $\lambda(W_h) = [-0,9405 \quad -0,1595]$ *são os autovalores das matrizes* W *e* W_h*, respectivamente.*

Neste caso, não houve solução para as matrizes P e Q que satisfizessem (4.37).

No entanto, para as seguintes matrizes:

Exemplo 4.2.4.

$$W = \begin{bmatrix} -3 & 2 \\ 2 & -6 \end{bmatrix}, \quad e \quad W_h = \begin{bmatrix} -0,3 & 0,3 \\ 0,6 & -0,6 \end{bmatrix} \tag{4.42}$$

onde $\lambda(W) = [-7 \quad -2]$ e $\lambda(W_h) = [0 \quad -0,9]$ são os autovalores das matrizes W e W_h, respectivamente.

A solução encontrada para (4.37), neste exemplo, foi:

$$P = \begin{bmatrix} 0,8655 & 0,2952 \\ 0,2952 & 0,4227 \end{bmatrix}, \quad e \quad Q = \begin{bmatrix} 2,1635 & 0,0385 \\ 0,0385 & 2,1057 \end{bmatrix} \tag{4.43}$$

e, outra solução:

$$P = \begin{bmatrix} 24,7209 & 0 \\ 0 & 15,4576 \end{bmatrix}, \quad e \quad Q = \begin{bmatrix} 55,4728 & 0 \\ 0 & 69,1429 \end{bmatrix} \tag{4.44}$$

para matrizes P e Q diagonais.

Podemos verificar que, para alguns casos onde uma das matrizes é singular, é possível encontrar soluções para a LMI (4.37). O script foi utilizado para diversos outros exemplos onde ambas as matrizes W e W_h eram singulares. Em nenhum exemplo as matrizes solução P e Q foram encontradas, dando indícios de que somente quando a matriz W_h é singular, é que é possível haver uma solução.

Foi possível notar também, através do script Matlab, indícios de que a solução para a LMI (4.37) somente é possível quando os elementos da matriz W_h forem, de maneira geral e aproximada, ao menos uma ordem de grandeza menores que os elementos da matriz W. Nos exemplos utilizados onde W e W_h possuíam elementos com a mesma ordem de grandeza, a solução não foi encontrada pois a LMI (4.37) não era factível.

Os exemplos dados anteriormente envolviam matrizes quaisquer. Para o próximo exemplo vamos buscar solução para a LMI (4.37) baseado nos parâmetros do modelo (4.34).

Exemplo 4.2.5. *Seja um cluster heterogêneo, formado por dois nós, com os seguinte parâmetros:* $t_{p_1} = 30\mu s$*,* $t_{p_2} = 90\mu s$ *e matriz de ganhos dada por* $K = diag(4530, \quad 3670)$*. Desta forma, temos:*

$$W = \begin{bmatrix} -2265 & 2265 \\ 1835 & -1835 \end{bmatrix}, \quad e \quad W_h = \begin{bmatrix} -611,7 & 611,7 \\ 6795,0 & -6795,0 \end{bmatrix} \tag{4.45}$$

Para o exemplo (4.2.5) também não foi possível encontrar as matrizes solução para a LMI (4.37). Esta fato já era de certa forma esperado, pois as matrizes W e W_h no modelo (4.34) são estruturalmente singulares devido à matriz R.

Apesar do problema de viabilidade da solução de (4.37) quando aplicada ao modelo (4.34), a estrutura da LMI (4.37) torna-se interessante para o problema de síntese de controladores utilizando a teoria de redes neurais e LMI, pois veremos que é possível descrever o problema de síntese em uma LMI com uma estrutura semelhante a (4.37).

Uma outra observação merece ser destacada ainda. Se a matriz (4.37) é factível, então a estabilidade é garantida para todo $h > 0$. Neste caso, a estabilidade é dita *delay-independent*.

4.2.1.2 Versão Linear com Atraso Descrito na Variável y

Retomando o modelo dado pela equação diferencial (3.28), descrita aqui por conveniência:

$$\dot{y}(t) = -RKy(t) + RTKy(t-h) + R\lambda \tag{4.46}$$

onde $W = -RK$ e $W_h = RTK$. Temos que, em termos de formato, ela não mudará em relação à equação (4.35). Logo, os resultados alcançados na subseção anterior são os mesmos para o modelo (4.46).

Um procedimento para evitarmos o problema de viabilidade de (4.37) quando aplicado ao problema de estabilidade do modelo (4.46) é fazermos um pequeno deslocamento na diagonal principal do elemento $(1,1)$ da LMI (4.37) na direção do semi-plano lateral esquerdo do plano complexo, da seguinte forma:

$$\begin{bmatrix} (-\varepsilon I + PW + W^T P + Q) & PW_h \\ W_h^T P & -(1-\alpha)Q \end{bmatrix} < 0 \tag{4.47}$$

onde ε é uma constante positiva e próxima do valor nulo ($\varepsilon \approx 0$), e I é a matriz identidade.

Exemplo 4.2.6. *Seja um cluster heterogêneo, formado por dois nós, com os seguinte parâmetros:* $t_{p_1} = 30\mu s$, $t_{p_2} = 90\mu s$, $\varepsilon = 10^{-7}$ *e matriz de ganhos dada por* $K = diag(4530, \quad 3670)$. *Desta forma, temos:*

$$W = \begin{bmatrix} -2265 & 1835 \\ 2265 & -1835 \end{bmatrix}, \quad e \quad W_h = \begin{bmatrix} -6795,0 & 611,7 \\ 6795,0 & -611,7 \end{bmatrix} \tag{4.48}$$

Substituindo as matrizes W *e* W_h *acima na LMI modificada (4.47), temos como solução (script Kokame_Mori_Exemplo2.m):*

$$P = 10^{-11} \begin{bmatrix} 0,1112 & 0 \\ 0 & 0,0907 \end{bmatrix}, \quad e \quad Q = 10^{-7} \begin{bmatrix} 0,7095 & 0 \\ 0 & 0,6935 \end{bmatrix} \tag{4.49}$$

4.2.2 Versão Não-Linear com Atraso

Vamos iniciar o estudo da estabilidade das versões não-lineares e com atraso. Como tem sido feito até aqui, primeiro iremos analisar o modelo expresso na variável x, e depois na variável y. Como poderemos constatar, o modelo não-linear na variável x não é adequado para ser tratado como uma particularidade do modelo de rede neural de Hopfield-Tank com atraso, necessitando assim de uma transformação de variáveis.

Baseado no trabalho de SINGH [70] sobre convergência de rede neurais com atraso via LMI, será proposta uma nova função de Lyapunov adequada para o **Modelo de Rede Neural** (3.30). Em seguida, a função será expressa na forma de uma LMI, a qual será utilizada no capítulo (5) para a síntese de controladores para o **MRN**.

4.2.2.1 Versão Não-Linear com Atraso Descrito na Variável x

O modelo não-linear com atraso descrito na variável x (3.29), reproduzido aqui por conveniência

$$\dot{x}(t) = -K\phi(y(t)) + TK\phi(y(t-h)) + \lambda \tag{4.50}$$

forma uma classe de sistema não-linear com atraso para a qual não foram, a priori, encontrados resultados de estabilidade publicados. O próprio modelo (2.1) é o que mais se aproxima deste, servindo então de base para a prova de convergência - seção (2.2.2).

4.2.2.2 Versão Não-Linear com Atraso Descrito na Variável y

Diversos trabalhos têm sido publicados sobre convergência de redes neurais com atraso devido ao grande interesse no conhecimento dos efeitos deste sobre o comportamento global da rede. Dentre eles podemos destacar [54, 71–73], entre outros.

Vários destes trabalhos foram analisados e não se adequaram para o **MRN** por dependerem fortemente da realimentação linear negativa do modelo (2.30).

Por outro lado, como será visto no capítulo 5, quando utilizamos uma LMI para síntese de controladores [74, 75], ela normalmente torna-se uma **Inequação Matricial Bilinear** ou **BMI**, a qual envolve multiplicações de variáveis e, conseqüentemente, tornando-se de difícil solução. Para contornar este problema, é usual empregar transformações de congruência e substituição de variáveis e, para que o processo seja viável, é desejável obter uma LMI com poucos elementos matriciais e poucas variáveis, permitindo um tratamento algébrico adequado de modo a transformá-la em uma LMI novamente.

Desta forma, para o problema de síntese de controladores utilizando o **MRN** (3.30), é necessário encontrar uma LMI que possua poucas variáveis e com poucos elementos matriciais.

Em SINGH [54], o autor realiza uma transformação de variáveis na equação (2.30) para promover um deslocamento do ponto de equilíbrio $y^*(t) = [y_1^*(t), \ldots, y_n^*(t)]^T$ para a origem, usando a relação $z(\cdot) = y(\cdot) - y^*$. Seu modelo utiliza a função não-linear dada por (2.31) e mostrada na Figura 2.8, resultando em:

$$\dot{z}(t) = -z(t) + W\tilde{\phi}(z(t)) + W_h\tilde{\phi}(z(t-h)) \tag{4.51}$$

onde $z(\cdot) = [z_1(\cdot), \ldots, z_n(\cdot)]^T$ é o novo vetor de estados, $\tilde{\phi}(z(\cdot)) = [\tilde{\phi}_1(z_1(\cdot), \ldots, \tilde{\phi}_n(z(\cdot))]^T$ representa o vetor de saída do modelo transformado, e $\tilde{\phi}_i(z_i(\cdot)) = y_i(z_i(\cdot) + x_i^*) - y_i(x_i^*)$, $i = 1, \ldots, n$. As funções $\tilde{\phi}_i(z_i(\cdot))$ satisfazem:

$$\tilde{\phi}^2(z_i(\cdot)) \leq z_i(\cdot)\tilde{\phi}(y_i(z_i(\cdot)), \quad \tilde{\phi}_i(0) = 0, \quad i = 1, \ldots, n. \tag{4.52}$$

Vimal Singh propõe a seguinte função de Lyapunov:

$$V(z(t)) = z^T(t)Pz(t) + 2\sum_{i=1}^{n} d_i \int_0^{z_i(t)} \tilde{\phi}_i(s)ds + \int_{t-\tau}^{t} \tilde{\phi}^T(z(\varsigma))Q\tilde{\phi}(z(\varsigma))d\varsigma \qquad (4.53)$$

e apresenta o seguinte teorema, o qual fornece condições suficientes para garantir a origem de (4.51) como único ponto de equilíbrio e sua estabilidade global assintótica:

Teorema 4.3. *[54] Suponha que existam matrizes definidas positivas simétricas $P = (p_{ij}) \in \mathbb{R}^{n\times n}$ e $Q = (q_{ij}) \in \mathbb{R}^{n\times n}$, e uma matriz definida positiva diagonal $D = (d_i) \in \mathbb{R}^{n\times n}$ tal que*

$$N = \begin{bmatrix} 2P & -PW & -PW_h \\ -W^TP & 2D - Q - DW - W^TD & -DW_h \\ -W_h^TP & -W_h^TD & Q \end{bmatrix} \succ 0 \qquad (4.54)$$

Então a origem de (4.51) é o único ponto de equilíbrio e ele é globalmente assintoticamente estável.

Baseado nos trabalhos de SINGH [54, 70], propomos uma função de Lyapunov que seja adequada ao modelo **MRN**:

$$\dot{z}(t) = W\tilde{\phi}(z(t)) + W_h\tilde{\phi}(z(t-h)) \qquad (4.55)$$

onde $W = -RK$, $W_h = RTK$, $z(\cdot) = [z_1(\cdot), \cdots, z_n(\cdot)]^T$ é o novo vetor de estados, e $\tilde{\phi}(z(\cdot)) = [\tilde{\phi}_1(z_1(\cdot)), \cdots, \tilde{\phi}_n(z_n(\cdot))]^T$ representa o novo vetor de funções de ativação do sistema transformado, e $\tilde{\phi}_i(z_i) = \phi_i(z_i(\cdot) + y_i^*) - \phi_i(y_i^*)$, $i = 1, \cdots, n$. Como a função $\phi(y_i(t))$ (2.3) satisfaz a condição (4.52), podemos propor a seguinte função de Lyapunov:

$$V(z(t)) = 2\sum_{i=1}^{n} d_i \int_0^{z_i(t)} \tilde{\phi}_i(s)ds + \int_{t-h}^{t} \tilde{\phi}^T(z(\varsigma))Q\tilde{\phi}(z(\varsigma))d\varsigma \qquad (4.56)$$

A derivada no tempo de $V(z)$ ao longo das trajetórias de (4.55) é dada por:

$$\dot{V}(z(t)) = \nabla^T z(t) \qquad (4.57)$$

Aplicando a equação acima em (4.56), temos:

$$\begin{aligned}\dot{V}(z(t)) =&[2D(\tilde{\phi}(z(t)) - \tilde{\phi}(0))]^T \dot{z}(t)+ \\ &+ \tilde{\phi}^T(z(t))Q\tilde{\phi}(z(t)) - \tilde{\phi}^T(z(t-h))Q\tilde{\phi}(z(t-h))\end{aligned} \tag{4.58}$$

Como $\tilde{\phi}(0) = 0$, temos

$$\dot{V}(z(t)) = 2\tilde{\phi}^T(z(t))D\dot{z}(t) + \tilde{\phi}^T(z(t))Q\tilde{\phi}(z(t)) - \tilde{\phi}^T(z(t-h))Q\tilde{\phi}(z(t-h)) \tag{4.59}$$

Substituindo (4.55) na equação acima, temos

$$\begin{aligned}\dot{V}(z(t)) =&2\tilde{\phi}^T(z(t))D[W\tilde{\phi}(z(t)) + W_h\tilde{\phi}(z(t-h))]+ \\ &+ \tilde{\phi}^T(z(t))Q\tilde{\phi}(z(t)) - \tilde{\phi}^T(z(t-h))Q\tilde{\phi}(z(t-h))\end{aligned} \tag{4.60}$$

Realizando as multiplicações necessárias, teremos

$$\begin{aligned}\dot{V}(z(t)) =&2\tilde{\phi}^T(z(t))DW\tilde{\phi}(z(t)) + 2\tilde{\phi}^T(z(t))DW_h\tilde{\phi}(z(t-h))+ \\ &+ \tilde{\phi}^T(z(t))Q\tilde{\phi}(z(t)) - \tilde{\phi}^T(z(t-h))Q\tilde{\phi}(z(t-h))\end{aligned} \tag{4.61}$$

Fazendo um re-arranjo nos termos multiplicados por 2

$$\begin{aligned}\dot{V}(z(t)) =&\tilde{\phi}^T(z(t))(DW - W^TD)\tilde{\phi}(z(t)) \\ &+ \tilde{\phi}^T(z(t))(DW_h^T + W_h^TD)\tilde{\phi}(z(t-h))+ \\ &+ \tilde{\phi}^T(z(t))Q\tilde{\phi}(z(t)) - \tilde{\phi}^T(z(t-h))Q\tilde{\phi}(z(t-h))\end{aligned} \tag{4.62}$$

e colocando na forma matricial:

$$\dot{V}(z(t)) = \begin{bmatrix}\tilde{\phi}^T(z(t)) & \tilde{\phi}^T(z(t-h))\end{bmatrix} M \begin{bmatrix}\tilde{\phi}(z(t)) \\ \tilde{\phi}(z(t-h))\end{bmatrix} \tag{4.63}$$

onde

$$M = \begin{bmatrix} DW + W^T D + Q & DW_h \\ W_h^T D & -Q \end{bmatrix} \tag{4.64}$$

Bastará então que a matriz M seja definida negativa para que o sistema possua um único ponto de equilíbrio globalmente estável. Desta forma, poderemos enunciar um teorema de estabilidade global assintótica para o modelo de rede neural em questão.

Teorema 4.4. *Dado o sistema (3.30), suponha que existam uma matriz simétrica $Q \in \mathbb{R}^{n\times n}$ definida positiva e uma matriz diagonal $D \in \mathbb{R}^{n\times n}$ definida positiva tal que*

$$M = \begin{bmatrix} DW + W^T D + Q & DW_h \\ W_h^T D & -Q \end{bmatrix} \prec 0 \tag{4.65}$$

Então, o modelo (3.30) é globalmente assintoticamente estável.

No entanto, as matrizes W e W_h são singulares, fazendo com que a LMI (4.65) não seja factível.

É possível notar uma certa semelhança com a LMI de Kokame e Mori (4.37). O escalonamento promovido no elemento $(2, 2)$ da LMI (4.37) se mostrará útil, de alguma forma, para a síntese de controladores para a rede neural proposta no capítulo 5.

4.3 Resumo do Capítulo

Neste capítulo, verificamos as condições de estabilidade para algumas das versões de modelos apresentados no capítulo 3, dando ênfase naquelas versões de maior interesse em relação ao modelo **MRN** (3.30).

O fato das matrizes $W = -RK$ e $W = RTK$ serem estruturalmente singulares apresentou uma dificuldade numérica na determinação da estabilidade. Isto é importante pois a determinação numérica da estabilidade global será um item fundamental no capítulo 5, de forma a buscarmos controladores que possuam convergência global garantida.

A introdução de um escalar ε, positivo e suficientemente pequeno, possibilitou a viabilidade dos testes de estabilidade das LMIs (4.37) e (4.65).

O aproveitamento da LMI (4.65) diretamente na síntese de controladores no capítulo 5

introduz uma nova dificuldade, pois a LMI torna-se uma BMI (Bilinear Matrix Inequality) nas variáveis matriciais D e K. Transformações de congruência e novas mudanças de variáveis permitirão transformar esta BMI em uma LMI novamente que, junto com uma Equação Algébrica de Riccati, auxiliará na definição do problema de síntese de controladores.

Podemos observar também que existe uma certa similaridade entre a LMI de Kokame e Mori, que assegura convergência para sistemas lineares com atraso, e a LMI proposta neste trabalho, que assegura convergência global para uma versão não-linear do modelo descrito em [59]. Na verdade, podemos concluir que este trabalho estende um pouco mais o trabalho publicado em [59].

```
>> hdstab(RR)

 Solver for LMI feasibility problems L(x) < R(x)
    This solver minimizes  t  subject to  L(x) < R(x) + t*I
    The best value of t should be negative for feasibility

 Iteration   :    Best value of t so far

     1                        0.023952
     2                    3.131281e-003
     3                    3.131281e-003
     4                    5.094711e-005
     5                    5.094711e-005
     6                    5.094711e-005
     7                    4.029840e-007
     8                    4.029840e-007
     9                    4.029840e-007
    10                    4.029840e-007
    11                    1.545738e-007
    12                    1.545738e-007
    13                    5.787263e-009
    14                    5.787263e-009
    15                    5.787263e-009
* switching to QR
    16                    5.787263e-009
    17                    1.288212e-010
    18                    1.317866e-011
    19                    1.317866e-011
    20                    7.104332e-013
    21                    7.104332e-013
    22                    1.312958e-014
    23                    1.312958e-014
    24                    1.312958e-014
    25                    1.218550e-015
    26                    -2.049876e-017

 Result:  best value of t: -2.049876e-017
          f-radius saturation:  0.000% of R = 1.00e+009


ans =

    1.0000         0         0
         0    1.0000         0
         0         0    1.0000

>>
```

Figura 4.1: Determinação da matriz P para demonstrar a estabilidade segundo Lyapunov

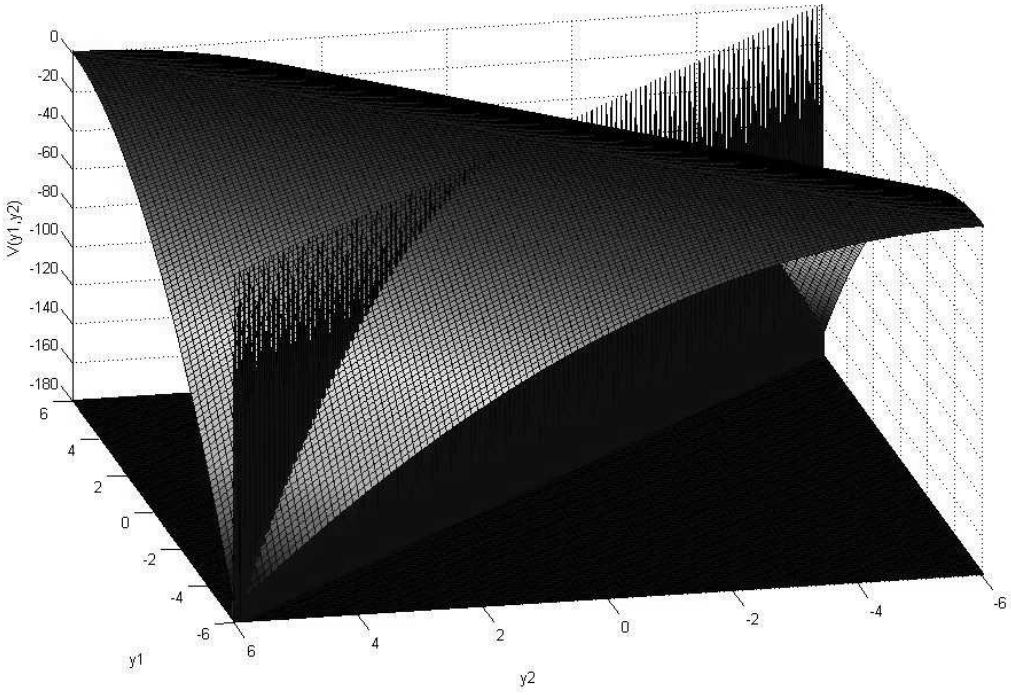

Figura 4.2: $\dot{V}(y_1, y_2)$ sujeita à $y_1 = -y_2$ é a parabola formada pela intersecção do plano com a calha

Capítulo 5

O Problema de Controle Objetivando Desempenho

O objetivo principal deste capítulo é: Uma vez que o **MB** tenha sido transformado em uma forma particular de rede neural de Hopfield com atraso - Modelo **MRN** (3.30), buscar um método para obter os ganhos da malha de controle que assegurem a estabilidade do sistema (não-linear) e assegure um critério de otimalidade (localmente).

A proposta é a de estabelecer uma função custo quadrático para o modelo versão linear na variável y (3.5), expressa na forma de uma LMI, a fim de assegurar um critério de otimalidade local e, simultaneamente, aplicar um critério de estabilidade para o **MRN** (não-linear) para assegurar a estabilidade. Expressando este critério de estabilidade também na forma de uma LMI, pode-se obter vantagem ao tratarmos desempenho local e estabilidade global utilizando o Método dos Pontos Interiores [76] simultaneamente.

A região hachurada da Figura 5.1 representa a região onde o sistema é linear e onde há uma imposição de desempenho. Fora desta região, não há como garantir o desempenho, mas é possível garantir a convergência global.

Como o modelo versão linear na variável y utiliza em sua malha de controle uma realimentação de saída - Figura 5.5, os métodos de alocação de pólos - *Pole Placement Design* - não se aplicam ao problema neste caso, pois o sistema não é observável. Faz-se necessário então uma abordagem pela chamada *Lyapunov Control Function*.

Diferentes técnicas de solução do problema de controle serão examinadas, de forma a verificarmos o custo e a eficácia de cada uma delas.

Para efeito de referência em um problema de pequenas dimensões, uilizamos uma função custo que é dada pelo índice de desempenho ITAE (Integral of Time Multiplied by Absolute Error) sujeita à condição de convergência dada pelo Teorema 4.4 para o modelo **MRN** (3.30). Para minimizar este índice de desempenho, utilizaremos algoritmos genéticos.

A vantagem do uso deste índice é que ele reduz grandes erros iniciais e enfatiza erros que acontecem mais tarde na resposta temporal do sistema [77]. Sua desvantagem é o alto custo computacional necessário para avaliar cada indivíduo da população de cromossomas, tornando-o inviável para grandes dimensões do problema. Ese índice servirá de *benchmark* para efeito de comparação com as outras propostas.

Uma outra proposta consiste em minimizar uma função custo quadrático para a versão linear na variável y dada pela norma da variável Z_1 da LMI (5.21). A vantagem desta proposta reside no custo computacional menor, porém, espera-se que os ganhos obtidos resultem em um desempenho menor do sistema, quando comparada com o índice ITAE, por esta não atuar diretamente sobre a saída $y(t)$ do modelo **MRN**.

O método proposto nesta tese consiste em minimizar uma função custo quadrático para a versão linear na variável y (3.5) sujeita à condição de convergência do Teorema (4.4) para o modelo **MRN** (3.30). Reescrevendo-se a função custo na forma de uma Equação Algébrica de Riccati (ARE) e colocando-a na forma de uma LMI, é possível fazer com que ela seja solucionada simultaneamente à LMI (4.65), que assegura a estabilidade do modelo não-linear. Tais LMIs deverão ser readequadas para que elas possuam variáveis em comum, possibilitando assim esta solução simultânea. Desta forma, os ganhos obtidos deverão proporcionar um bom desempenho para o sistema (garantido pela função custo quadrático reescrita na forma de uma LMI ARE) e simultaneamente satisfazer as condições de convergência do modelo não-liner (garantida pela LMI (4.65)).

O que justifica esta abordagem é que, devido à função não-linear - Figura 2.2, para pequenas amplitudes positivas da variável $y(t)$ (região I da figura), o sistema comporta-se como um sistema linear e, como tal, podemos impor o comportamento de um controlador quadrático. Fora desta região (regiões II e III), a convergência será assegurada pelo Teorema 4.4.

As simulações e os experimentos mostram melhora no desempenho utilizando essa abordagem e a Figura 5.1 ilustra a idéia da proposta.

5.1 Leis de Controle

Nesta seção, faremos uma análise da lei de controle (2.2) proposta no modelo (2.1) e de outras leis de controle estudadas durante o desenvolvimento deste trabalho.

Primeiramente analisaremos o significado da função não-linear $uhsat(y(t))$ utilizada no modelo (2.1) e a possibilidade de utilização de outras funções - Figura 5.3. Em seguida, analisaremos o resultado de uma matriz de ganhos K não diagonal, como citado na subseção 3.1.1, sobre o modelo (2.1).

5.1.1 Lei de controle do Modelo (2.1)

A lei de controle $v_i(t)$ (2.2) dada no modelo (2.1) atua de forma contínua e linear entre os seus estados liga e desliga (região I da Figura 2.2). Esta ação de "liga-desliga" indica se um determinado nó irá receber ou transferir tarefas no modelo (2.1).

Reproduzindo a equação (2.2) aqui por conveniência:

$$v_i(y_i(t)) = -K_i uhsat(y_i(t)) \tag{5.1}$$

temos que a variável de decisão $y_i(t)$ é o argumento da função não-linear $uhsat$. Isto significa que, se o tamanho da i-ésima fila estiver acima da média local, então a variável $y_i(t)$ é positiva e a função $uhsat$ terá valores positivos.

Em seguida, a função $uhsat$ é multiplicada pelo ganho K_i. A Figura 5.2 mostra a função $v_i(y_i(t))$ para diferentes valores de ganhos.

As curvas na Figura 5.2 traduzem-se em uma maior ou menor quantidade de tarefas retiradas da i-ésima fila, conforme o ganho maior ou menor, respectivamente. A saturação y_{max_i} da função $uhsat(y_i(t))$ (2.6) é basicamente uma restrição dada pela largura de banda da rede, e pode ser determinada através da seguinte equação [46]:

$$y_{max_i} = \frac{\text{Largura de Banda da Rede (bps)}}{\text{Tamanho Médio da Tarefa (bits)}} \times t_{p_i} \tag{5.2}$$

que por sua vez, é adimensional.

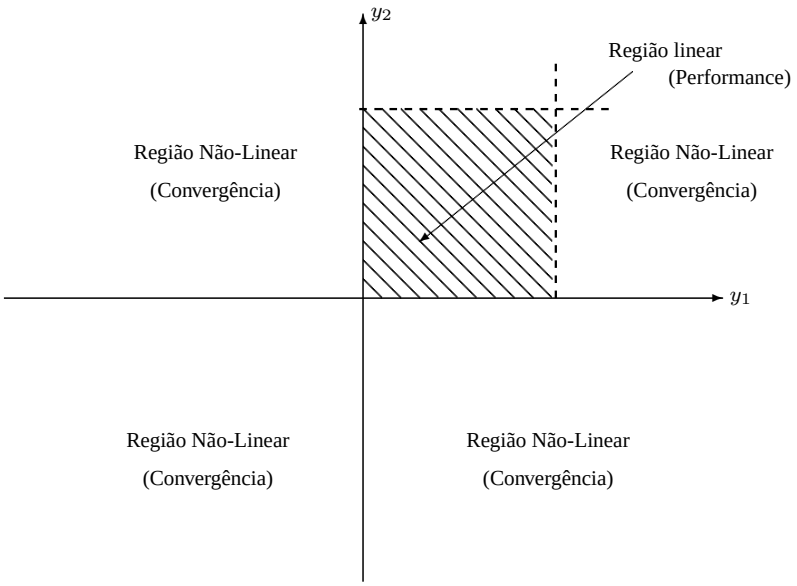

Figura 5.1: Convergência do Controlador Proposto

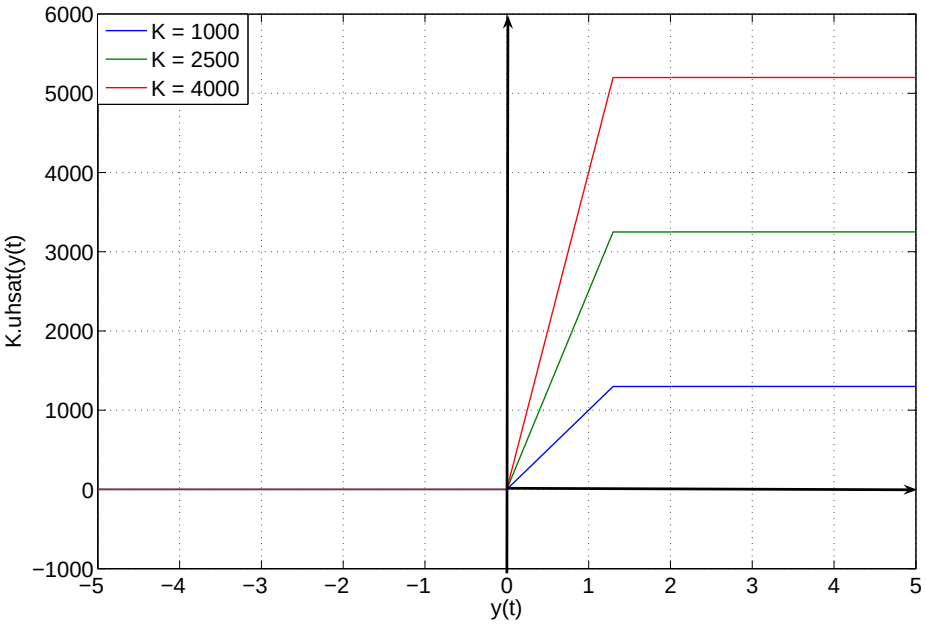

Figura 5.2: Função $v_i(y_i(t))$ para diferentes valores de ganho

5.1.2 Lei de controle proposta

Como foi assumido na seção (3.1.1) que a matriz de ganhos K não será diagonal, a lei de controle passa a ser dada por:

$$v_i(y(t)) = \sum_{j=1}^{n} K_{ij}\phi(y_j(t)) \tag{5.3}$$

A idéia desta lei de controle a partir de uma matriz não diagonal é flexibilizar o esforço de controle, aumentando o grau de liberdade do modelo (2.1). Com uma matriz geral, espera-se obter um desempenho melhor dos controladores e utilizando ganhos menores.

Adotando esta nova lei de controle, o modelo (2.1) pode ser reescrito na seguinte forma:

$$\dot{x}_i(t) = \sum_{j=1}^{n} K_{ij}\phi(y_j(t)) + \sum_{l=1}^{n}\sum_{m=1}^{n}\sum_{j=1}^{n} K_{ij}p_{il}\frac{t_{p_i}}{t_{p_j}}K_{mj}\phi(y_j(t-h)) \tag{5.4}$$

Qualquer proposição de um controlador para o modelo (2.1) deverá manter uma de suas principais características que é a conservação do número de tarefas quando o termo λ for nulo. No apêndice (A) encontra-se a prova de que a conservação do número total de tarefas para uma matriz não-diagonal se mantém.

5.1.3 Outras Leis de Controle

Outra possibilidade que é investigada é do uso de outras funções não-lineares, tanto de primeiro quadrante como de primeiro-terceiro quadrante, tais como:

sat(y(t)) Esta função corresponde à uma função não-linear de primeiro e terceiro quadrante, ou seja, é a função uhsat(y(t)) incluindo a parte inferior;

sign(y(t)) Corresponde à uma função não-linear do tipo *on-off* introduzindo uma descontinuidade no sistema;

tanh(y(t)) Esta função é uma função contínua, suave e derivável.

A Figura 5.3 mostra graficamente as funções não-lineares investigadas na simulação [09].

5.2 Sistemática para Análise dos Resultados das Simulações

Para facilitar as comparações nas simulações, será estabelecido nesta seção um conjunto de parâmetros que irão auxiliar a mensurar os resultados, tornando-os menos subjetivos.

Uma das técnicas de comparação de desempenho na teoria de controle linear é a resposta ao degrau unitário, onde uma entrada de valor unitário é aplicada a um sistema linear com condições iniciais nulas.

Em seguida, é feita uma análise do regime transitório do sistema em malha fechada e em relação à vários parâmetros, dentre eles: *Maximum Overshoot* (M_p), *Maximum Overshoot Instant* (M_t) e *Settling Time* (t_s).

O *maximum overshoot* é definido como o desvio máximo da saída sobre a amplitude do sinal de entrada durante o estado de transição. O valor máximo de overshoot é normalmente dado em forma percentual, ou seja:

$$M_p \quad (\%) = \frac{\text{overshoot máximo}}{\text{valor de regime}} \tag{5.5}$$

No âmbito desta tese, a variável que nos interessa em maior grau é o sinal $y(t)$ que corresponde à variável erro do processo de balanceamento de carga.

Como o valor de regime desejado é nulo, a definição de *overshoot* não é aplicável. Para contornar este problema, vamos definir o overshoot como o valor absoluto do desvio máximo em relação à ordenada do gráfico do sinal vetorial de erro $y(t)$, quando o gráfico a ser analisado for o do erro, e como o valor absoluto do desvio máximo em relação ao valor de equilíbrio, quando o gráfico a ser analisado for o do tempo de espera estimado.

O instante em que ocorre o maximum overshoot é definido como *maximum overshoot instant*.

O *settling time* é definido como o tempo necessário para que a resposta ao degrau diminua e permaneça dentro de um determinado percentual do seu valor final. Um valor normalmente utilizado é de 5%, o que corresponde ao valor $0,05$ do degrau unitário.

No contexto desta tese, o settling time será definido como sendo o tempo necessário para que a resposta transitória entre e permaneça dentro de uma região em torno da ordenada $y(t)$. Esta região será definida como:

$$t_{s_i} = t | y_i(t) \leq 5t_{p_i} \tag{5.6}$$

Como entrada, valores constantes serão atribuídos ao vetor $\lambda = [\lambda_1, \ldots, \lambda_n]$ com condições iniciais nulas durante um intervalo de tempo t_λ, o que significa inserir tarefas à uma taxa constante em um cluster inicialmente sem tarefas em suas filas.

Estas entradas serão como pulsos λ_p de amplitude e largura t_λ convenientes para excitar as funções não-lineares do modelo **MB** (3.1), levando-as a excursionar nas regiões lineares e não-lineares.

A Figura 5.4 nos mostra a entrada (λ_p) que será utilizada para excitar o sistema no caso de um cluster com três processadores ($n = 3$). O uso destes pulsos se justificam porque eles reproduzirão a situação em que as filas do sistema estarão vazias inicialmente e, em seguida, receberão tarefas em uma determinada taxa constante (amplitude dos pulsos) e por um período de tempo predeterminado (largura dos pulsos).

As amplitudes dos pulsos devem ser escolhidas de forma que ocorram erros iniciais $y(0) \neq 0$ - ver equação (2.4) - a ponto das funções não-lineares $\phi(y(t))$ serem levadas à saturação superior (região II). Com isto, poderemos colher informações do comportamento do sistema quando atuando nas regiões não-lineares.

Ao término do pulso (t_λ), deveremos observar uma situação semelhante ao início de uma outra simulação, onde a entrada passa a ser nula e as condições iniciais passam a ser aqueles valores do vetor $y(t)$ quando do término do pulso, ou seja, $y(0) = [y_1(t_{pulso}) \quad y_2(t_{pulso}) \quad y_3(t_{pulso})]^T$, onde t_{pulso} é o instante exato do término do pulso. Após a aplicação desta entrada, os parâmetros M_p e t_s serão analisados e tabelados de acordo com a origem da matriz de ganhos: exemplos publicados ou propostos pela síntese de controladores.

Esta metodologia de simulação nos permitirá obter informações de uma forma mais sistemática.

A fim de monitorarmos a atividade da lei de controle durante as simulações, os sinais oriundos da saída das funções não-lineares $\phi(y(t))$ foram armazenados, e serão mostrados conforme o estabelecido a seguir - Ver Figura 2.2.

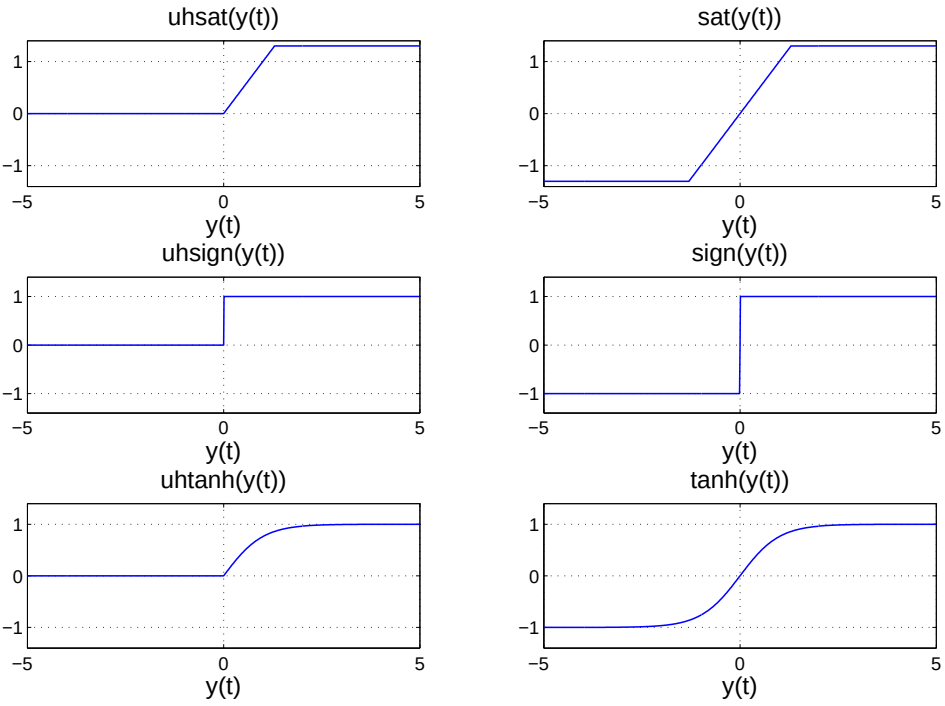

Figura 5.3: Funções não-lineares analisadas na simulação [09].

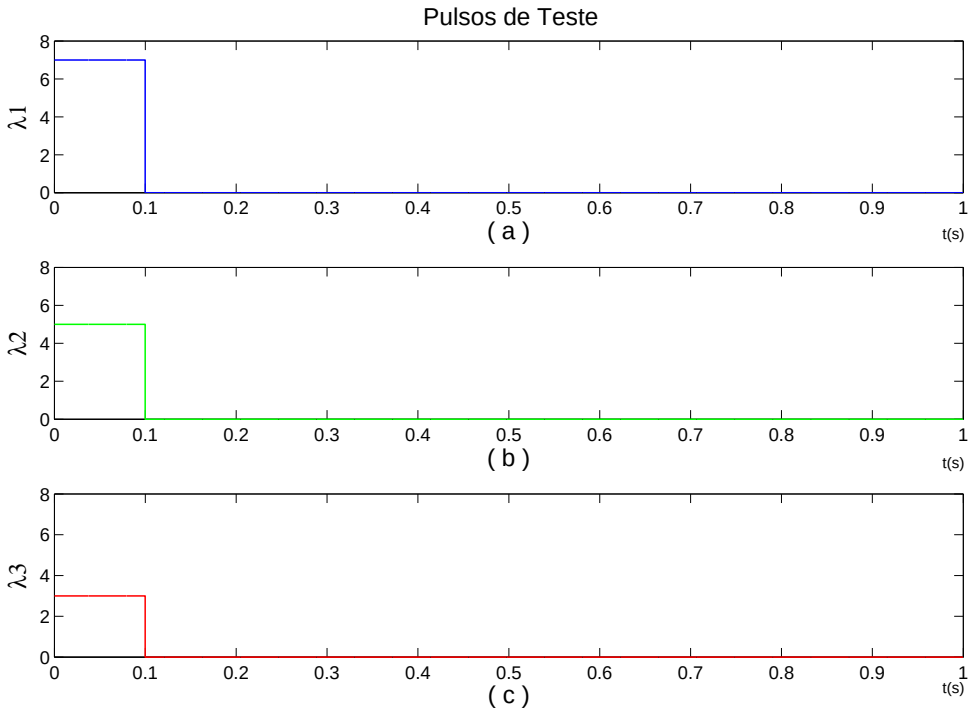

Figura 5.4: Pulsos de entrada $\lambda_p = [7 \quad 5 \quad 3]^T$ e $t_\lambda = 100ms$

$$\phi(y_i(t)) = \begin{cases} 1, & \text{se } y_i(t) \text{ estiver na região não-linear II} \\ 0,5, & \text{se } y_i(t) \text{ estiver na região linear I} \\ 0, & \text{se } y_i(t) \text{ estiver na região não-linear III} \end{cases} \tag{5.7}$$

Como exemplo, a Figuras 5.9, entre outras, apresenta os sinais dos controladores - gráfico 5.9a - junto ao sinal de erro $y(t)$ - gráfico 5.9b.

5.3 Síntese de Controladores

Como foi visto no capítulo (3), passamos de um modelo com realimentação de saída (3.16):

$$\begin{cases} \dot{x}(t) = (A + BKR)x(t) + \lambda, \quad A = 0, \quad B = (-I + T) \\ y(t) = Rx(t) \end{cases} \tag{5.8}$$

com lei de controle $u(t) = KRx(t)$, para um modelo de realimentação de estados (3.20):

$$\begin{cases} \dot{y}(t) = (A + BK)y(t) + R\lambda, \quad A = 0, \quad B = R(-I + T) \\ z(t) = Cy(t), \quad C = I \end{cases} \tag{5.9}$$

com lei de controle $u(t) = Ky(t)$, dada pela mudança de variáveis $y(t) = Rx(t)$ (3.13).

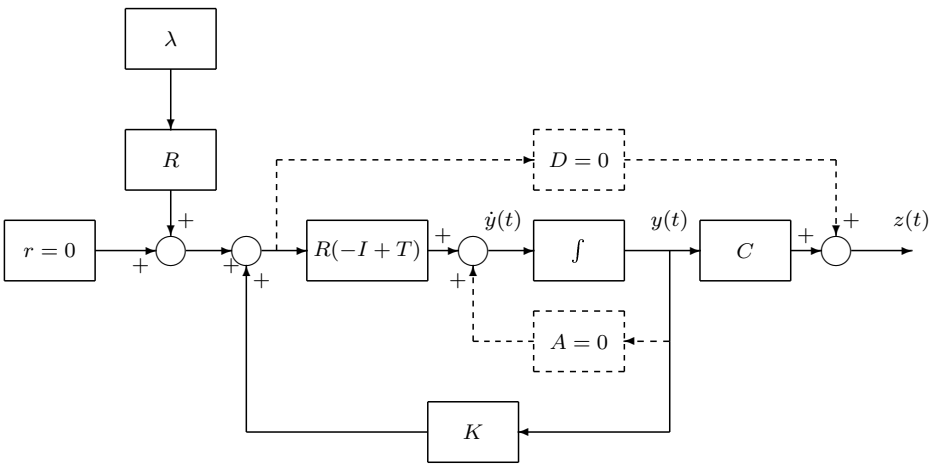

Figura 5.5: Diagrama em blocos do modelo versão linear na variável y sem atraso

Do ponto de vista da teoria de controle, a matriz de ganhos K regula o tamanho do erro que deve ser realimentado para o sistema, de forma a reduzí-lo até o valor nulo.

A busca pela minimização de um índice de desempenho a fim de atingirmos um controle ótimo significa escolher, dentre as trajetórias possíveis, aquela que forneça o menor tempo de convergência ou de menor energia.

No âmbito dos trabalhos publicados sobre redes neurais com atraso, os autores normalmente abordam apenas a convergência global, seja assintótica ou exponencial, dependente do atraso ou não. Porém, em princípio, não foi encontrado nenhum trabalho sobre controle de custo garantido aplicado à redes neurais.

Neste capítulo, a proposta da síntese de controladores será baseada em trabalhos publicados sobre controladores de custo garantido (*Guaranteed Cost Control*) via LMI [78–81], dentre outros.

O problema da síntese de controladores para a rede neural proposta pode ser enunciado da seguinte forma:

$$\min_{K} J = f(y(t), K) \tag{5.10}$$

sujeito às condições de estabilidade do teorema (4.4)

$$M = \begin{bmatrix} DW + W^T D + Q & DW_h \\ W_h^T D & -Q \end{bmatrix} \prec 0 \tag{5.11}$$

onde, $Q \succ 0$ e $D \succ 0$ é uma matriz diagonal.

Adequando o problema de síntese para o enunciado acima, temos que:

1. Aplicar as condições de estabilidade ao problema de síntese de controladores pretendido e,

2. Definir uma função custo J baseada em um índice de desempenho $f(y(t), K)$.

Os autores mencionados acima [78–81] propõem uma síntese de controladores baseada em uma função custo do tipo

$$\min_{K} J = \int [y^T(t)\Omega y(t) + u^T(t)\Psi u(t)]dt \tag{5.12}$$

5.3.1 Condições de Estabilidade ao Problema de Síntese do Sistema Não-Linear

Para aplicar as condições de estabilidade no problema de síntese de controladores para a rede neural com atraso, devemos substituir $W = -RK$ e $W_h = RTK$ em (5.11), onde as matrizes R e T são matrizes paramétricas conhecidas.

Após as substituições, teremos:

$$M = \begin{bmatrix} (-DRK - K^T RD + Q) & DRTK \\ K^T T^T RD & -Q \end{bmatrix} \preceq 0 \tag{5.13}$$

que deixa de ser uma LMI e passa a ser uma BMI em D e K, de tratamento numérico difícil.

Como é desejável o uso desta BMI diretamente na síntese de controladores, é necessário aplicar uma transformação de congruência $diag(D^{-1}, D^{-1})$ em M pela esquerda e pela direita a fim de torná-la uma LMI novamente:

$$\begin{bmatrix} D^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix}^T \begin{bmatrix} (-DRK - K^T RD + Q) & DRTK \\ K^T T^T RD & -Q \end{bmatrix} \begin{bmatrix} D^{-1} & 0 \\ 0 & D^{-1} \end{bmatrix} \preceq 0 \tag{5.14}$$

Após as multiplicações matriciais, temos a seguinte BMI equivalente:

$$\begin{bmatrix} (-RKD^{-1} - D^{-1}K^T R + D^{-1}QD^{-1}) & RTKD^{-1} \\ D^{-1}K^T T^T R & -D^{-1}QD^{-1} \end{bmatrix} \preceq 0 \tag{5.15}$$

Substituindo KD^{-1} por Z_1 e $D^{-1}QD^{-1}$ por Z_2, transformamos a BMI em uma LMI:

$$\tilde{M} = \begin{bmatrix} (-RZ_1 - Z_1^T R + Z_2) & RTZ_1 \\ Z_1^T T^T R & -Z_2 \end{bmatrix} \preceq 0 \tag{5.16}$$

Desta forma, o enunciado do problema de síntese de controladores passa a ser[1]:

$$\min_{Z_1, Z_2} J = f(y(t), Z_1, Z_2) \tag{5.17}$$

sujeito à

[1]Como houve uma mudança de variáveis, a função custo J não pode ser mais expressa em termos de K, e sim em termos de Z_1 e/ou Z_2.

$$\tilde{M} \preceq 0, \quad Z_1 \succ 0, \quad Z_2 \succ 0 \tag{5.18}$$

A transformação de congruência altera os autovalores da matrizes M, porém não altera os sinais destes autovalores, conservando assim as propriedades de inércia, conforme o Teorema da Inércia de Sylvester.

Teorema 5.1 (Teorema da Inércia de Sylvester)**.** *[62] Dada uma matriz simétrica $A \in \mathbb{R}^{n\times n}$, uma matriz C não-singular e a seguinte transformação:*

$$A \rightarrow C^T AC \tag{5.19}$$

A matriz $C^T AC$ possui o mesmo número de autovalores positivos que a matriz A, o mesmo número de autovalores negativos e o mesmo número de autovalores nulos.

Resolvendo as equações (5.17) e (5.18) pelo método dos pontos interiores, as matrizes Z_1 e Z_2 serão determinadas. O ganho poderá ser obtido a partir da relação:

$$K = Z_1 D^{-1} \tag{5.20}$$

No entanto, para definirmos completamente o ganho, é necessário determinar a matriz diagonal D.

Introduzindo-se em $\tilde{M}$ uma perturbação εD^{-1}, obtemos a seguinte LMI:

$$Z_c = \begin{bmatrix} (-\varepsilon D^{-1} - RZ_1 - Z_1^T R + Z_2) & RTZ_1 \\ Z_1^T T^T R & -Z_2 \end{bmatrix} \prec 0, \quad Z_1 \succ 0, \quad Z_2 \succ 0 \tag{5.21}$$

onde $\varepsilon \in \mathbb{R}^+$ e $\varepsilon \approx 0$, que garante convergência global para o sistema (3.30).

Nesta nova forma, as incógnitas D^{-1}, Z_1 e Z_2 da LMI (5.21) são determinadas, permitindo agora a recuperação da matriz de ganhos desejada através da equação (5.20).

Reescrevendo então o enunciado do problema de síntese de controladores, teremos:

$$\min_{Z_1, Z_2} J = f(y(t), Z_1, Z_2) \tag{5.22}$$

sujeito à

$$Z_c \prec 0, \quad Z_1 \succ 0, \quad Z_2 \succ 0 \tag{5.23}$$

A escolha da função custo $J = f(y(t), Z_1, Z_2)$ possui um papel preponderante no sucesso da síntese de controladores.

5.3.2 Definição da Função Custo

De acordo com a teoria de controle ótimo, o índice de desempenho ITAE (Integral of Time Multiplied by Absolute Error) fornece a melhor seletividade dentre os índices de desempenho mais comuns (ISE, IAE e ITSE) [77], isto é, o valor mínimo da integral é prontamente discernível ao serem variados os parâmetros de um sistema.

O índice de desempenho ITAE é dado por

$$J_{ITAE} = \int_0^T t|e(t)|dt \tag{5.24}$$

onde $e(t)$ é o erro de saída do sistema, normalmente dado por $e(t) = y(t) - r(t)$, $r(t)$ é o sinal de referência para o sistema em malha fechada. No caso deste trabalho, $r(t) = 0$.

Este índice possui como vantagem a redução de grandes erros iniciais no valor da integral de desempenho e enfatiza erros que acontecem mais tarde na resposta. Sendo assim, vamos definir o problema de síntese de controladores para a rede neural com atraso da seguinte forma:

Definição 5.1 (Síntese com índice ITAE)**.** *Seja o problema de minimização com restrições dado por*

$$\min_K J_{ITAE} = \sum_{i=1}^{n} \int_0^{t_f} t|y_i(t)|dt \tag{5.25}$$

sujeito à equação

$$\dot{y}(t) = -RK\phi(y(t)) + RTK\phi(y(t-h)) \tag{5.26}$$

Encontrar o valor de K que minimize a função custo J_{ITAE}.

O índice ITAE pode fornecer ganhos ideais pois ele ajuda a minimizar diretamente o erro da saída do sistema, pois $y(t) = e(t)$ - vide equação (2.4), no entanto, não garante

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 10\mu s \quad 10\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0,6 \quad 0,4 \quad 0,2]^T$
λ	$[0 \quad 0 \quad 0]^T$
y_{max}	1,3

Tabela 5.1: Parâmetros para a simulação [01]

estabilidade pois o sistema é não-linear.

Controladores utilizando este índice como funcional serão um tipo de *benchmark* para permitir comparações com outros métodos que possuam um custo computacional menor.

Utilizaremos algoritmos genéticos (AGs) para resolver o problema enunciado na Definição 5.1 pois os AGs possuem a vantagem de não depender das características de não-linearidade e atrasos do problema. Sua principal desvantagem no entanto, é o alto custo computacional.

5.3.2.1 Simulação [01]: Desempenho com a matriz de ganhos diagonal

Vamos reapresentar os resultados da simulação mostrada no exemplo (2.2.1) para permitir uma comparação do resultado publicado em [30] com os resultados encontrados pela solução do problema da Definição 5.1.

Os parâmetros serão repetidos na Tabela 5.1 por questões de conveniência.

Como em algumas simulações utiliza-se mais de uma matriz de ganhos para permitir comparações de desempenho, para facilitar o discernimento da matriz de ganhos K entre as diferentes simulações, vamos introduzir aqui a seguinte notação: $K_{[c]}$ é uma das matrizes de ganhos publicada por Chiasson *et al.* em [33], e $K^j_{[i]}$ para denotar a j-ésima matriz de ganhos K da i-ésima simulação.

A matriz de ganhos $K_{[c]}$ dada em [30] é:

$$K_{[c]} = \begin{bmatrix} 6667 & 0 & 0 \\ 0 & 4167 & 0 \\ 0 & 0 & 5000 \end{bmatrix} \tag{5.27}$$

Para esta matriz de ganhos, o resultado da simulação pode ser visto na Figura 5.6.

A Figura 5.6a: Tempo de espera representa o tempo de espera estimado para uma tarefa que esteja chegando à i-ésima fila - variável $x_i(t)$; A Figura 5.6b: Erro representa o quanto (tempo médio) a i-ésima fila está acima ou abaixo da média geral - variável $y_i(t)$; A Figura 5.6c: Tamanho das filas representa o número de tarefas presentes em cada fila - variável $q_i(t)$ e, finalmente, a Figura 5.6d: Quantidade de Tarefas representa o total de tarefas nas filas, ou seja, a soma em cada instante da quantidade de tarefas em cada uma delas.

Na Figura 5.6b, podemos notar que as curvas partem de $y(0) = [0,2 \quad 0 \quad -0,2]^T$. Isto ocorre porque, de acordo com a equação (3.13), temos:

$$y(0) = \frac{1}{3}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}\begin{bmatrix} 0,6 \\ 0,4 \\ 0,2 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0,0 \\ -0,2 \end{bmatrix} \tag{5.28}$$

As três curvas apresentam overshoot e convergem em um settling time de $4,12$ ms - ver fim de atividade dos controladores na Figura 5.8b. Já a Figura 5.6c nos mostra, inicialmente, um declínio no número total de tarefas nas filas, retornando ao número inicial conforme o sistema se aproxima do equilíbrio. Este declínio ocorre porque as filas começam a trocar tarefas entre si imediatamente, enviando-as pela rede, e reduzindo assim o número total de tarefas presentes nas filas. Quando as tarefas transferidas começam a chegar nos seus destinos, o número total contabilizado nas filas começa a aproximar-se do número inicial (princípio da conservação de tarefas).

5.3.2.2 Simulação [02]: Desempenho com a matriz de ganhos dada pela síntese com índice ITAE + AG

Vamos repetir a simulação [01] porém com os ganhos encontrados pela proposta da Definição 5.1. Os parâmetros utilizados no algoritmo genético encontram-se na Tabela 5.2.

Utilizando os parâmetros mostrados nesta Tabela, a matriz de ganhos determinada pelo algoritmo genético, resultado da minimização do índice de desempenho ITAE, é:

$$K_{[02]} = \begin{bmatrix} 1823,0 & -1623,5 & -1941,4 \\ 87,9 & 1541,3 & 1974,4 \\ -1503,8 & 996,3 & 1380,8 \end{bmatrix} \tag{5.29}$$

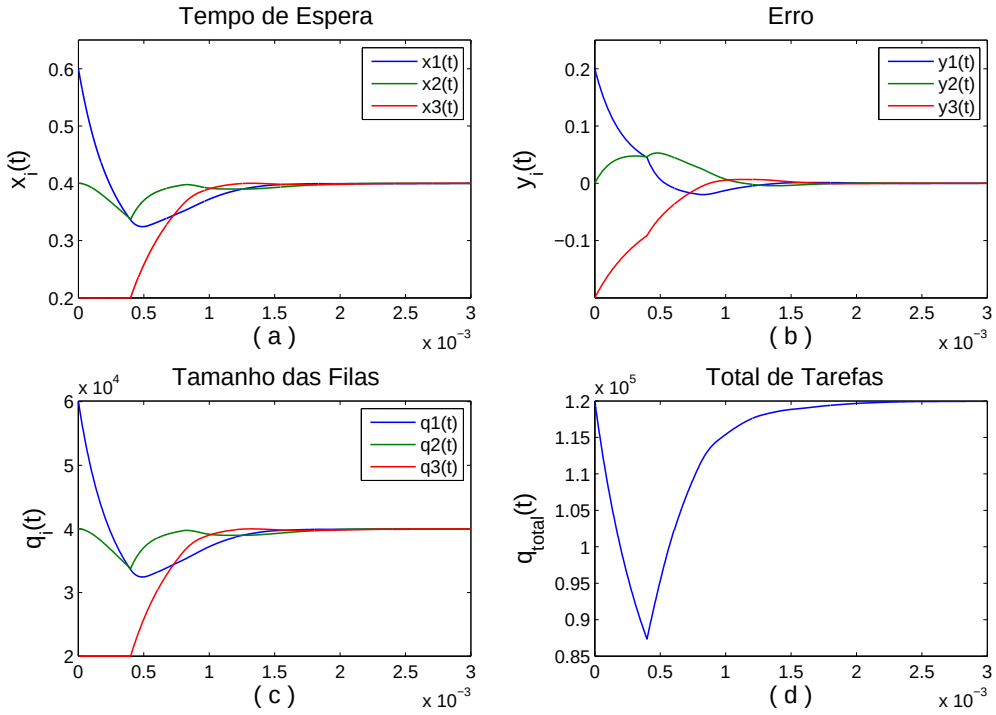

Figura 5.6: Simulação [01]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[c]}$

Parâmetro	**Valor(es)**
Gerações	100
População	200
Intervalo de Ganhos	$[-2000 \quad 2000]$
Taxa de Elitismo	20%
Variáveis	9

Tabela 5.2: Parâmetros do Algoritmo Genético para a simulação [02]

	Matriz de Ganhos $K_{[c]}$			Matriz de Ganhos $K_{[02]}$		
Node	M_p	M_t	t_s	M_p	M_t	t_s
Node 1	-0,019	0,83 ms	3,95 ms	-8,45E-3	1,39 ms	3,29 ms
Node 2	0,053	0,47 ms	3,34 ms	2,96E-3	0,67 ms	2,76 ms
Node 3	0,006	1,13 ms	4,12 ms	0	-	0 ms

Tabela 5.3: Parâmetros de desempenho entre as simulações $[01]$ e $[02]$

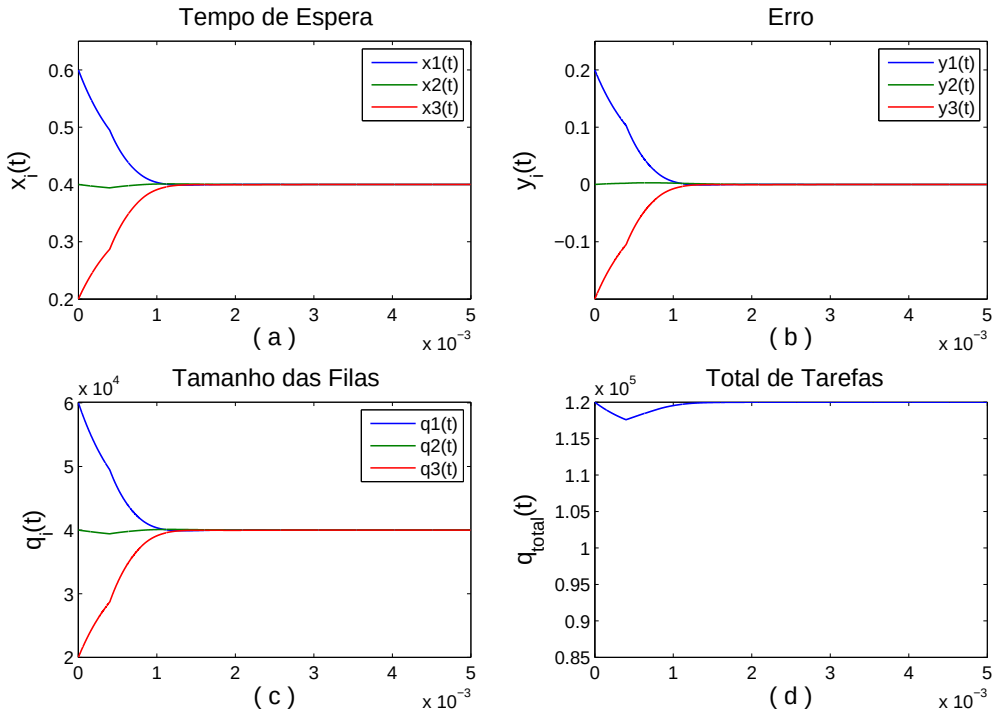

Figura 5.7: Simulação [02]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[02]}$

Comparando as Figuras 5.6 e 5.7 podemos notar que as curvas foram, de modo geral, mais suaves nesta última. Em especial, a figura que mostra a quantidade de tarefas em 5.7d indica que um número menor de tarefas foi transferido durante o processo de balanceamento (cerca de 3000 tarefas), enquanto na Figura 5.6, este número foi consideravelmente maior (cerca de 32500 tarefas movidas).

Isto nos permite verificar que a síntese de controladores utilizando o índice ITAE oferece um bom resultado.

A Figura 5.9 destaca a Figura 5.7b, relacionando-a às atividades dos sinais de controle. No gráfico 5.9b podemos notar que as atividades de controle cessam em $3,29$ ms, que é definido como o settling time.

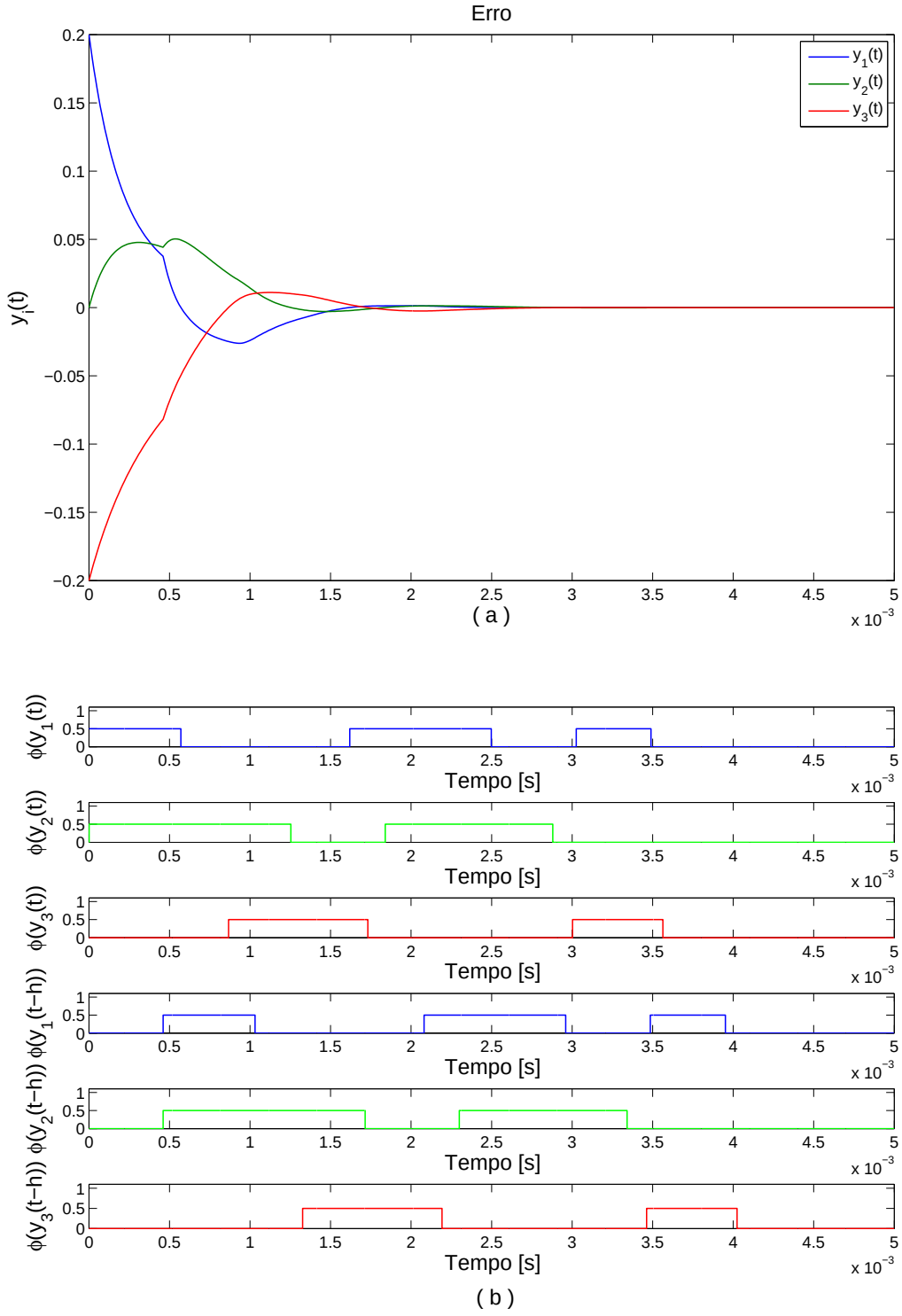

Figura 5.8: Simulação [01]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[c]}$ (5.27). O gráfico (a) apresenta o sinal de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

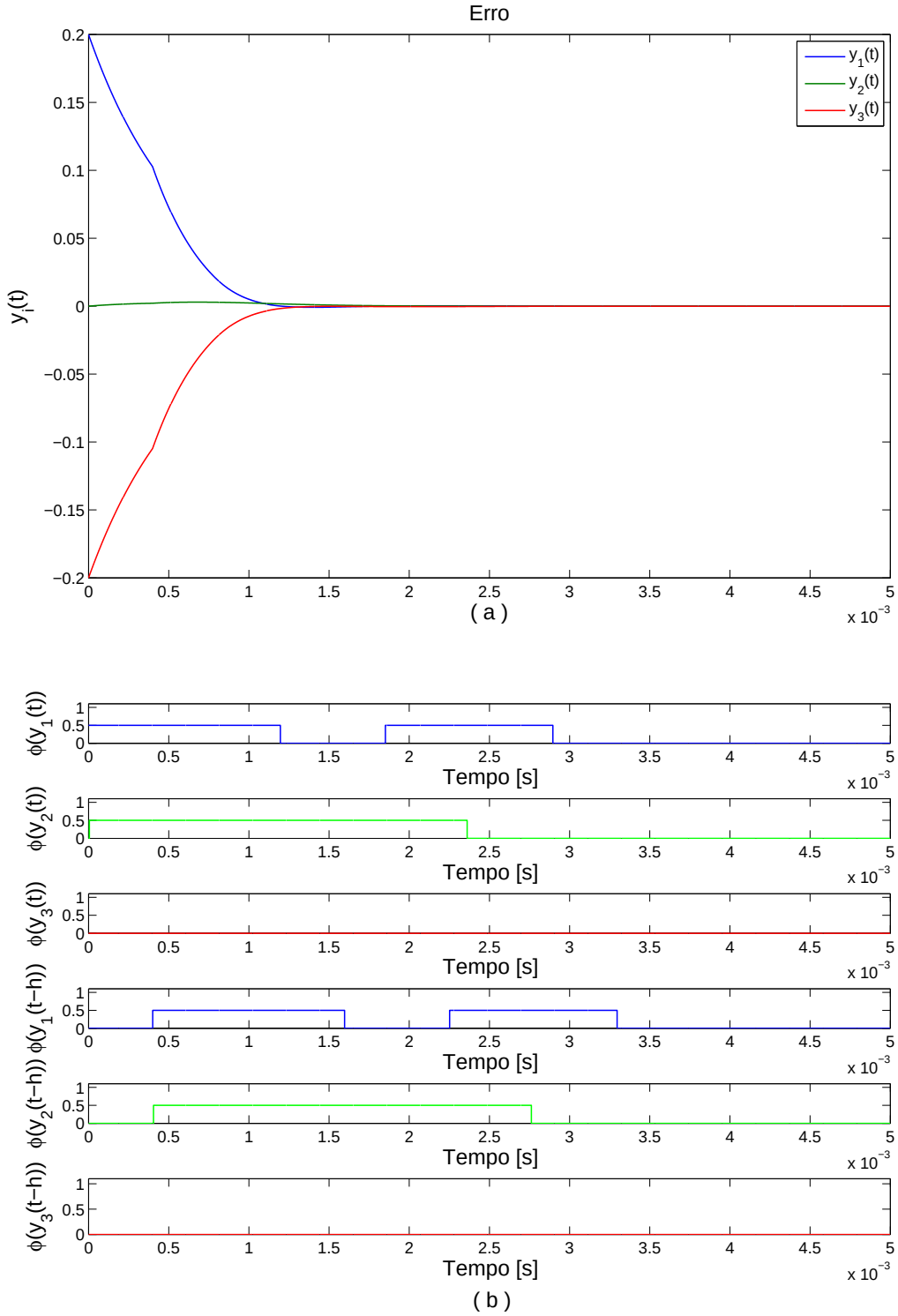

Figura 5.9: Simulação [02]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[02]}$ (5.29). O gráfico (a) apresenta o sinal de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

No entanto, o índice de desempenho ITAE somente não garante uma solução estável para qualquer condição inicial do problema. De fato, se introduzirmos uma perturbação de apenas 5% em um dos componentes do vetor de condições iniciais $x(0)$, como por exemplo, $x(0) = [0,6 \quad 0,4 \quad 0,21]^T$, o sistema escapa de sua região de estabilidade, conforme mostrado na Figura 5.10.

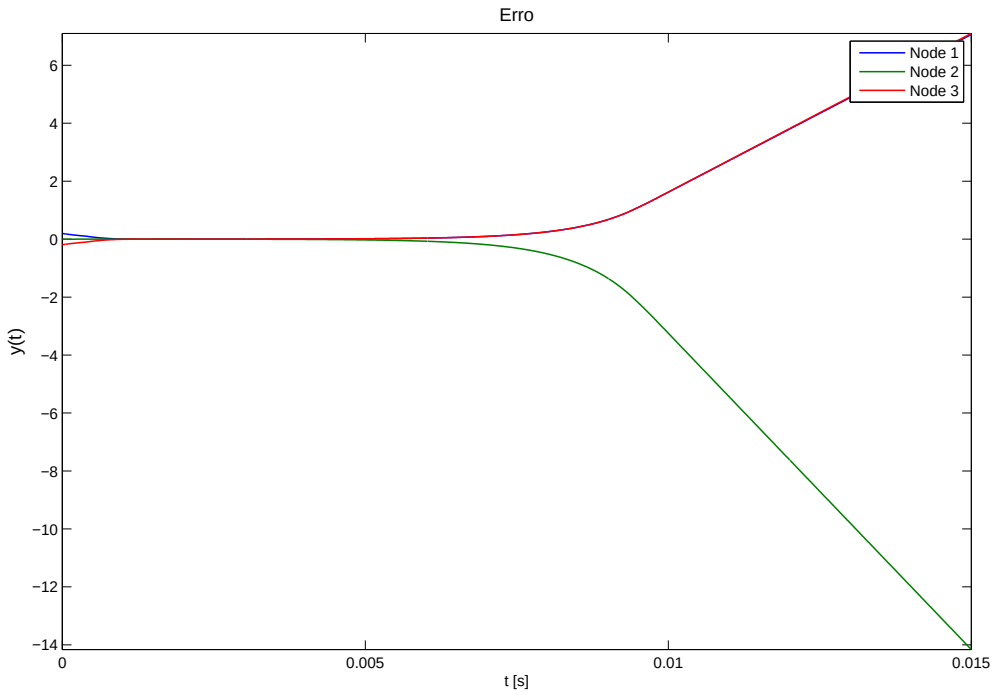

Figura 5.10: Simulação [02]: Resultado da simulação para a matriz de ganhos $K_{[01]}$ com uma nova condição inicial $x(0) = [0,6 \quad 0,4 \quad 0,21]^T$

Isto acontece porque o índice ITAE depende das condições iniciais do problema, uma vez que ele necessita da integração da resposta temporal do sistema, e esta, por sua vez, necessita das condições iniciais para ser determinada.

Tal fato vem corroborar a necessidade de garantir a convergência global, além do desempenho.

Em relação ao tempo utilizado pelo algoritmo genético para determinar a matriz de ganhos $K_{[02]}$, solução do problema dado pela Definição 5.1, o tempo de processamento consumido foi de $57,6s$. Este custo aumenta conforme a ordem do sistema, tornando-o computacionalmente inviável para clusters com um grande número de nós.

A Tabela 5.4 apresenta o custo computacional da primeira proposta em relação à ordem

n	Tempo gasto
3	57,6 s
5	1m 2s
10	20m 40s
20	6h 32m 46s

Tabela 5.4: Custo computacional para a solução do problema dado pela Definição 5.1.

do sistema.

A fim de reduzir o tempo empregado na otimização e garantir a convergência global, vamos buscar um índice de desempenho e uma proposta de solução que evitem a necessidade da resposta temporal do sistema e/ou avaliações baseadas na integração de algum parâmetro.

Definição 5.2 (Síntese de menor norma)**.** *Seja o problema de minimização com restrições dado por*

$$\min_{Z_1} J = -||Z_1||_2 \tag{5.30}$$

sujeito às LMIs

$$\begin{bmatrix} (-\varepsilon D^{-1} - RZ_1 - Z_1^T R + Z_2) & RTZ_1 \\ Z_1^T T^T R & -Z_2 \end{bmatrix} \prec 0 \tag{5.31}$$

$$Z_1 \succ 0 \quad Z_2 \succ 0, \quad D^{-1} \succ 0 \tag{5.32}$$

Encontrar o valor de K que minimize a função custo J.

A escolha de minimizarmos $-||Z_1||$ se justifica devido à equação 5.20 que está diretamente relacionada à matriz de ganhos K, além de reforçar a soma matricial $(-\varepsilon D^{-1} - RZ_1 - Z_1^T R + Z_2)$, fazendo com que esta seja negativa definida, condição esta necessária, porém não suficiente, para que toda a LMI seja definida negativa.

Este problema pode ser resolvido através da solução do problema equivalente de minimização de uma função custo linear sujeita à uma LMI [74, 75]:

$$\min c^T x \tag{5.33}$$

sujeito à

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 10\mu s \quad 10\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0,6 \quad 0,4 \quad 0,2]^T$
λ	$[0 \quad 0 \quad 0]^T$

Tabela 5.5: Parâmetros para a simulação [03]

$$F(x) \preceq 0 \tag{5.34}$$

onde x é o vetor de variáveis. Em muitos problemas de controle, tais funções custo são expressas em termos de variáveis matriciais em vez de um vetor. Exemplos incluem norma(x), onde x é uma variável matricial simétrica [75].

A função *defcx* da *toolbox de controle robusto* do Matlab ajuda a expressar a função custo $J = \text{norma}(Z_1)$ na forma $c^T x$, e a função *mincx* da mesma toolbox resolve o problema de minimização.

O ganho pode ser obtido então a partir da relação (5.20).

5.3.2.3 Simulação [03]: Desempenho com a matriz de ganhos dada pela síntese por otimização de norma

Utilizando-se os mesmos parâmetros da simulação [01] (Tabela 5.1), vamos empregar a Definição 5.2 para determinarmos a matriz de ganhos $K_{[03]}$.

Para o parâmetro $\varepsilon = 10^{-10}$, a matriz de ganhos obtida pela Definição 5.2 é:

$$K_{[03]} = \begin{bmatrix} 14,8127 & 0,0659 & 0,1082 \\ 0,0659 & 14,8299 & 0,0910 \\ 0,1082 & 0,0910 & 14,7876 \end{bmatrix} \tag{5.35}$$

onde $K = Z_1 D^{-1}$.

A Figura 5.11 apresenta os resultados alcançados para o problema de balanceamento de carga para os parâmetros definidos na Tabela 5.6 a partir da solução do problema dado pela Definição 5.2.

Podemos notar que as curvas são suaves, porém a convergência se dá por volta de $0,8s$.

Mais lenta quando comparada com as simulações anteriores.

Comparando os valores da matriz de ganhos anterior $K_{[03]}$ com os valores apresentados para a matriz de ganhos $K_{[02]}$, determinada pelo par índice ITAE + AG, podemos notar que eles são menores. Isto parece apontar para um problema de escala na variável matricial Z_1 ou D^{-1}.

Para a simulação [03], obtivemos:

$$Z_1 = 10^8 \begin{bmatrix} 4,28 & 0,01 & 0,03 \\ 0,01 & 4,29 & 0,02 \\ 0,03 & 0,02 & 4,28 \end{bmatrix}, \quad Z_2 = 10^8 \begin{bmatrix} 3,44 & -1,71 & -1,72 \\ -1,71 & 3,44 & -1,72 \\ -1,72 & -1,72 & 3,45 \end{bmatrix} \tag{5.36}$$

e

$$D^{-1} = 10^{-7} \begin{bmatrix} 2,89 & 0 & 0 \\ 0 & 2,89 & 0 \\ 0 & 0 & 2,89 \end{bmatrix}, \quad Q = 10^{23} \begin{bmatrix} 2,88 & -1,44 & -1,44 \\ -1,44 & 2,88 & -1,44 \\ -1,44 & -1,44 & 2,89 \end{bmatrix} \tag{5.37}$$

Como a matriz Z_1 possui elementos com ordem de grandeza de 10^8 e a matriz D possui elementos com ordem de grandeza de 10^{-7}, os ganhos, obtidos pela equação (5.20) resultaram em elementos com ordem de grandeza de 10^1.

A Figura 5.11 apresenta os resultados da simulação do modelo (2.1) com a matriz de ganhos $K_{[03]}$.

Podemos notar que o sistema converge lentamente para o seu valor de equilíbrio. Outras simulações mostraram que, escalonando a matriz $K_{[03]}$ por um fator de 10^3, a escala de tempo é escalonada por um fator de 10^{-3}.

Outro fato observado foi que, variando-se o parâmetro ε, a matriz de ganhos sofria alguma influência.

Para $\varepsilon = 10^{-8}$, obteve-se:

$$K_{[03]} = \begin{bmatrix} 11,8829 & 0,0387 & 0,0615 \\ 0,0387 & 11,8185 & 0,1259 \\ 0,0615 & 0,1259 & 11,7957 \end{bmatrix} \tag{5.38}$$

Para $\varepsilon = 10^{-12}$, obteve-se:

$$K_{[03]} = \begin{bmatrix} 11,5976 & 0,0603 & 0,0303 \\ 0,0603 & 11,5436 & 0,0843 \\ 0,0303 & 0,0843 & 11,5736 \end{bmatrix} \tag{5.39}$$

Sugerindo que existe um valor ótimo para ε entre 10^{-8} e 10^{-12}. Nas simulações, não houve diferenças significativas entre os resultados para as matrizes de ganho (5.35), (5.38) e (5.39).

Para $\varepsilon \approx 10^{-24}$, começaram a ocorrer problemas numéricos com a LMI, o que a tornava não-factível.

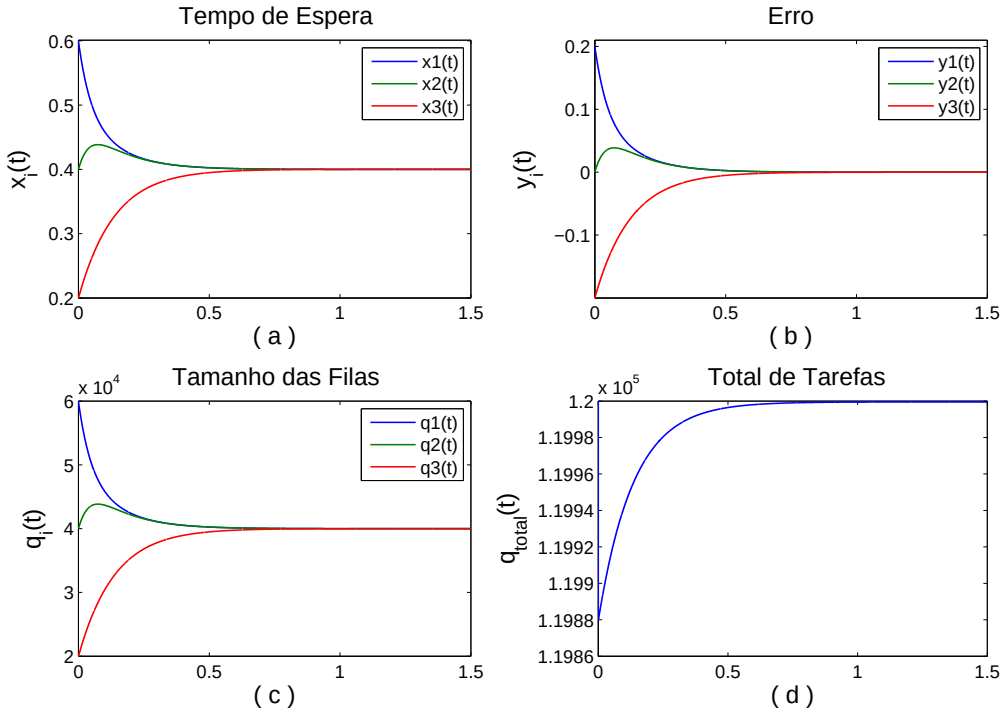

Figura 5.11: Simulação [03]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[03]}$ (5.39).

A vantagem na proposta apresentada pela Definição 5.2 é o custo computacional, que é bem menor do que o da primeira proposta. Para o exemplo de síntese desta simulação, o tempo necessário foi de $0,11s$.

A Tabela 5.6 nos mostra o custo computacional desta proposta para diferentes ordens do sistema.

n	Tempo gasto
3	0,11 s
5	0,12 s
10	0,16 s
15	0,18 s
30	0,32 s
40	1,65 s

Tabela 5.6: Custo computacional para a solução do problema dado pela Definição 5.2.

5.3.3 Formulação Geral de Controle Ótimo - Controladores de Custo Quadrático Garantido

Seja o sistema linear invariante no tempo dado pela equação (3.7), reproduzida aqui por conveniência

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{5.40}$$

e a lei de controle dada por

$$u(t) = Kx(t) \tag{5.41}$$

O problema de controle ótimo quadrático consiste-se em encontrar a matriz de ganhos K que minimiza a função custo quadrático

$$\min_K J = \int_0^\infty [y^T(t)\Omega y(t) + u^T(t)\Psi u(t)]dt \tag{5.42}$$

onde $\Omega \succeq 0$ e $\Psi \succ 0$ são matrizes de ponderação da parte transitória e do sinal de controle, respectivamente [82].

A matriz quadrada simétrica e semi-definida positiva Ω é chamada de *Matriz de Ponderação de Estados*, e a matriz quadrada simétrica e definida positiva Ψ é chamada de *Matriz de Custo de Controle*.

Em TEWARI [83], o autor mostra que a solução da Equação Algébrica de Riccati

$$A^T P_{are} + P_{are}A - P_{are}B\Psi^{-1}B^T P_{are} + \Omega = 0 \tag{5.43}$$

associada ao sistema (5.40), permite calcular o ganho ótimo de K para a lei de controle (5.41) que minimiza a função custo (5.42).

O ganho ótimo é dado então por

$$K = -\Psi^{-1}B^T P_{are} \tag{5.44}$$

onde P_{are} é a única matriz definida positiva que satisfaz a equação (5.43).

Por sua vez, a equação algébrica de Riccati (5.43) pode ser reescrita na forma de uma LMI [82]:

$$\begin{bmatrix} (A^T P_{are} + P_{are} A + \Omega) & P_{are} B \\ B^T P_{are} & \Psi \end{bmatrix} \succeq 0 \tag{5.45}$$

De forma a garantirmos, ao menos, desempenho local para o sistema (3.30), vamos empregar um critério de desempenho quadrático dado pela função custo (5.42) aplicado ao modelo versão linear na variável y sem atraso (3.20). Tal função custo poderá ser minimizada determinando o ganho K pela equação (5.44). No entanto, para isto, necessitaremos da única matriz definida positiva P_{are} solução da equação de Riccati (5.43). Equação esta que pode ser reescrita na forma de uma LMI (5.45).

Isto deverá nos fornecer um bom desempenho local para o modelo de rede neural com atraso quando a lei de controle $v_i(t)$ estiver atuando na região linear - região I da Figura 2.2. Quando a lei de controle estiver atuando fora da região linear, a condição de estabilidade $Z_c \prec 0$ deverá garantir que o sistema não se torne instável.

Este objetivo deverá ser alcançado se for possível acoplar e resolver as duas LMIs simultaneamente: A primeira garantindo o desempenho local (Z_d), e a segunda garantindo a convergência global (Z_c).

Para que possamos resolver as duas LMIs simultaneamente, poderemos utilizar a seguinte estrutura [74, 75]:

$$\left[\begin{array}{c|c} Z_d(P_{are}, Z_1, Z_3) & 0 \\ \hline 0 & Z_c(P_{are}, Z_1, Z_2) \end{array}\right] \prec 0 \tag{5.46}$$

Relembrando o modelo (3.20), reescrito aqui por questões de conveniência, temos:

$$\dot{y}(t) = (0 + R(-I + T)K)y(t) \tag{5.47}$$

onde podemos definir duas matrizes A_ε e B_ε, como sendo $A_\varepsilon = 0$ e $B_\varepsilon = R(-I + T)$.

Substituindo a equação (5.47) em (5.45), teremos:

$$\begin{bmatrix} (A_\varepsilon^T P_{are} + P_{are} A_\varepsilon + \Omega) & P_{are} B_\varepsilon \\ B_\varepsilon^T P_{are} & \Psi \end{bmatrix} \succeq 0 \tag{5.48}$$

onde redefinimos $A_\varepsilon = \varepsilon I$, $\varepsilon \approx 0$ para permitirmos a existência de uma solução.

Invertendo os sinais da LMI (5.48) e aplicando sobre ela a seguinte transformação de

congruência: $blocodiag(I, \Psi^{-1})$, podemos readequá-la para as mesmas variáveis da LMI (5.21), operação esta fundamental para que ambas possam ser resolvidas simultaneamente e de forma acoplada.

$$\begin{bmatrix} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & -P_{are} B_\varepsilon \Psi^{-1} \\ -\Psi^{-1} B_\varepsilon^T P_{are} & -\Psi^{-1} \end{bmatrix} \prec 0 \tag{5.49}$$

Podemos notar que o elemento $(2,1)$ e seu simétrico $(1,2)$ de (5.49) correspondem ao ganho ótimo da equação (5.44) e ao ganho ótimo transposto, respectivamente. Substituindo $-\Psi^{-1} B_\varepsilon^T P_{are}$ por K em (5.49) e realizando a próxima transformação de congruência com $blocodiag(I, D^{-1})$, temos:

$$\begin{bmatrix} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & K^T D^{-1} \\ D^{-1} K & -D^{-1} \Psi^{-1} D^{-1} \end{bmatrix} \prec 0 \tag{5.50}$$

Aplicando mais uma transformação de congruência, desta vez com a matriz bloco diagonal $blocodiag(I, D^{-1} K^T K^{-1} D)$, temos:

$$\begin{bmatrix} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & K D^{-1} \\ D^{-1} K^T & -D^{-1} K^T K^{-1} \Psi^{-1} (K^{-1})^T K D^{-1} \end{bmatrix} \prec 0 \tag{5.51}$$

Aplicando a seguinte seqüência de substituição de variáveis em (5.51):

$$Z_1 = K D^{-1} \tag{5.52}$$

$$Z_3 = D^{-1} K^T K^{-1} \Psi^{-1} (K^{-1})^T K D^{-1} \tag{5.53}$$

Aplicando uma nova substituição de variáveis em (5.53):

$$Z_3 = \Gamma Z_3' \tag{5.54}$$

temos:

$$Z_d = \begin{bmatrix} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & Z_1 \\ Z_1^T & -\Gamma Z_3' \end{bmatrix} \prec 0 \tag{5.55}$$

onde $Z_3' \succ 0$ é uma matriz simétrica a ser determinada e $\Gamma \succ 0$ é uma matriz simétrica constante, que vem devolver a possibilidade de escalonarmos a matriz Ψ^{-1} indiretamente, uma vez que, após todas as transformações de congruência e substituição de variáveis, a matriz Ψ deixou de ser acessível.

Para permitir o acoplamento da LMI (5.55) com a LMI (5.21), vamos definir que:

$$P_{are} = D^{-1} \tag{5.56}$$

Desta forma, teremos:

$$\left[\begin{array}{cc|cc} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & Z_1 & 0 & 0 \\ Z_1^T & -\Gamma Z_3' & 0 & 0 \\ \hline 0 & 0 & -\varepsilon P_{are} - RZ_1 - Z_1^T R + Z_2 & RZ_1 \\ 0 & 0 & Z_1^T T^T R & -Z_2 \end{array}\right] \prec 0 \tag{5.57}$$

onde

$$Z_1 = K P_{are} \tag{5.58}$$

$$Z_2 = P_{are} \Psi P_{are} \tag{5.59}$$

A matriz Ξ (5.57) é formada por dois blocos diagonais: O primeiro representando o desempenho, onde encontramos apenas o ganho. No segundo bloco, temos a estabilidade do modelo (3.30).

Encontrar uma solução para a LMI Ξ significará encontrar uma matriz de ganhos que minimize a função custo (5.42) simultaneamente às condições de estabilidade global de (5.21).

Desta forma, podemos enunciar a próxima metodologia para a síntese de controladores empregando LMIs de forma acoplada.

Definição 5.3 (Síntese por Função Custo Quadrático)**.** *Seja a LMI bloco diagonal*

$$\Xi = \left[\begin{array}{cc|cc} (-A_\varepsilon^T P_{are} - P_{are} A_\varepsilon - \Omega) & -Z_1 & 0 & 0 \\ -Z_1^T & -\Gamma Z_3' & 0 & 0 \\ \hline 0 & 0 & -\varepsilon P_{are} - RZ_1 - Z_1^T R + Z_2 & RZ_1 \\ 0 & 0 & Z_1^T T^T R & -Z_2 \end{array}\right] \prec 0 \tag{5.60}$$

e as matrizes simétricas $\Omega \succeq 0$ *e* $\Gamma \succ 0$*. Encontrar as matrizes simétricas definidas positivas* P_{are}*,* Z_1*,* Z_2 *e* Z_3 *de forma que a LMI seja definida negativa.*

Após as transformações de congruência e substituição de variáveis, o acesso direto à Matriz de Custo de Controle Ψ não existe mais. Desta forma, poderemos ter um acesso indireto à esta matriz através de Γ que irá escalonar a variável Z_3 e, desta forma, estará escalonando indiretamente a matriz Ψ.

Esta metodologia de síntese recai no chamado *feasibility problem*, e que pode ser resolvido pelo Método dos Pontos Interiores de Nesterov e Nemirovsky [76].

Dadas as matrizes Ω e Γ, o problema se resume a encontrar as matrizes P_{are}, Z_1, Z_2 e Z_3 de modo que a LMI seja definida negativa.

5.3.3.1 Simulação [04]: Desempenho com a matriz de ganhos dada pela síntese por função custo quadrático - o caso homogêneo

O objetivo desta simulação é o de demonstrar um primeiro resultado alcançado pela síntese por função de custo quadrático via LMI para um cluster homogêneo. Para ilustrar a metodologia proposta pela Definição 5.3, vamos retornar aos parâmetros dados na simulação [01]. A Tabela 5.7 apresenta os parâmetros utilizados.

O valor do parâmetro ε foi ajustado para 10^{-14}. Para valores menores a LMI Ξ torna-se inviável (não possui solução).

A matriz de ganhos K depende numericamente do parâmetro ε, mas existe um valor de compromisso pois é desejado que este parâmetro seja suficientemente pequeno e que ainda permita que a LMI Ξ seja factível. Em todas as simulações, o parâmetro ε foi mantido o mínimo possível para que a LMI fosse viável.

Como não foi empregado nenhum método de otimização para encontrar bons valores

Parâmetro	**Valor**
n	3
t_p	$[10\mu s \quad 10\mu s \quad 10\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0,6 \quad 0,4 \quad 0,2]^T$
λ	$[0 \quad 0 \quad 0]^T$
ε	10^{-14}
Ω	$diag(1,365 \times 10^{-6} \quad 1,365 \times 10^{-6} \quad 1,365 \times 10^{-6})$
Γ	$diag(10^{-10} \quad 10^{-10} \quad 10^{-10})$

Tabela 5.7: Parâmetros para a simulação [04]

para os parâmetros Ω e Γ, estas matrizes foram inicialmente arbitradas como sendo a matriz identidade. Ao longo das simulações, Γ manteve-se como a identidade, porém, escalonada por um valor de 10^{-10}, para manter a simplicidade.

A matriz Ω foi escalonada pelo valor de $1,365 \times 10^{-6}$ por tentativa e erro.

Utilizando a *toolbox* de LMI do Matlab, a matriz $K_{[04]}$ foi determinada em $219,4\ ms$, e é apresentada a seguir:

$$K^1_{[04]} = \begin{bmatrix} 3415,6 & -460,1 & -460,1 \\ -460,1 & 3415,6 & -460,1 \\ -460,1 & -460,1 & 3415,6 \end{bmatrix} \tag{5.61}$$

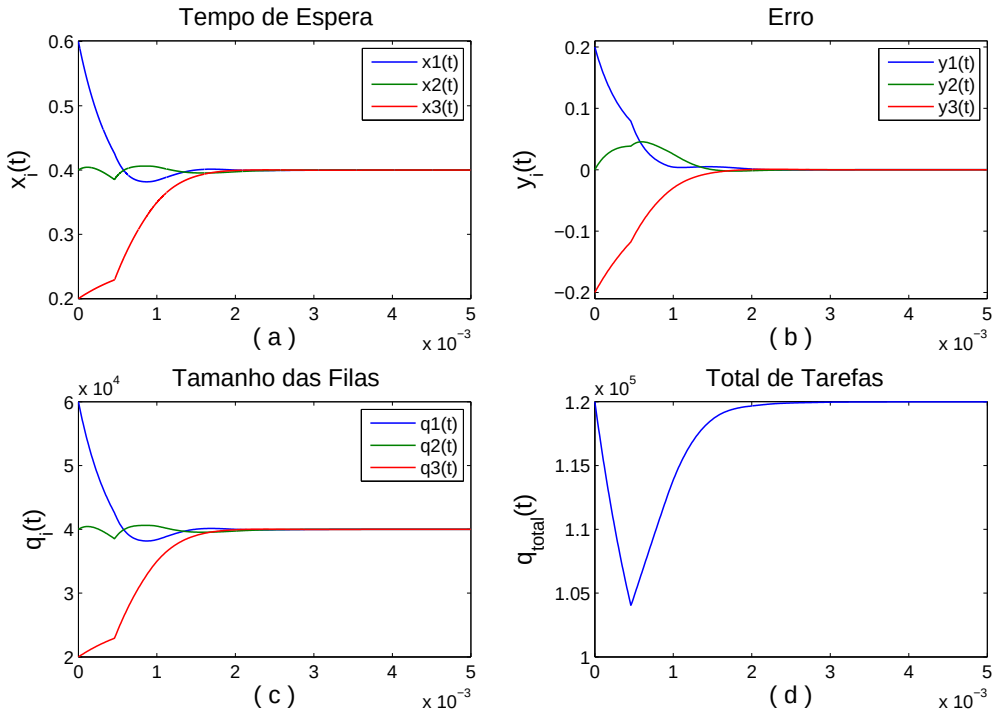

Figura 5.12: Simulação [04]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[04]}$ (5.61).

No caso da matriz de ganhos $K_{[04]}$ (5.61), podemos notar que é uma matriz simétrica cujo elementos da diagonal principal são iguais. Isto se deve pelo fato do exemplo ser um cluster homogêneo, o que leva a matriz T a ser simétrica, e pela escolha das matrizes Ω e Γ.

Como exemplo de que nem sempre a matriz de ganhos K obtida é simétrica, vamos realizar uma outra escolha para as matrizes Ω e Γ.

$$\Omega = 10^{-8}\begin{bmatrix} 10,28 & -6,01 & 5,81 \\ -6,01 & 12,35 & 3,10 \\ 5,81 & 3,10 & 14,73 \end{bmatrix}, \quad \Gamma = 10^{-15}\begin{bmatrix} 12,28 & -6,01 & 8,81 \\ -6,01 & 14,35 & 5,10 \\ 8,81 & 5,10 & 16,73 \end{bmatrix} \tag{5.62}$$

Para estas escolhas de Ω e Γ, a solução do problema dado pela Definição 5.3 é:

$$K^2_{[04]} = \begin{bmatrix} 1145,4 & -613,8 & 475,2 \\ -626,4 & 1370,6 & 260,8 \\ 477,7 & 256,9 & 1207,8 \end{bmatrix} \tag{5.63}$$

Os resultados da simulação são, de uma forma em geral, apresentados na Figura 5.12. A

	Matriz de Ganhos $K_{[c]}$			Matriz de Ganhos $K_{[04]}$		
Node	M_p	M_t	t_s	M_p	M_t	t_s
Node 1	-0,019	0,83 ms	3,95 ms	-3,66E-4	2,49 ms	2,66 ms
Node 2	0,053	0,47 ms	3,34 ms	0,045	0,60 ms	3,88 ms
Node 3	0,006	1,13 ms	4,12 ms	7,77E-4	2,23 ms	3,34 ms

Tabela 5.8: Parâmetros de desempenho entre as simulações $[01]$ e $[04]$

Figura 5.13 nos mostra uma ampliação das Figuras 5.8a e 5.12b.

Comparando a Figura 5.13a com a Figura 5.13b, é possível notar que Figura 5.13b apresenta curvas com overshoots e settling time menores.

A Figura 5.13a é a resposta do modelo (2.1) para a matriz de ganhos $K_{[c]}$ (5.27), cujo gráfico apresenta um Maximum Overshoot de $0,053$ e um settling time de $4,12$ ms, enquanto a Figura 5.13b é a resposta do modelo (2.1) para a matriz de ganhos $K_{[04]}$ (5.61), cujo gráfico apresenta um Maximum Overshoot de $0,045$ e um settling time de $3,88$ ms.

Como as matrizes Ω e Γ não são matrizes ótimas, então existe espaço para otimização sobre estes parâmetros.

A importância da redução dos overshoots reside no fato do tráfego na rede diminuir, reduzindo assim os custos de comunicação. Tais custos assumem uma importância maior para os chamados *grids* de processadores, cuja infra-estrutura de comunicação cobre grandes distâncias geográficas e, muitas vezes, toda ou parte desta infra-estrutura é tarifada pelo volume de dados trafegado, como no caso das redes Frame-Relays [84].

A menor atividade de transferência de carga pode ser confirmada examinando-se o gráfico de Quantidade de Tarefas entre as Figuras, 5.6d e 5.12d. Na Figura 5.12d, é possível notar que o vale possui uma amplitude menor do que o seu correspondente na Figura 5.6d, significando que um número menor de tarefas foram transferidas, 16.000 tarefas transferidas para o gráfico 5.12d contra 33.000 tarefas transferidas no gráfico 5.6d.

Os ganhos de $K_{[04]}$ também são menores, quando comparados com os ganhos de $K_{[01]}$.

Para flexibilizar a síntese de controladores, a opção de matriz não-diagonal foi empregada para arbitrar Ω e Γ mas houve grande dificuldade em ajustar manualmente os valores do seus elementos de forma a melhorar o desempenho do sistema. No entanto, as repetidas simulações com diferentes parâmetros mostraram indícios de que as matrizes Ω e Γ na sua forma não-diagonal levam a melhorias no desempenho e na redução do chamado efeito de

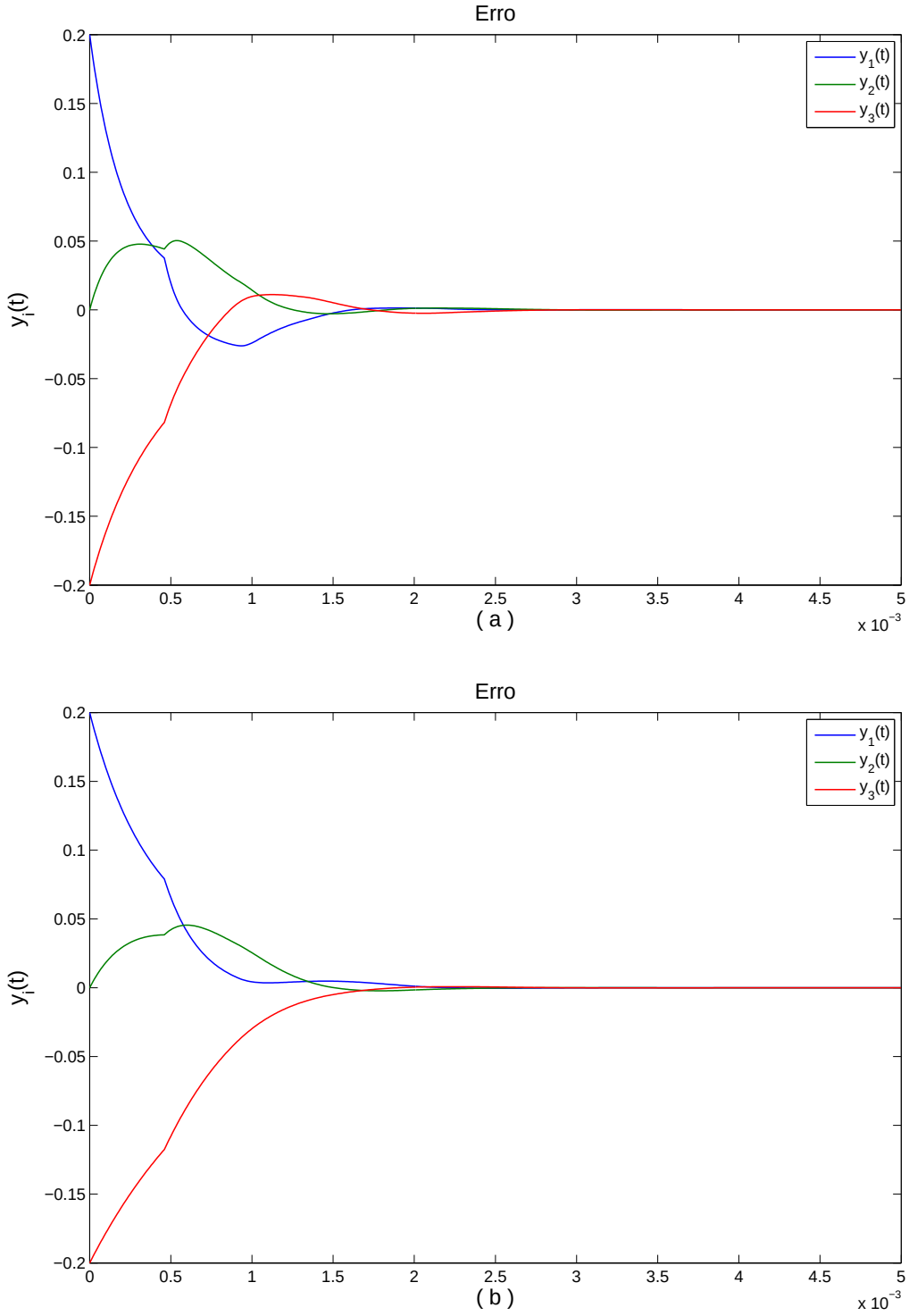

Figura 5.13: Simulação [04]: Comparação entre os sinais de erro das simulações $[01]$ e $[04]$. O gráfico (a) apresenta os sinais de erro com os ganhos dados por $K_{[c]}$ (5.27), e o gráfico (b) apresenta os sinais de erro com os ganhos dados por $K_{[04]}$ (5.61).

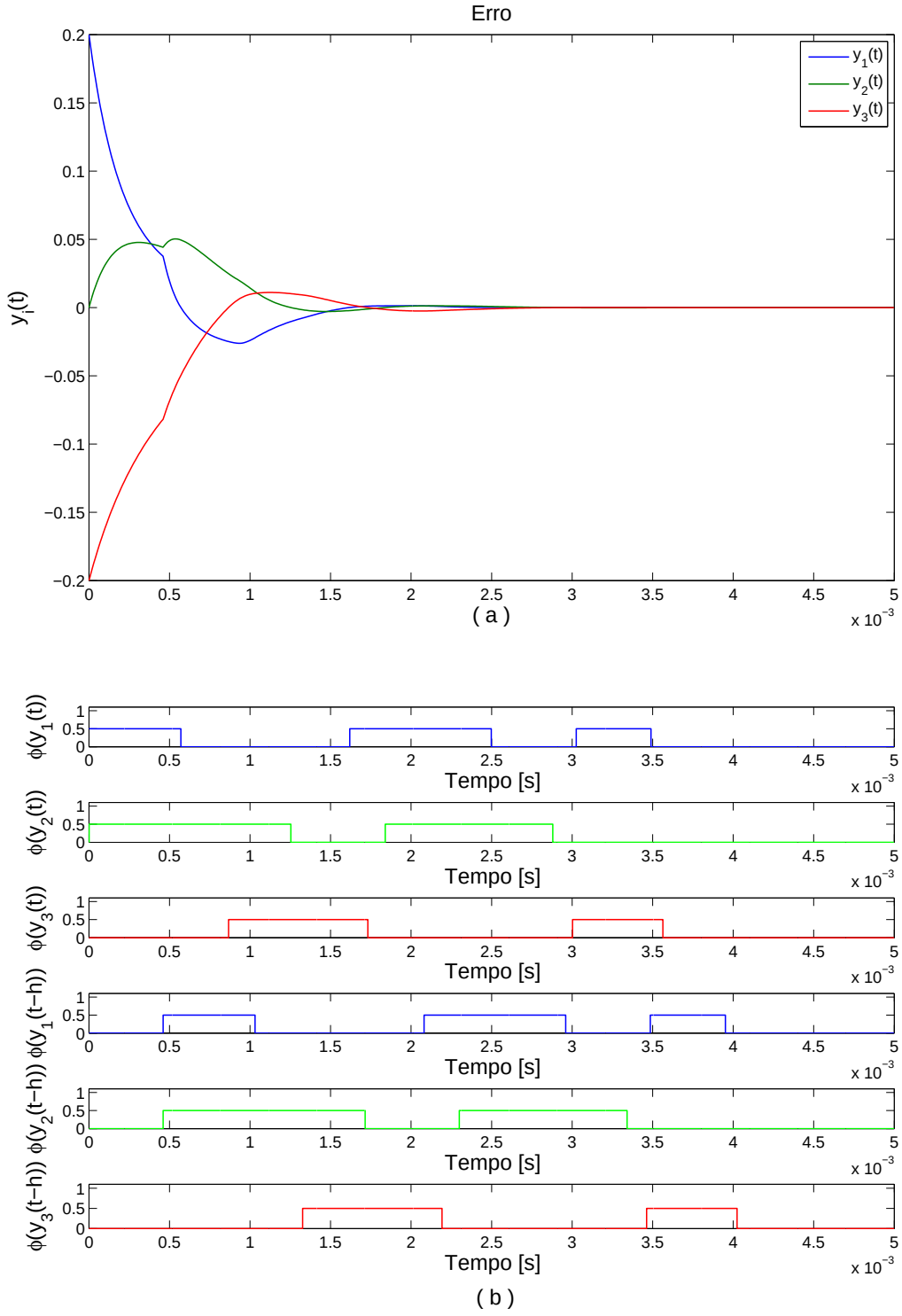

Figura 5.14: Simulação [04]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[c]}$ (5.27). O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

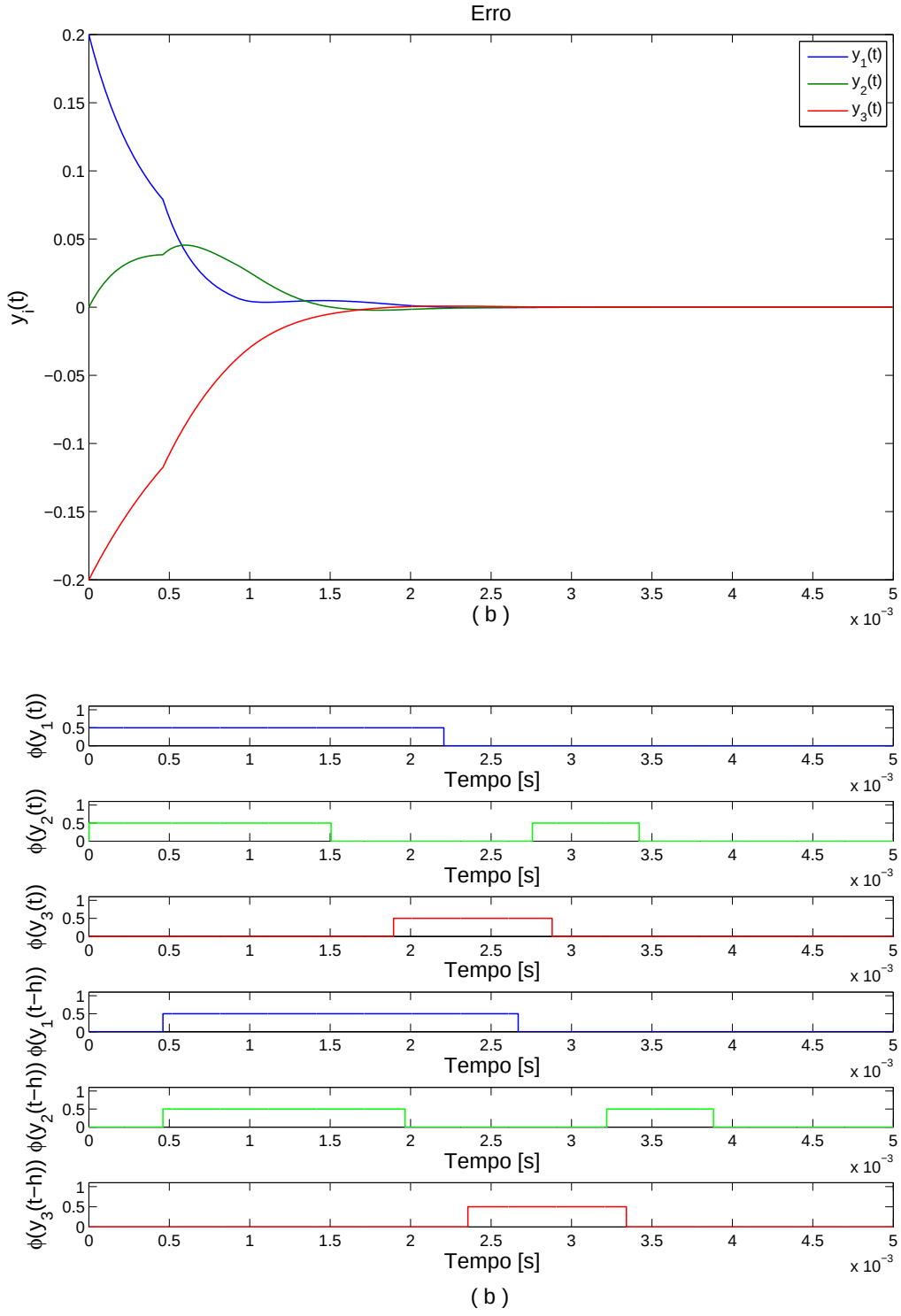

Figura 5.15: Simulação [04]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[04]}$ (5.61), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

n	tempo
3	0,17 s
4	0,19 s
5	0,22 s
6	0,27 s
7	0,36 s
8	0,52 s
9	0,64 s
10	0,92 s
20	25,9 s
30	4m 15s
40	23m 33s

Tabela 5.9: Custo computacional para a solução do problema dado pela Definição 5.3.

bursting, a ser explicado na simulação [11].

De uma forma em geral, matrizes Ω e Γ que levam a matriz K a possuir elementos fora da diagonal principal com magnitude (em módulo) próximas da dos elementos da diagonal, apresentam melhoria de desempenho.

A Tabela 5.9 apresenta o custo computacional em função do número de nós do cluster para a síntese de controladores através da Definição 5.3.

A Figura 5.16 apresenta a curva do custo computacional em função do número de nós n do cluster.

Apesar do rápido crescimento do custo computacional para encontrar-se uma solução numérica para a solução da LMI (5.60), é possível obter uma escalabilidade melhor para o problema utilizando-se uma formulação paralela e escalonável do método dos pontos interiores [85].

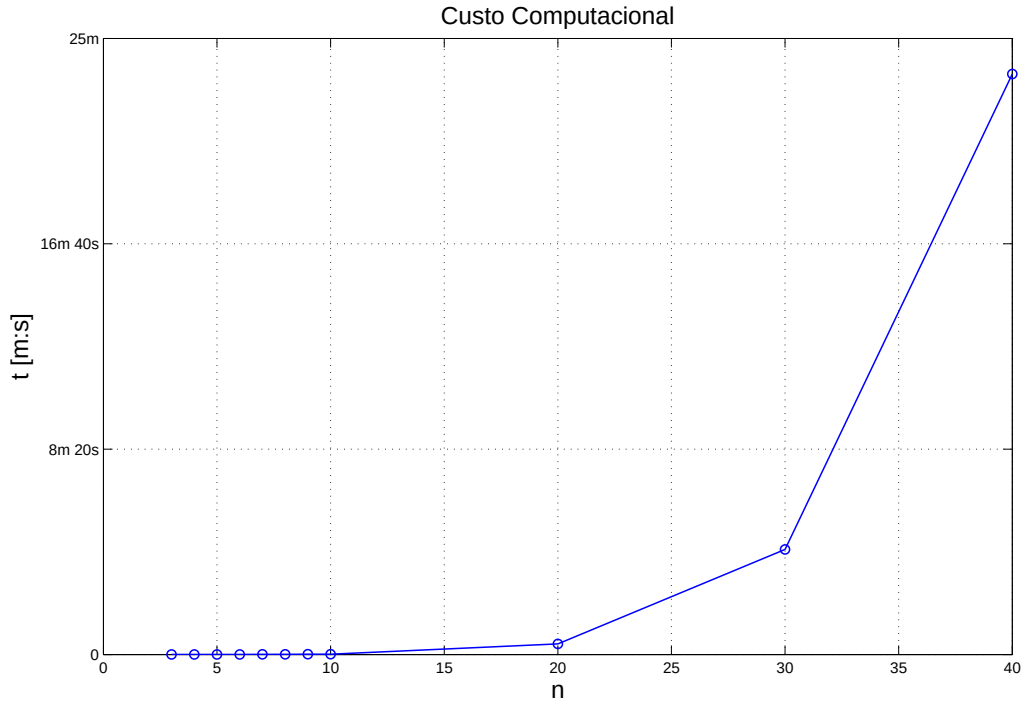

Figura 5.16: Custo computacional da síntese de controladores pela Definição 5.3 em função do número de nós n de um cluster.

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 40\mu s \quad 120\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0,6 \quad 0,4 \quad 0,2]^T$
λ	$[0 \quad 0 \quad 0]^T$
K	$diag(12000 \quad 7500 \quad 9000)$
ε	10^{-14}
Ω	$diag(15 \times 10^{-12} \quad 15 \times 10^{-12} \quad 15 \times 10^{-12})$
Γ	$diag(10^{-10} \quad 10^{-10} \quad 10^{-10})$

Tabela 5.10: Parâmetros para a simulação [05]

5.3.3.2 Simulação [05]: Desempenho com a matriz de ganhos dada pela síntese por função custo quadrático - o caso heterogêneo

O objetivo desta simulação é o de demonstrar resultados de síntese por função custo quadrático utilizando a LMI Ξ, para o caso de um cluster heterogêneo.

Seja um cluster heterogêneo de três nós com parâmetros dados pela Tabela 5.10. Supondo que Δt_i seja o intervalo de tempo médio entre execuções sucessivas do algoritmo de balanceamento de carga no i-ésimo nó, definidos como: $\Delta t_1 = 75\mu s$, $\Delta t_2 = 125\mu s$ e $\Delta t_3 = 100\mu s$, como descrito na subseção 2.2.3, e definindo a matriz de ganhos discreta no tempo $K_z = 0,9$, temos:

$$\begin{aligned} K_{11} &= 0,9/75\mu s = 12000 \\ K_{22} &= 0,9/120\mu s = 7500 \\ K_{33} &= 0,9/100\mu s = 9000 \end{aligned} \tag{5.64}$$

A partir dos ganhos K_{11}, K_{22} e K_{33}, temos a matriz de ganhos $K^1_{[05]}$:

$$K^1_{[05]} = \begin{bmatrix} 12000 & 0 & 0 \\ 0 & 7500 & 0 \\ 0 & 0 & 9000 \end{bmatrix} \tag{5.65}$$

A partir dos parâmetros t_{p_i} e das matrizes Ω e Γ, definidas na Tabela 5.10, determina-se as matrizes R e T de forma a encontrar a matriz de ganhos $K^2_{[05]}$ através da Definição 5.3 - ver seção 5.4.

	Matriz de Ganhos $K^1_{[05]}$			Matriz de Ganhos $K^2_{[05]}$		
Node	M_p	M_t	t_s	M_p	M_t	t_s
Node 1	-0,4806	0,64 ms	7,76 ms	-0,0054	1,32 ms	3,32 ms
Node 2	0,1475	0,96 ms	7,78 ms	0,0381	0,44 ms	3,32 ms
Node 3	0,4477	0.57 ms	7,76 ms	0,0097	1,26 ms	2,08 ms

Tabela 5.11: Parâmetros de desempenho entre duas matrizes de ganhos $K^1_{[05]}$ (5.65) e $K^2_{[05]}$ (5.66) para a simulação [05].

Desta forma, a matriz de ganhos $K^2_{[05]}$ é definida como:

$$K^2_{[05]} = \begin{bmatrix} 925,0 & -1582,0 & -221,1 \\ -1585,9 & 6079,3 & -684,1 \\ -223,1 & -688,6 & 6346,1 \end{bmatrix} \tag{5.66}$$

A Figura 5.17 apresenta o sinal de erro $y(t)$ da resposta do modelo (2.1) para a matriz de ganho $K^1_{[05]}$, determinada segundo CHIASSON *et al.* [33].

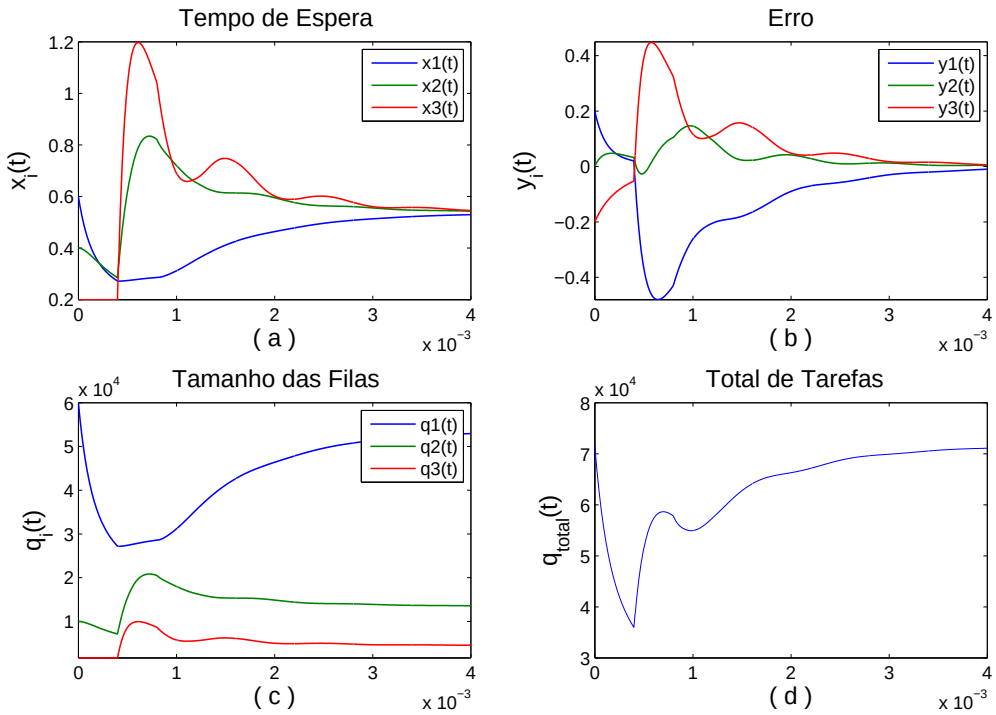

Figura 5.17: Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^1_{[05]}$ (5.65).

A Figura 5.19 apresenta os sinais de erro $y(t)$ para a matriz de ganhos $K^1_{[05]}$ e a atividade

dos controladores.

A Figura 5.18 apresenta o sinal de erro $y(t)$ da resposta do modelo (2.1) para a matriz de ganho $K^2_{[05]}$, solução do problema dado pela Definição 5.3.

No gráfico 5.18a, as curvas tendem para o ponto de equilíbrio cujo valor é $0,5379s$, e no gráfico 5.18c, o tamanho das filas são diferentes pois o processador mais rápido (nó 1) recebe uma carga de tarefas maior, ficando com uma fila maior também. Já o processador do nó 3, por ser o mais lento, permanece com a menor quantidade de tarefas.

A Figura 5.20 apresenta os sinais de erro $y(t)$ para a matriz de ganhos $K^2_{[05]}$ e a atividade dos controladores.

Através de comparações dos resultados, podemos verificar que a síntese por função custo quadrático, conforme dada pela Definição 5.3, apresenta bons resultados. Vale ressaltar ainda que, mais uma vez, as matrizes Ω e Γ não sofreram nenhum processo de otimização. Suas escolhas foram baseadas em poucas tentativas e erros.

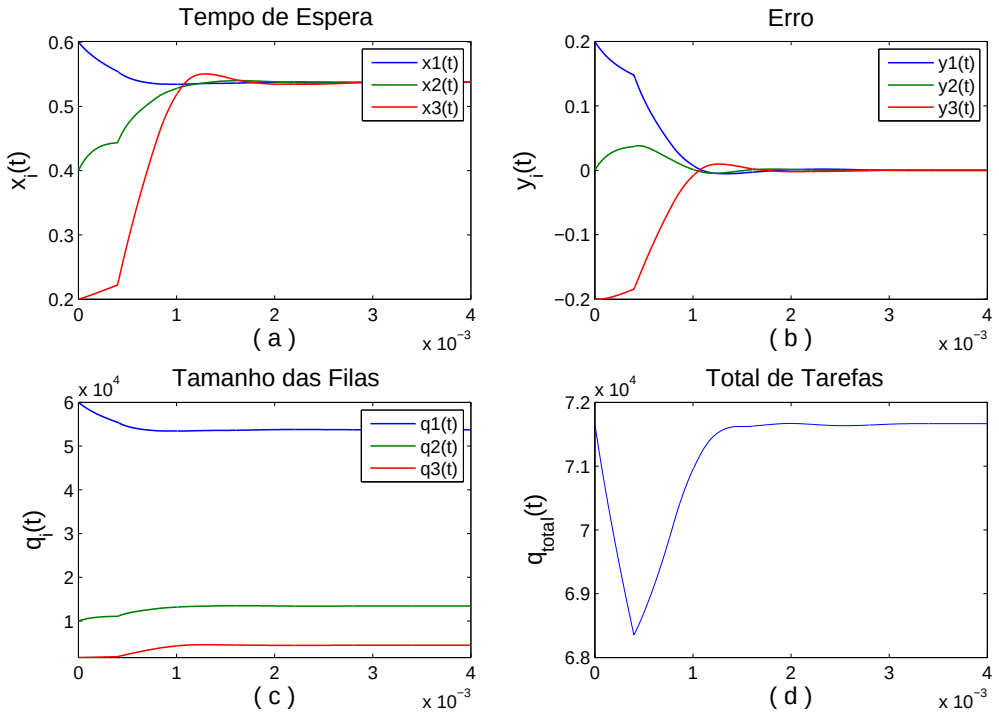

Figura 5.18: Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^2_{[05]}$ (5.66).

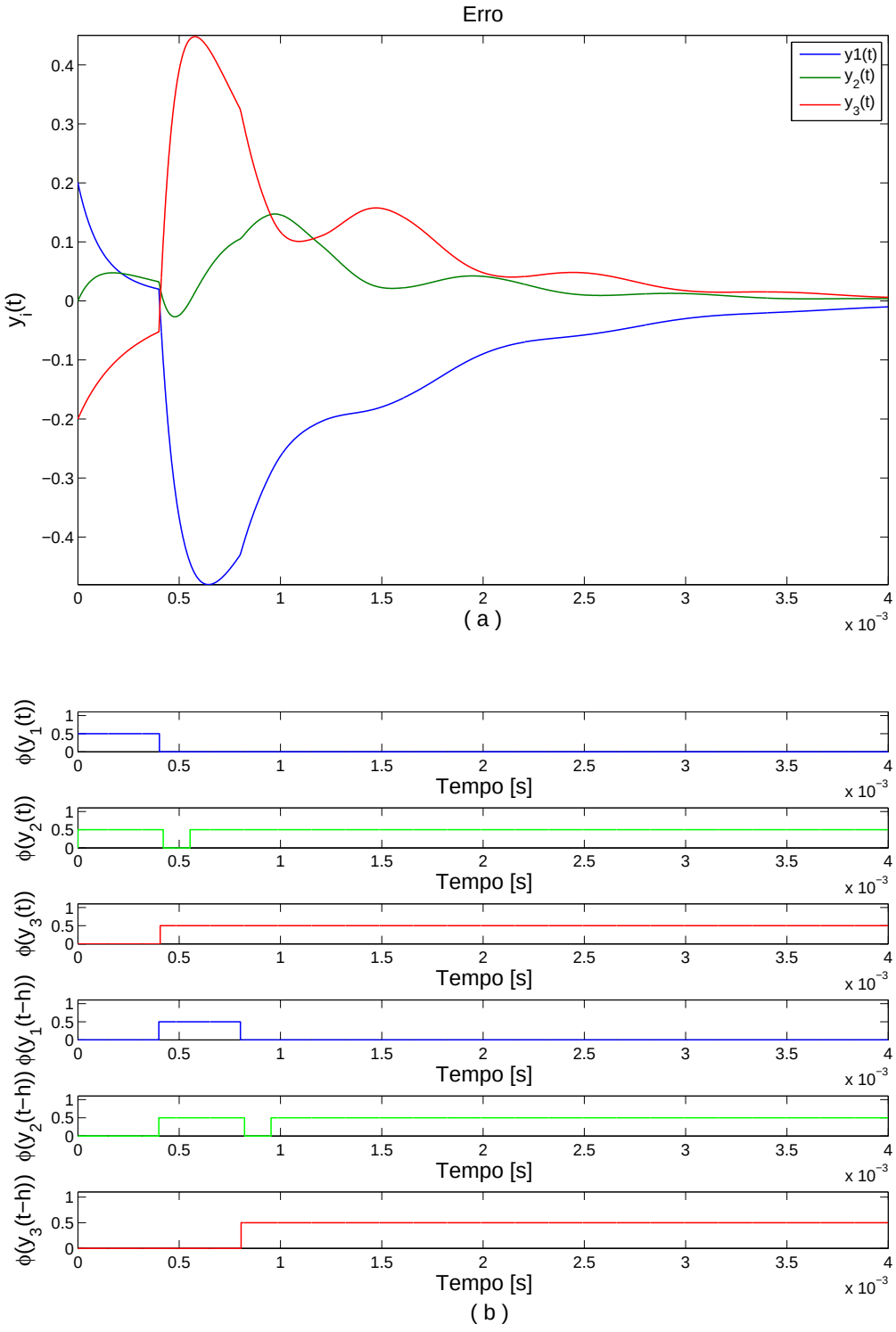

Figura 5.19: Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K^1_{[05]}$ (5.66), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

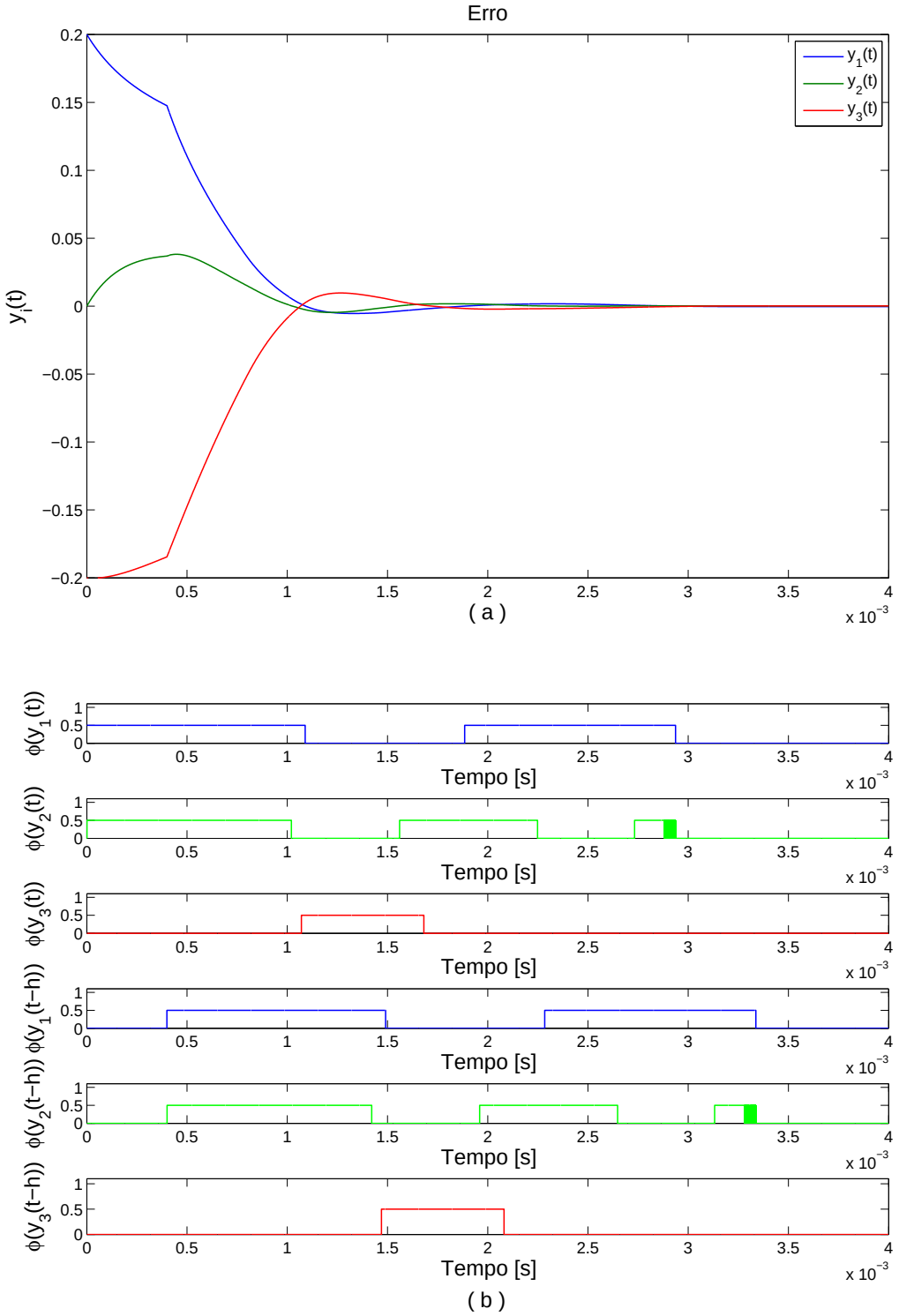

Figura 5.20: Simulação [05]: Resposta do modelo (2.1) para os ganhos dado pela matriz $K_{[04]}$ (5.61), O gráfico (a) apresenta os sinais de erro $y(t)$ e o gráfico (b) apresenta a atividade dos controladores conforme definido em (5.7).

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 40\mu s \quad 120\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0 \quad 0 \quad 0]^T$
λ	$[50 \quad 25 \quad 5]^T$
t_λ	10 ms
y_{max}	1,3

Tabela 5.12: Parâmetros para a simulação [06]

5.3.3.3 Simulação [06]: Resposta ao pulso

O objetivo desta simulação é comparar as respostas do sistema a uma entrada em pulsos, conforme definido na seção 5.2, utilizando as matrizes de ganhos $K_{[c]}$ (5.27) e $K_{[04]}$ (5.61), reapresentadas aqui por questões de conveniência.

$$K_{[c]} = \begin{bmatrix} 6667 & 0 & 0 \\ 0 & 4167 & 0 \\ 0 & 0 & 5000 \end{bmatrix}, \quad K_{[04]} = \begin{bmatrix} 3415,6 & -460,1 & -460,1 \\ -460,1 & 3415,6 & -460,1 \\ -460,1 & -460,1 & 3415,6 \end{bmatrix} \tag{5.67}$$

Os parâmetros de simulação utilizados são apresentados na Tabela 5.12 e os resultados da simulação para as matrizes $K_{[c]}$ e $K_{[04]}$ são apresentados, respectivamente, nas Figuras 5.21 e 5.22. As Figuras 5.23 e 5.24 destacam o erro do sinal $y(t)$ em relação à atividade dos sinais de controle.

Tanto na Figura 5.21a quanto na Figura 5.22a é possível verificar que o sistema parte de condições iniciais nulas, e o Tempo de Espera de cada uma das filas cresce com taxas diferentes nos primeiros milisegundos, e depois a taxa permanece a mesma para os três nós.

Isto ocorre porque sendo o sistema dinâmico, o pulso gera uma resposta transitória (como se fosse uma resposta ao degrau) que demanda alguns milisegundos para o sistema acompanhá-lo. Quando o pulso cessa a sua duração, o sistema passa a comportar-se como um sistema sem entradas e com condições iniciais não nulas.

Na Figura 5.21b, é possível notar que, enquanto o erro do nó 1 e nó 2 são positivos e crescentes, o erro do nó 3 é negativo e crescente.

Isto ocorre porque no momento inicial $t = 0s$, temos que:

$$\begin{bmatrix} \dot{x}_1(0) \\ \dot{x}_2(0) \\ \dot{x}_3(0) \end{bmatrix} = \begin{bmatrix} 50 \\ 25 \\ 5 \end{bmatrix} \tag{5.68}$$

e a média das filas no instante inicial é dada por $(50 + 25 + 5)/3 \approx 26,6$. Desta forma, temos que para o nó 1: $r(0) = (50 - 26,6) = 23,4$, logo, $y_1(0) > 0$. Para o nó 2: $(25 - 26,6) = -1,6$, logo, $y_2(0) < 0$. Para o nó 3: $(5 - 26,6) = -21,6$, logo, $y_3(0) < 0$.

No instante imediatamente posterior, somente o nó1 estará enviando tarefas para os outros nós. Como o nó 2 está um pouco abaixo da média $r(0)$, após a recepção das tarefas enviadas a ele, ele passará a um novo estado onde ele estará acima da média, fazendo com que ele diminua seu erro e tenda a aproximar-se de zero. Em outras palavras, isto fará com que a curva de erro do nó 2 na Figura 5.21b inicie-se com com valor negativo e volte a ser positivo em seguida. Isto pode ser confirmado se ampliarmos a Figura 5.21b em torno da origem ou olhando a Figura 5.23b para os sinais de controle $\phi(y_2(t))$ e $\phi(y_2(t-h))$, onde os sinais inicialmente estão no valor nulo e mudam de estado nos instantes $9,68 \times 10^{-5}s$ e $4,98 \times 10^{-4}s$, respectivamente.

Podemos concluir deste experimento que aplicando as mesmas entradas, o sistema responde de forma diferente em função da matriz de ganhos e pudemos observar qual o comportamento do sistema.

No caso da matriz de ganhos $K_{[04]}$, apesar dos ganhos dentro da matriz serem menores que os da matriz de ganhos $K_{[c]}$, os sinais de erro atingiram valores maiores.

Esta mesma simulação, quando realizada para clusters heterogêneos, apresenta um comportamento similar, sendo a maior diferença o fato de que, para as curvas de Tempo de Espera, elas deixam de crescer de forma aproximadamente paralela e passam a crescer em taxas diferentes em função da velocidade de processamento de cada nó.

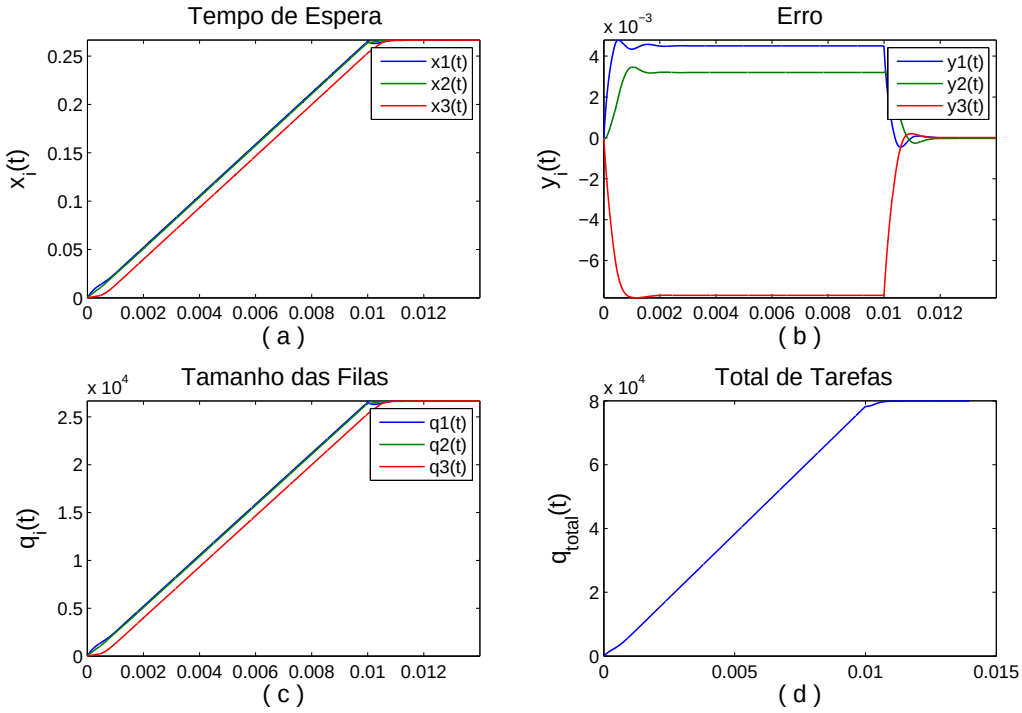

Figura 5.21: Simulação [06]: Resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz $K_{[c]}$ (5.27).

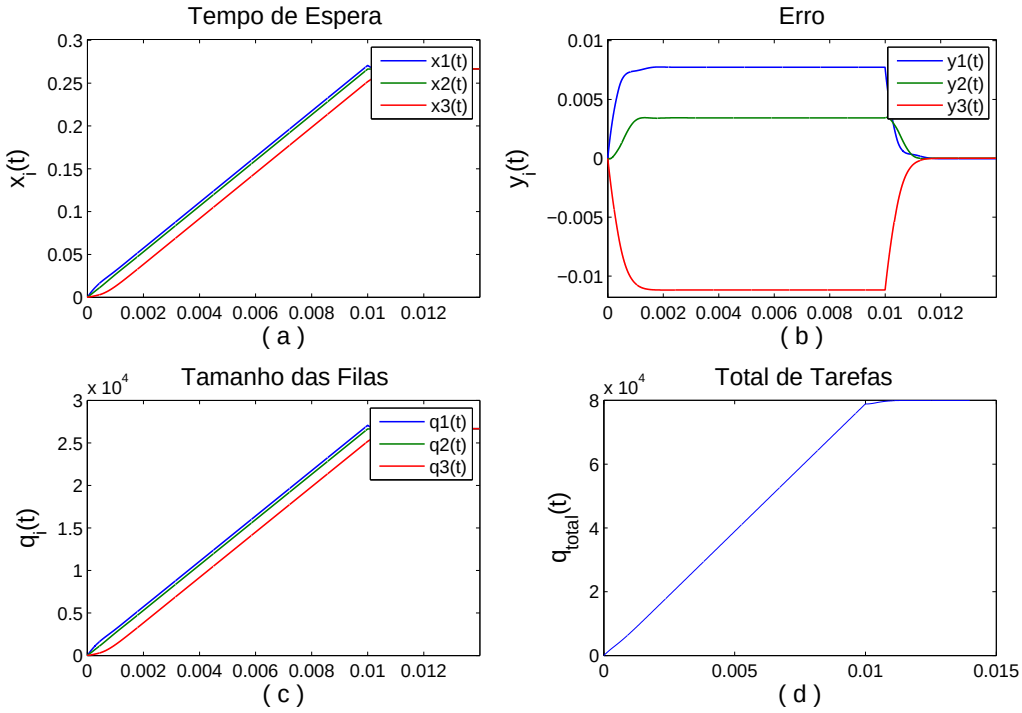

Figura 5.22: Simulação [06]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[04]}$ (5.61).

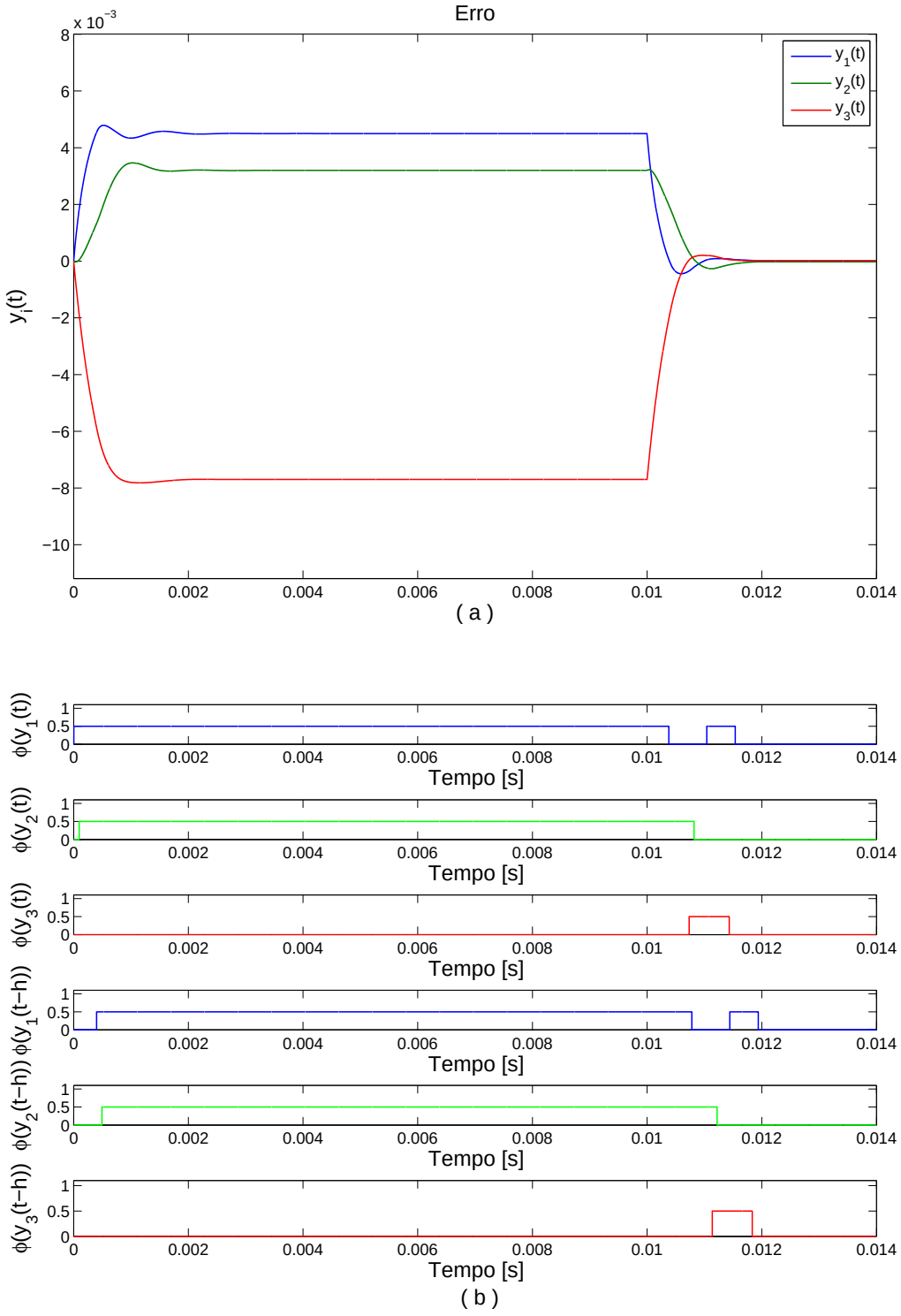

Figura 5.23: Simulação [06]: O gráfico (a) mostra a resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz de ganhos $K_{[c]}$ (5.27). O gráfico (b) mostra a atividade dos controladores conforme definido em (5.7).

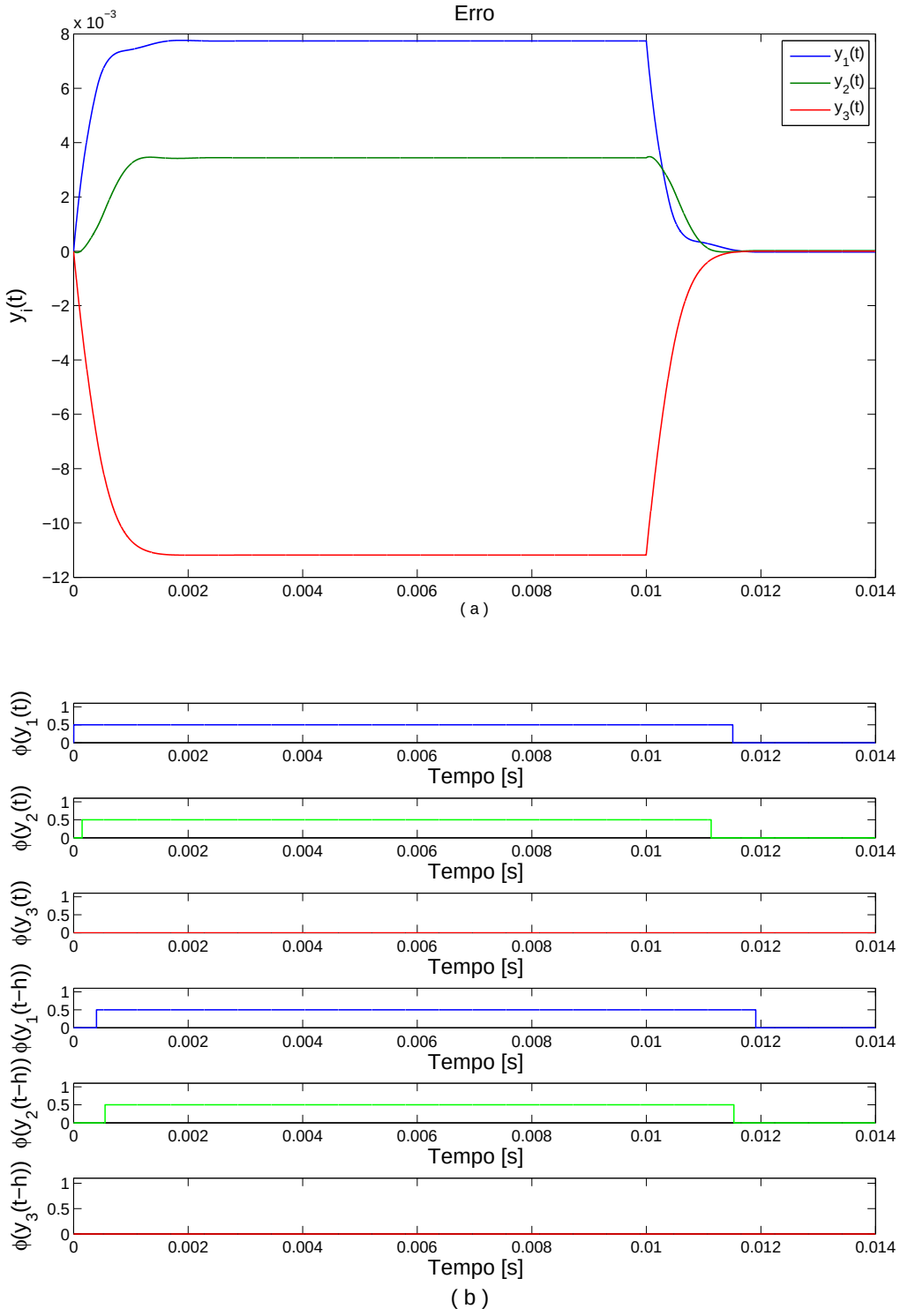

Figura 5.24: Simulação [06]: O gráfico (a) mostra a resposta do modelo (2.1) ao pulso $\lambda = [50 \quad 25 \quad 5]^T$, $t_\lambda = 10$ ms e condições iniciais nulas para a matriz de ganhos $K_{[04]}$ (5.61). O gráfico (b) mostra a atividade dos controladores conforme definido em (5.7).

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 40\mu s \quad 120\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0 \quad 0 \quad 0]^T$
λ	$[750 \quad 50 \quad 5]^T$
t_λ	10 ms
y_{max}	1,3
K	$diag(3997,5 \quad 999,4 \quad 333,1)$

Tabela 5.13: Parâmetros para a simulação [07]

5.3.3.4 Simulação [07]: Efeito do limite da largura de banda

A fim de expor com maiores detalhes a ação dos controladores, a simulação [07] possui como objetivo demonstrar o efeito do limite da largura de banda de uma rede sobre o balanceamento de carga. Limite este relacionado ao modelo (2.1) pelo parâmetro y_{max} da função não-linear $\phi(y(t))$ (2.6).

Os parâmetros de simulação utilizados são apresentados na Tabela 5.13.

A Figura 5.25 apresenta os resultados da simulação, onde podemos observar o rápido crescimento do número de tarefas presentes em cada uma das fila pelo gráfico Tamanho das Filas. Como o processador do nó 1 é o mais rápido ($t_{p_1} = 10\mu s$), cabe a ele uma maior quantidade de carga. O oposto acontece com o processador do nó 3, que permanece com o menor volume de carga.

Apesar da diferença no número de tarefas ao final da simulação, o gráfico Tempo de Espera converge para um valor de equilíbrio de $5,723s$, com settling time de $68ms$. Este é o tempo de espera necessário para que todas as tarefas, ao final do balanceamento de carga, tenham sido processadas.

A curva em magenta no gráfico Tempo de Espera da Figura 5.25 é a média global das filas $r(t)$. A média $r(t)$ acompanha a curva de $x_1(t)$, o que explica a oscilação encontrada no gráfico erro para $y_1(t)$.

A curva do gráfico Quantidade de Tarefas cresce e estabiliza-se em $7,629 \times 10^5$ tarefas, pois convertendo o pulso $\lambda_p = [750 \quad 50 \quad 5]^T$, que é uma taxa de entrada de tempo de espera nas filas em relação ao tempo, em uma taxa de entrada de tarefas em relação ao tempo, temos:

$$\dot{q}(t) = [750/t_{p_1} \quad 50/t_{p_2} \quad 5/t_{p_3}]^T \tag{5.69}$$

que é igual a

$$\dot{q}(t) = [750 \times 10^5 \quad 12,5 \times 10^5 \quad 0,41667 \times 10^5]^T \tag{5.70}$$

Logo, temos que

$$\sum_{i=1}^{3} \dot{q}_i(t) = 7,629 \times 10^7/s \tag{5.71}$$

Como o pulso λ_p teve duração de $0,01s$, então o total de tarefas inserido nas filas foi de $7,629 \times 10^5 \times 1 \times 10^{-2} = 7,629 \times 10^5$ tarefas.

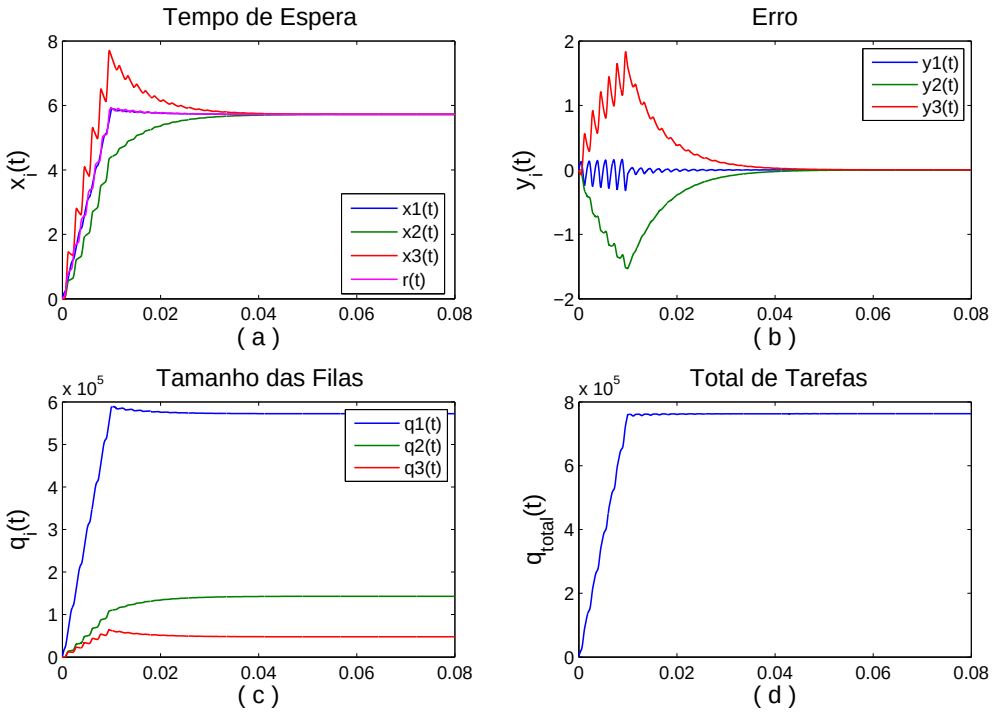

Figura 5.25: Simulação [07]: Resultado da simulação para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 1,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$.

A Figura 5.26 nos mostra um exemplo de atividade das funções não-lineares para um cluster heterogêneo $t_p = [10 \quad 40 \quad 120]^T$, com condições iniciais nulas e pulso vetorial de amplitude $[750 \quad 50 \quad 5]^T$ e $10ms$ de duração. A grande diferença de amplitude nos pulsos

decorre da necessidade de excitar as funções não-lineares de modo que elas atinjam a região de saturação superior.

O gráfico da Figura 5.26a representa os sinais de erro do balanceamento de carga. Nele, podemos notar que $y_1(t)$ oscila em torno de zero, $y_2(t)$ decresce de forma oscilatória e $y_3(t)$ cresce também de forma oscilatória, porém com amplitudes maiores do que $y_2(t)$).

Como ambos os gráficos encontram-se na mesma escala de tempo, é possível estabelecermos uma correspondência entre o gráfico superior e o gráfico inferior na figura.

A primeira informação que podemos extrair da Figura 5.26 é que $y_2(t)$ é negativo durante todo o tempo de simulação. Isto significa que a fila correspondente ao nó 2 está sempre abaixo da média global estimada. Conseqüentemente, o controlador para o nó 2 está sempre na região III da Figura 2.2 e, desta forma, a fila correspondente ao nó 2 está permanentemente recebendo tarefas.

No intervalo entre 0 e $5,9ms$ segundos, $y_3(t)$ está oscilando intensamente, porém, com picos abaixo de $y_{max} = 1,3$. Neste intervalo, tanto o controlador $\phi(y_1(t)$ como $\phi(y_3(t))$ estão transferindo tarefas de suas respectivas filas, simultaneamente à recepção de novas tarefas à taxa de 750 tarefas/segundo e 5 tarefas/segundo.

No intervalo entre $5,9ms$ e $11,7ms$, o sinal de erro $y_3(t)$ ultrapassa y_{max} e retrocede quatro vezes, levando o controlador $\phi(y_3(t))$ à região de saturação II da Figura 2.2. No gráfico inferior da Figura 5.26 é possível observar o sinal de controle $\phi(y_3(t))$ chegando ao valor unitário quatro vezes consecutivas.

Note que, apesar da taxa λ_1 ser muito maior que λ_3, o ganho do controlador $\phi(y_1(t))$ também é muito maior que o ganho do controlador $\phi(y_3(t))$.

O regime é atingido em $68ms$, quando a atividade do último controlador ainda fora da região III ($\phi(y_3(t-h))$) da Figura 2.2, cessa.

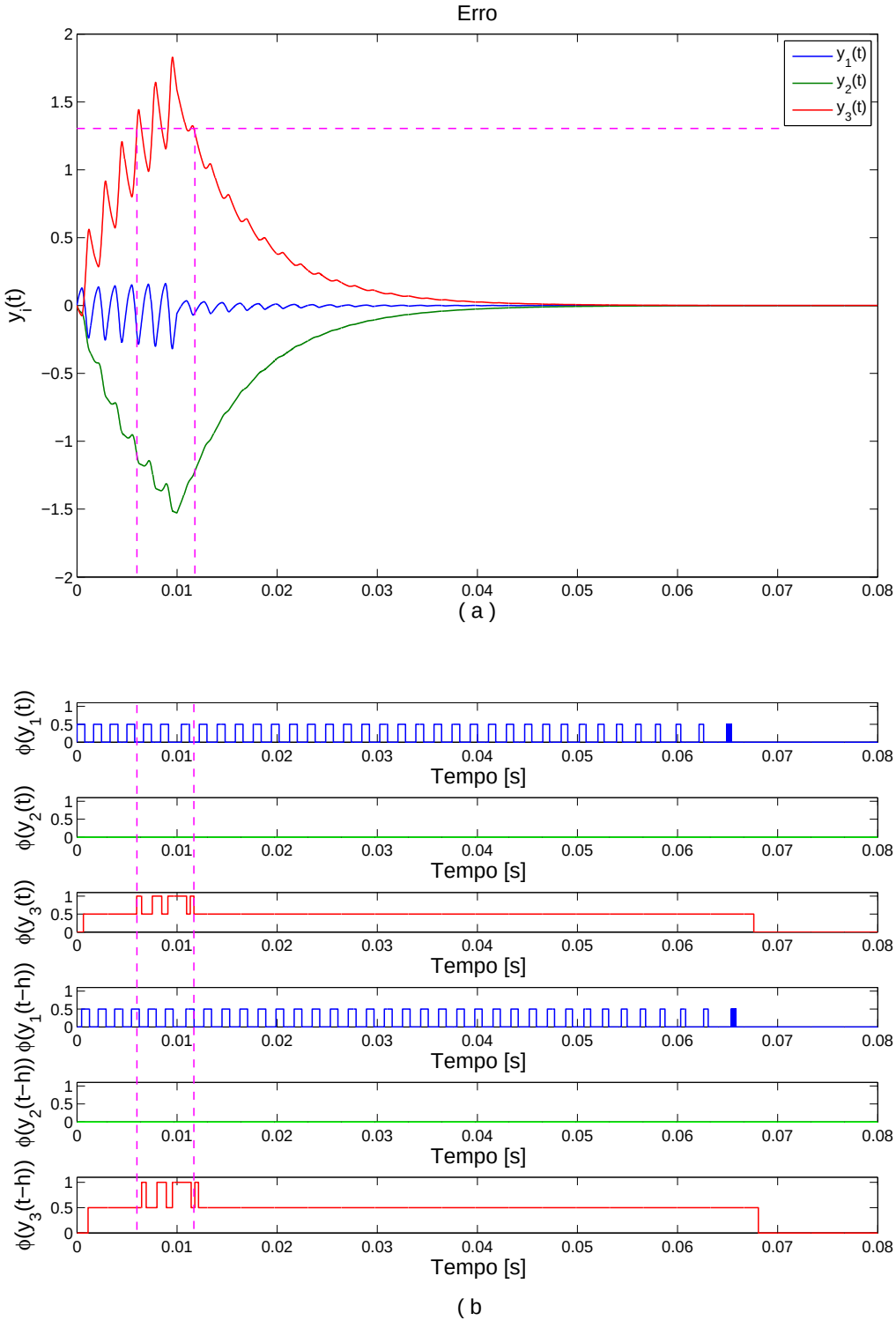

Figura 5.26: Simulação [07]: Exemplo de atividade das funções não-lineares $\phi(y_i(t))$ e $\phi(y_i(t-h))$ para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 1,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$.

Parâmetro	Valor
n	3
t_p	$[10\mu s \quad 40\mu s \quad 120\mu s]^T$
h	$400\mu s$
τ	$-$
$x(0)$	$[0 \quad 0 \quad 0]^T$
λ	$[750 \quad 50 \quad 5]^T$
t_λ	10 ms
y_{max}	0,3
K	$diag(3997,5 \quad 999,4 \quad 333,1)$

Tabela 5.14: Parâmetros para a simulação [08]

5.3.3.5 Simulação [08]: Efeito do limite da largura de banda II

Esta simulação possui como objetivo verificar como o limite de largura de banda afeta a velocidade de convergência do sistema. Para isto, vamos refazer a simulação [07] modificando o parâmetro y_{max}.

Os parâmetros de simulação utilizados são apresentados na Tabela 5.14.

As Figuras 5.27 e 5.28 apresentam os resultados da simulação.

Pode-se notar que, devido à largura de banda menor, um número menor de tarefas pode ser transferido, aumentando o settling time que, neste caso é de aproximadamente $90ms$.

As simulações mostraram que, na maioria das vezes, o ponto de operação do sistema excursiona entre as regiões I e III das funções não-lineares - Figura 2.2. Esta observação pode ser explicada da seguinte forma: Como as condições iniciais $x(0)$ são nulas, o erro $y(0)$ também é nulo - Ver equação (3.13). Com o passar do tempo o erro tende a aumentar, mas a quantidade de tarefas transferidas entre os nós tende a aumentar proporcionalmente também pelo valor de K, reduzindo assim o valor de $y(t)$ ao final. Em outras palavras, é necessário uma grande diferença de amplitudes entre os pulsos de entrada para que as funções sejam levadas à saturação superior.

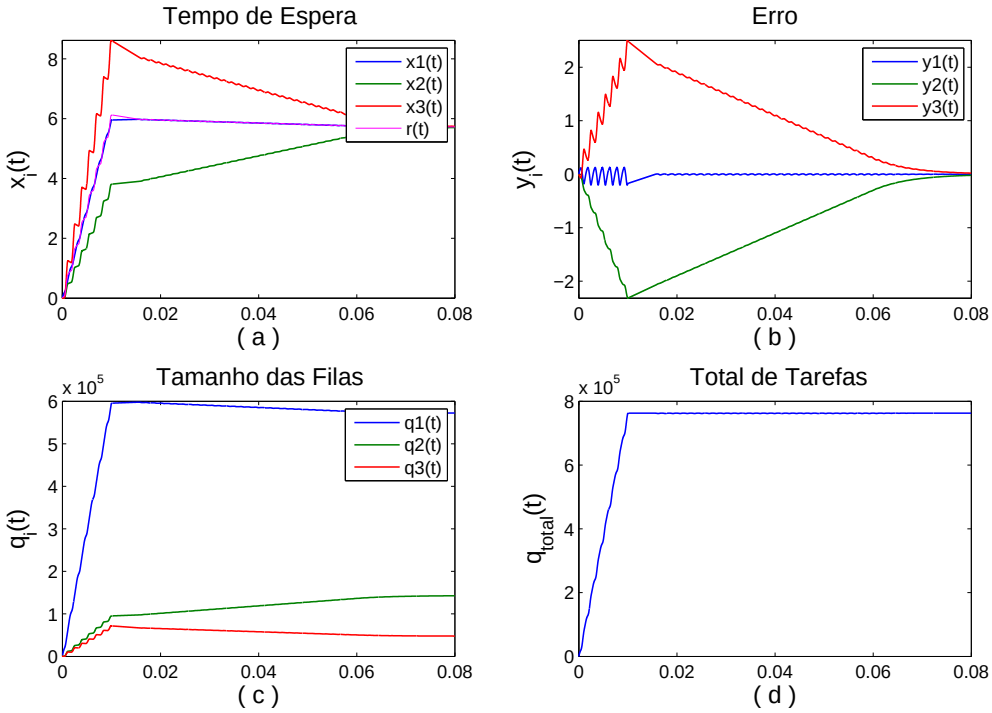

Figura 5.27: Simulação [08]: Resultado da simulação para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 0,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$.

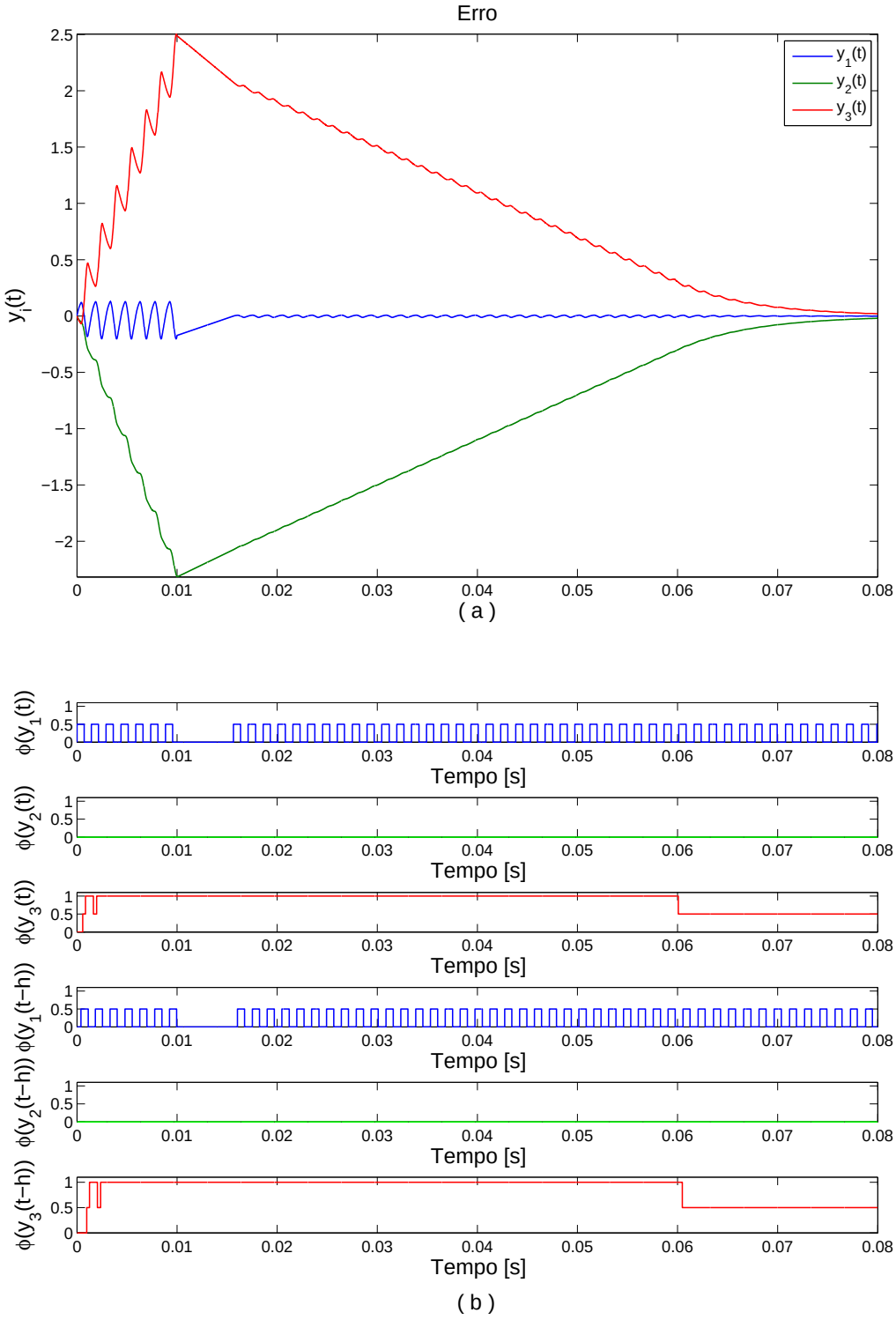

Figura 5.28: Simulação [08]: Exemplo de atividade das funções não-lineares $\phi(y_i(t))$ e $\phi(y_i(t-h))$ para um cluster heterogêneo com $t_{p_1} = 10\mu s$, $t_{p_2} = 40\mu s$ e $t_{p_3} = 120\mu s$, entrada $\lambda = [750 \quad 50 \quad 5]^T$, respectivamente, com duração de $t_\lambda = 10ms$, $y_{max} = 0,3$ e matriz de ganhos $K = diag(3997,5 \quad 999,4 \quad 333,1)$.

5.3.3.6 Simulação [09]: Outras leis de controle

O objetivo desta simulação é promover uma comparação entre as funções não-lineares apresentadas na subseção 5.1.3 a fim de verificarmos o comportamento de cada uma delas, principalmente no que tange ao emprego do terceiro quadrante.

Para as simulações, utilizaremos a matriz de ganhos:

$$K_{[09]} = \begin{bmatrix} 925,1 & -1582,2 & -221,1 \\ -1586,1 & 6080,1 & -684,2 \\ -223,1 & -688,7 & 6346,8 \end{bmatrix} \tag{5.72}$$

As Figuras 5.29-5.31 apresentam os sinais de erro $y_i(t)$ para o modelo (2.1) utilizando a matriz de ganhos (5.72) para as funções de primeiro e primeiro-terceiro quadrantes: $uhsat(y_i(t))$, $sat(y_i(t))$, $uhsign(y_i(t))$, $sign(y_i(t))$, $uhtanh(y_i(t))$ e $tanh(y_i(t))$.

Aparentemente, a resposta dada pelas funções não-lineares $uhsign(y(t))$ e $sign(y(t))$, mostrada nas Figuras 5.29c e 5.29d respectivamente, apresentam a melhor resposta em termos de convergência e overshoot. No entanto, a Figura 5.31 nos mostra que na verdade estas funções não são adequadas para o algoritmo de balanceamento, pois elas não satisfazem as condições apresentadas em (4.51). Através da Figura 5.31a é possível observar que o tempo de espera estimado ($x_i(t)$) tende para um ponto de equilíbrio inviável.

De forma geral, as funções que utilizam o primeiro-terceiro quadrante apresentam desempenho um pouco mais oscilatórios do que as funções de apenas primeiro quadrante, tanto para matrizes de ganho diagonais como não-diagonais.

Desta forma, as funções de primeiro quadrante foram as escolhidas para a implementação do algoritmo de balanceamento de carga.

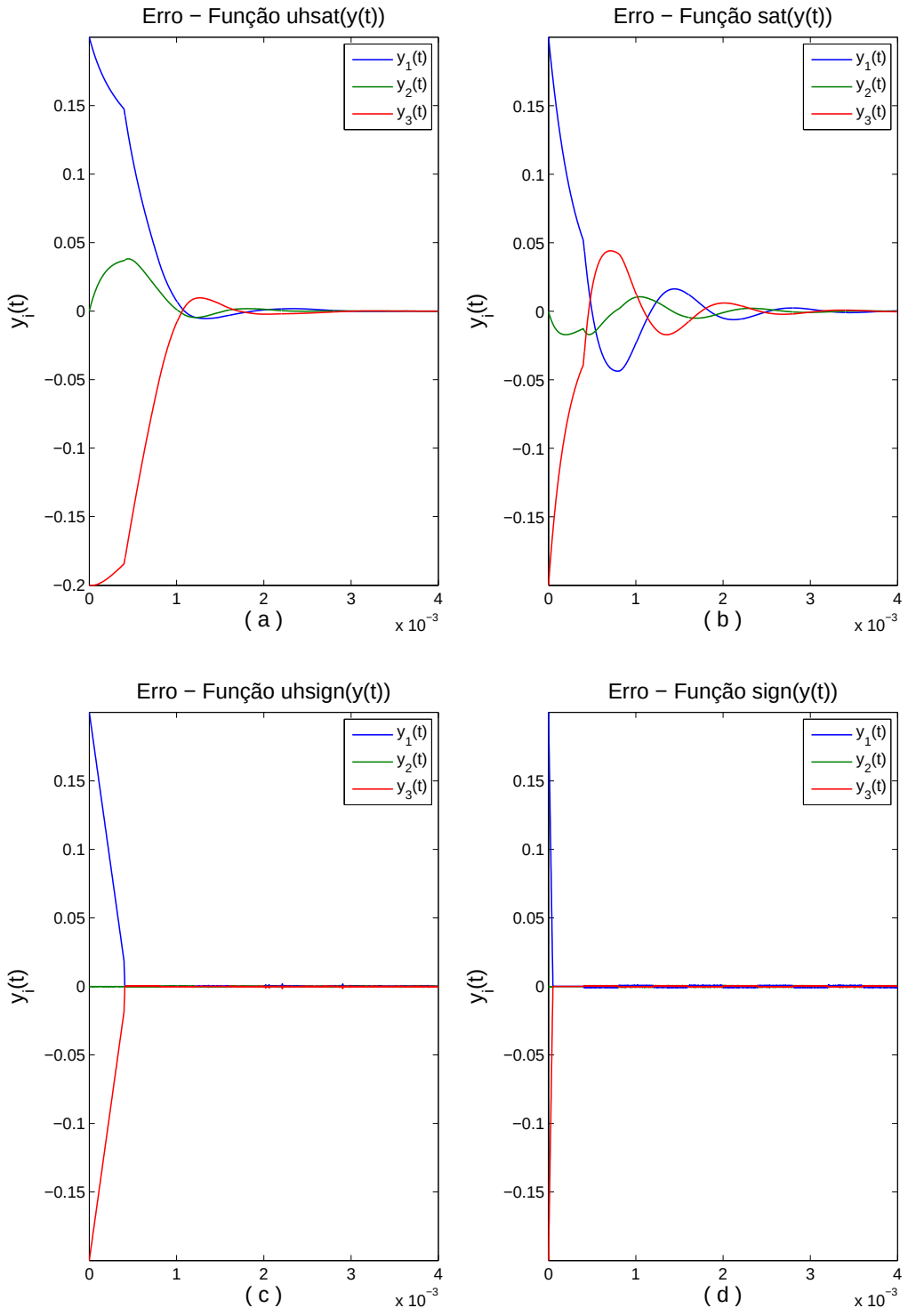

Figura 5.29: Simulação [09]: Comparação de desempenho para as funções não-lineares de primeiro e primeiro-terceiro quadrantes (uhsat, sat, uhsign, sign) para a matriz de ganhos $K_{[09]}$ (5.72).

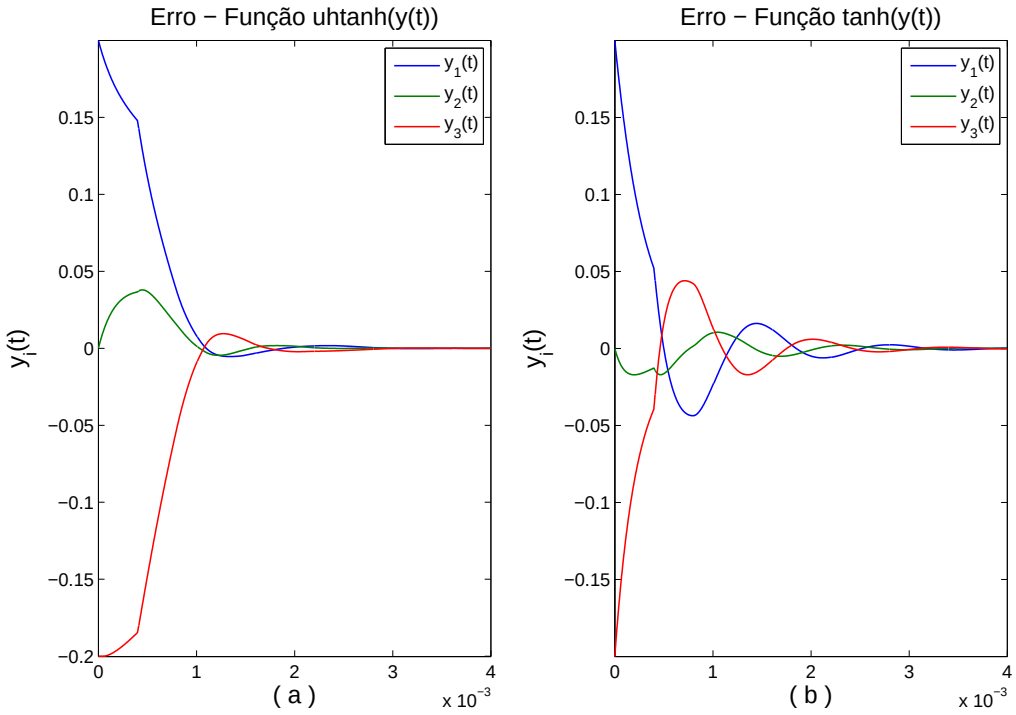

Figura 5.30: Simulação [09]: Comparação de desempenho para as funções não-lineares de primeiro e primeiro-terceiro quadrante (uhtanh e tanh) para a matriz de ganhos $K_{[09]}$ (5.72).

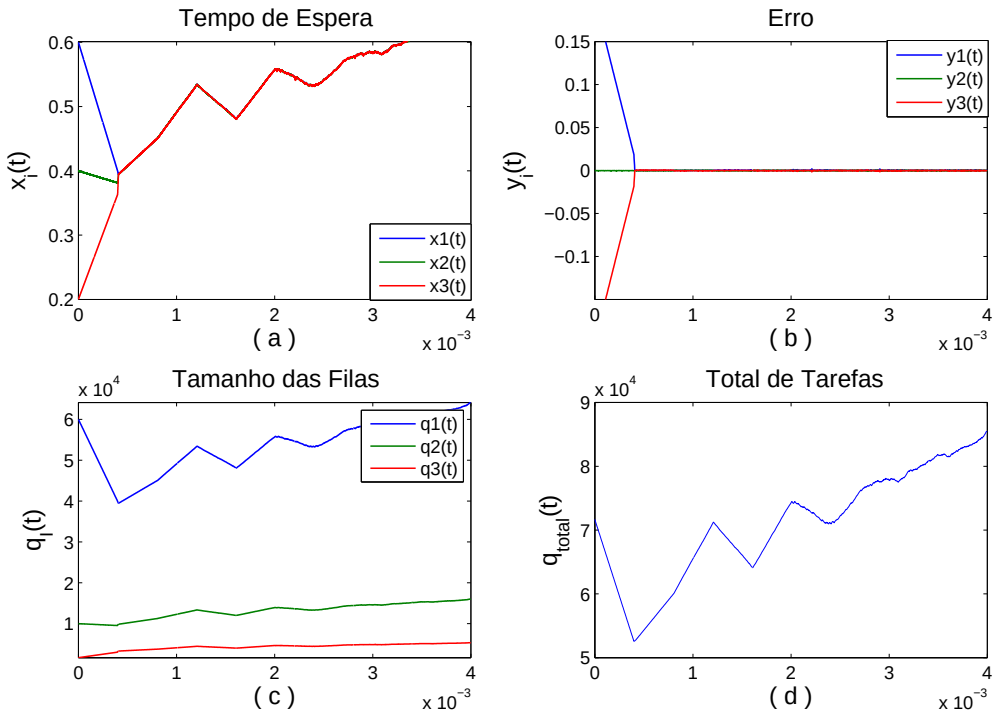

Figura 5.31: Simulação [09]: Resposta do modelo (2.1) para a matriz de ganhos $K_{[09]}$ (5.72) e função $sign(y(t))$.

5.3.3.7 Simulação [10]: O efeito de *bursting*

O objetivo desta simulação é destacar os efeitos dos atrasos quando estes tornam-se significativos em relação ao tempo da resposta transitória. A Figura 5.32 apresenta os resultados da simulação [10], com atraso h de $6,15ms$. A Figura 5.32e é uma ampliação da Figura 5.32b que, por sua vez, é relacionada à Figura 5.32f através de linhas verticais pontilhadas para auxiliar na visualização.

Na Figura 5.32e podemos verificar que o sistema converge para zero nos primeiros $6,15ms$. Em seguida ocorre um fenômeno que, no contexto desta tese, convencionou-se chamá-lo de *bursting*, fazendo com que o sistema aparentemente saia do seu ponto de equilíbrio para, logo em seguida, convergir novamente. Isto acontece sucessivamente e com amplitudes cada vez menores. CHIASSON *et al.* chamam este efeito de *ringing*.

Se analisarmos apenas a Figura 5.32e, podemos pensar que o balanceamento de carga terminou pois $y(t)$ se aproxima de zero dentro dos $6,15ms$ iniciais. No entanto, se acompanharmos a curva da Figura 5.32f, é fácil verificar que ainda existem tarefas trafegando na rede, o que significa que o balanceamento de carga não terminou na realidade.

Após a análise de diversas simulações com parâmetros diferentes, verificou-se que o efeito do bursting é devido aos atrasos presentes no sistema. Isto ocorre porque existe uma malha de realimentação com atraso no modelo (3.30) dado pelo termo $RTK\phi(y(t-h))$ e ela causa uma perturbação após h segundos. Se neste intervalo de tempo o sistema estiver perto do seu ponto de equilíbrio, as tarefas que chegam atrasadas irão perturbá-lo, fazendo com que os controladores entrem em ação novamente e busque acomodar a carga extra que chegou.

Na próxima simulação, será mostrado que os elementos fora da diagonal principal da matriz de ganhos K podem auxiliar na redução do efeito do bursting, dentro de determinados limites, corroborando o que foi dito na subseção 5.1.2 sobre a flexibilização do controlador.

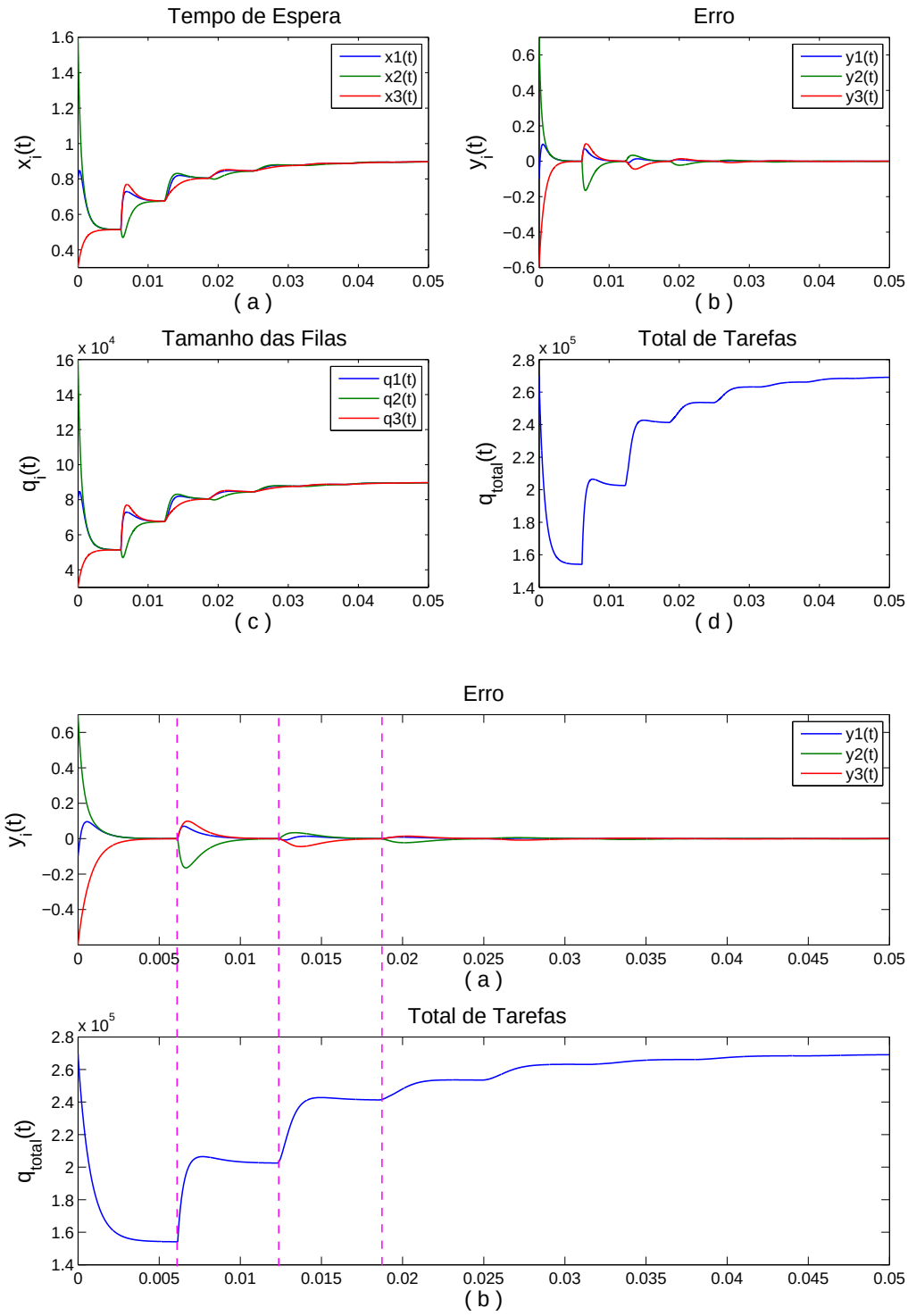

Figura 5.32: Simulação [10]: Resposta do modelo (2.1) a um atraso significativo em relação ao tempo da resposta transitória. O gráfico (f) destaca a resposta periódica onde o *burst* acontece.

5.3.3.8 Simulação [11]: A matriz de ganhos não-diagonal e o efeito do bursting

O objetivo desta seção é o de mostrar que os elementos fora da diagonal principal da matriz de ganhos K possui uma ação positiva em relação à redução do efeito do bursting.

Para esta simulação, vamos apresentar as respostas do modelo (2.1) para três matrizes de ganhos diferentes, a saber: $K^1_{[11]}$ que é uma matriz diagonal, $K^2_{[11]}$ que é uma matriz não-diagonal e $K^3_{[11]}$, que é a matriz não-diagonal $K^2_{[11]}$, porém com um aumento no módulo dos elementos $(1,3)$ e $(3,1)$. A idéia é a de mostrar que aumentando o módulo dos valores dos elementos fora da diagonal principal, a amplitude dos burstings pode ser reduzida, dentro de determinados limites.

Utilizando a matriz de ganhos diagonal $K^1_{[11]}$, a resposta do modelo (2.1) (sinal de erro $y(t)$) é apresentada na Figura 5.33a através das curvas sólidas em azul (node 1), verde (node 2) e vermelho (node 3).

$$K^1_{[11]} = \begin{bmatrix} 57089 & 0 & 0 \\ 0 & 43711 & 0 \\ 0 & 0 & 58277 \end{bmatrix} \tag{5.73}$$

A matriz $K^2_{[11]}$ é formada pela matriz $K^1_{[11]}$, porém substituindo os valores nulos por valores negativos, obtendo assim:

$$K^2_{[11]} = \begin{bmatrix} 57089 & -2593 & -23532 \\ -2593 & 43711 & -5890 \\ -23532 & -5890 & 58277 \end{bmatrix} \tag{5.74}$$

A resposta do modelo (2.1) (sinal de erro $y(t)$) é apresentada na Figura 5.33a através das curvas tracejadas em azul (node 1), verde (node 2) e vermelho (node 3).

Em seguida, para simplificar, vamos mudar o valor apenas do elementos $(1,3)$ da matriz de ganhos $K^2_{[11]}$ de -23532 para -35532, temos a matriz $K^3_{[11]}$.

$$K^3_{[11]} = \begin{bmatrix} 57089 & -2593 & -35532 \\ -2593 & 43711 & -5890 \\ -23532 & -5890 & 58277 \end{bmatrix} \tag{5.75}$$

A Figura 5.33 apresenta os resultados do aumento do módulo dos valores. A resposta do

Matriz de Ganho	$\mathbf{M_p}$	$\mathbf{M_t}$	t_s
$K^1_{[11]}$	0,04371	3,3E-5 s	37,82 ms
$K^2_{[11]}$	0,02480	2,7E-5 s	34,35 ms
$K^3_{[11]}$	0,01641	2,5E-5 s	29,16 ms

Tabela 5.15: Desempenho do nó 2 do modelo (2.1) para as três matrizes de ganho apresentadas: $K^1_{[11]}$, $K^2_{[11]}$ e $K^3_{[11]}$, mostrando o efeito dos valores fora da diagonal principal sobre o bursting e sobre o settling time.

modelo (2.1) (sinal de erro $y(t)$) é apresentada na Figura 5.33a através das curvas pontilhadas em azul (node 1), verde (node 2) e vermelho (node 3).

Podemos observar a redução da amplitude a medida que reforçamos a ação de controle através dos elementos fora da diagonal principal.

Nesta simulação, apesar das matrizes de ganho $K^1_{[11]}$, $K^2_{[11]}$ e $K^3_{[11]}$ terem sido arbitradas e construídas manualmente, isto não necessariamente gera matrizes estáveis.

Alterações nos elementos das matrizes Ω e Γ, aumentando ou diminuindo seus valores, aparentemente não possuem uma correlação direta com os elementos da matriz de ganho K, dificultando assim uma regra geral para a definição delas.

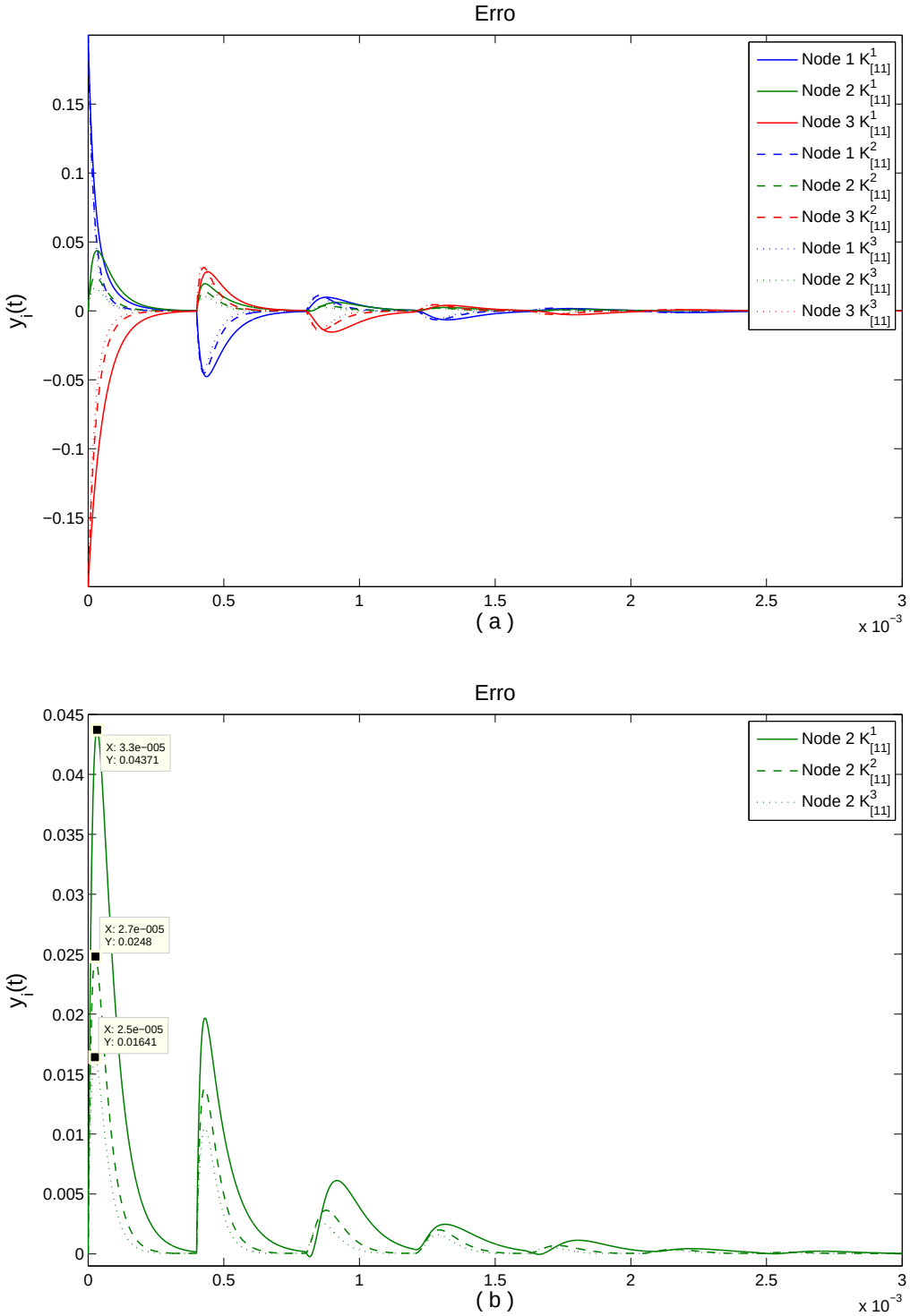

Figura 5.33: Simulação [11]: O gráfico (a) apresenta a resposta do modelo (2.1) para as matrizes de ganhos $K^1_{[11]}$, $K^2_{[11]}$ e $K^3_{[11]}$. O gráfico (b) destaca a resposta do nó 2, mostrando que os elementos fora da diagonal principal da matriz de ganhos contribui para a redução do efeito de bursting criado pelo atraso.

5.4 Metodologia Proposta para a Síntese

Nesta seção será apresentada a metodologia de síntese de controladores por Função de Custo Quadrático com solução via LMIs, onde os passos necessários para a sua aplicação serão enumerados.

1. Estimar experimentalmente o atraso de transferência médio entre os nós do cluster - parâmetro h (necessário apenas para a simulação);

2. Estimar experimentalmente o tempo médio para uma tarefa ser executada no i-ésimo nó - parâmetro t_{p_i};

3. Determinar a matriz R do **Modelo de Rede Neural** (3.30);

4. Determinar as matrizes W e W_h;

5. Arbitrar a Matriz de Ponderação de Estados $\Omega \succeq 0$ e a matriz $\Gamma \succ 0$;

6. Resolver o problema de feasibility da LMI Ξ (5.60) utilizando um pacote numérico como o Matlab, Sedumi ou um outro;

7. Determinar a matriz de ganhos K utilizando a equação

$$K = Z_1 P_{are}^{-1} \tag{5.76}$$

8. Substituir K no **Modelo (2.1)** (3.1);

9. Se necessário for, ajustar os valores das matrizes Ω e Γ e repetir os passos anteriores a partir do número 6.

A metodologia apresentada exige que a Matriz de Ponderação dos Estados Ω e a matriz Γ sejam arbitradas. Nas simulações, os valores utilizados iniciaram-se sempre com a matriz identidade para Ω e $diag(10^{-10} \quad 10^{-10} \quad 10^{-10})$ para Γ.

A medida que reduzimos os valores de Γ, isoladamente, o sistema vai tornando-se mais lento (ganhos menores), pois menor ênfase (peso) será dado ao controle. Por outro lado, a medida que os valores dos elementos de Γ vão aumentando, o sistema vai tornando-se mais

rápido e, conseqüentemente, apresentando oscilações cada vez maiores até que a LMI Ξ não seja mais factível (sistema instável).

Em relação à matriz Ω, foi utilizada uma matriz diagonal e, a medida que os valores eram reduzidos, o sistema também tornava-se mais lento, e vice-versa, repetindo o comportamento de Γ.

5.5 Resumo do Capítulo

Neste capítulo, apresentamos três diferentes metodologias para a síntese de controladores, a saber: minimização do índice ITAE por Algoritmos Genéticos, minimização da norma de $|Z_1|$ e minimização de Função Custo Quadrático via LMIs.

A Tabela 5.16 apresenta o custo computacional de cada uma das metodologias para encontrar uma solução (Δt_{sol}) e o settling time encontrado nas simulações [02], [03] e [04].

Podemos notar através desta tabela que o melhor desempenho obtido foi através da Definição 5.1: minimização do índice ITAE via Algoritmos Genéticos. No entanto, seu custo é demasiadamente alto.

Em seguida, um bom desempenho foi encontrado através da Definição 5.3: Minimização de Função Custo Quadrático via LMIs, cujo custo computacional é bem menor do que o demandado pelos Algoritmos Genéticos, tornando-se uma boa solução de compromisso entre desempenho e custo computacional.

A substituição da matriz de ganhos K na LMI (5.21), em todas as simulações, apresentou autovalores negativos, mostrando que a LMI é satisfeita.

Apesar da metodologia de síntese dada pela Definição 5.2 não ter correspondido com bons resultados, ela serviu com base para a continuarmos os estudos e chegar à metodologia dada pela Definição 5.3.

Através de outras simulações, foi possível constatar a eficiência da metodologia dada pela Definição 5.3 para o caso de clusters heterogêneos, bem como constatar os efeitos de elevadas taxas de geração de tarefas e suas implicações com o limite da largura de banda da rede de comunicação.

O custo computacional aumenta consideravelmente conforme aumenta a ordem n do sistema, porém, a uma taxa menor que o custo do algoritmo genético. O uso de métodos paralelos para o algoritmo de pontos interiores pode contribuir de forma positiva para melhorar

	Definição 5.1	Definição 5.2	Definição 5.3
Δt_{sol}	57,6 s	117,2 ms	219,4 ms
Settling time	3,29 ms	800 ms	3,88 ms

Tabela 5.16: Tabela comparativa para as três diferentes metodologias de síntese de controladores. Δt_{sol} é o tempo necessário utilizado para encontrar uma solução, e os respectivos settling time encontrados nas simulações [02],[03,[04].

a escalabilidade da metodologia proposta.

Por fim, evidenciamos o problema do bursting e apresentamos dados que apontam para possíveis melhorias na redução deste efeito pelo emprego de matrizes de ganho não-diagonais.

Capítulo 6

Simulações e Experimentos em Clusters Homogêneos e Heterogêneos

Neste capítulo iremos apresentar os resultados numéricos obtidos a partir da implementação da lei de controle 5.3 com ganhos determinados pelas três metodologias apresentadas no capítulo 5, e a metodologia empregada em CHIASSON *et al.* [33].

Cada um dos experimentos será precedido de uma simulação, a fim de verificar a adequabilidade do modelo (2.1) ao problema de balanceamento de carga.

Na seção 6.1 descreveremos a arquitetura do software desenvolvido para realizar os experimentos de balanceamento de carga, bem como algumas características de *threads* e das bibliotecas de *Interface de Passagem de Mensagens* - as chamadas bibliotecas MPI - e algumas considerações sobre o uso simultâneo de threads e MPI.

Na seção 6.2 descreveremos o setup experimental implementado para realizar os experimentos, descrevendo como foram realizadas as estimativas de atraso de comunicação entre os nós do cluster e as estimativas de tempo de processamento médio de uma tarefa em um determinado processador. Tais informações foram utilizadas para as simulações que precedem os experimentos.

Nas seções 6.3 e 6.4, serão apresentadas os experimentos realizados em clusters homogêneos e heterogêneos precedidos de suas respectivas simulações.

Na seção 6.5 será realizada uma análise geral dos resultados apresentados através das simulações e dos experimentos.

E, finalmente, a seção 6.6 apresentará o resumo do capítulo.

6.1 Arquitetura de Implementação

A arquitetura de implementação do programa para validar os conceitos da tese é mostrada na Figura 6.1.

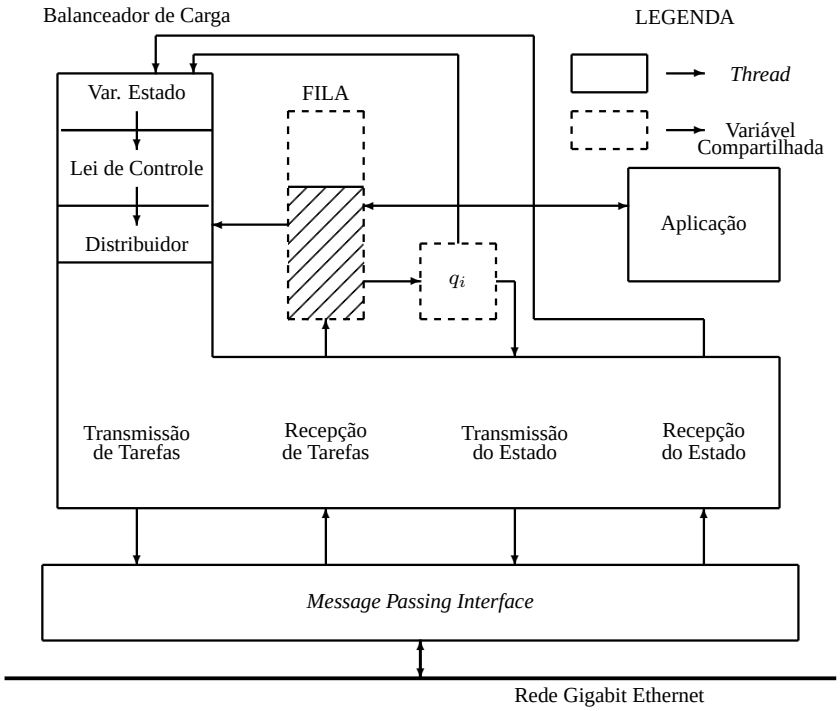

Figura 6.1: Arquitetura de Implementação

Esta arquitetura de implementação foi realizada através da codificação de um programa na linguagem C, baseado em um processo principal (*main*) e um outro processo independente (thread), e a biblioteca MPI (Message Passing Interface) para comunicação ente os nós. Cada nó do cluster executa o mesmo algoritmo.

O programa, batizado de LoadBalancer, consiste na criação de duas variáveis compartilhas: Uma estrutura de dados estática (vetor de tarefas) chamada fila, e uma variável escalar, chamada de q, que armazena o número de tarefas existentes na fila.

Uma fila de tarefas é normalmente criada através de uma estrutura dinâmica chamada de lista encadeada [45]. Dentre as suas vantagens é possível citar a facilidade de inserção ou remoção de tarefas que estejam no meio da fila, reordenação conforme a prioridade das tarefas entre si, e a possibilidade de alocação dinâmica de memória, fazendo com que o limite de tamanho da fila seja dado pela quantidade de memória física do sistema.

Em contrapartida, a sua implementação é mais complexa e exige um custo computaci-

onal maior para gerenciá-la quando comparada com uma implementação estática (vetor de tarefas), pois necessita de chamadas ao sistema para alocar e liberar memória a todo instante.

A implementação de filas através de uma estrutura de dados estática como um array permite uma redução na complexidade do programa e no esforço computacional para gerenciá-la. No entanto, esta implementação possui uma capacidade limitada de armazenamento definida no chamado tempo de compilação.

Para um programa com o objetivo de testar os conceitos apresentados nesta tese, a opção para o LoadBalancer foi adotar a estrutura em array, a fim de evitar o custo computacional extra e minimizar a implementação por não haver a necessidade de termos uma fila ilimitada.

O capítulo 4 de LANGSAM *et al.* [86] apresenta uma boa discussão sobre Filas, Listas Encadeadas e suas implementações e gerenciamento na linguagem C/C++.

6.1.1 Processos Concorrentes e threads

Os processos são programas carregados na memória que representam algoritmos e podem ser classificados como leves e pesados.

Os processos pesados são os tradicionais, possuindo um conjunto de operações iniciais básicas que começam a sua execução.

Um processo leve, ou thread, é uma parte de um processo pesado que é executado concorrentemente.

Em um sistema com mais de um processador, o sistema operacional passa a dispor de mais CPUs para alocar os processos. O resultado é a execução simultânea real de vários processos.

Todos os processos que existam ao mesmo tempo são então dito concorrentes. Eles podem funcionar completamente independentes uns dos outros, ou ocasionalmente necessitar de sincronização e cooperação [87].

O uso de threads permite a estruturação do código em um conjunto de atividades independentes, de forma que elas possam ser executadas em concorrência, permitindo o paralelismo.

As threads permitem ainda uma redução do gasto de memória e processamento, uma vez que o processo pesado (principal) e os processos leves (threads) compartilham, além do segmento de código, o segmento de dados e espaço de endereçamento. As threads possuem apenas sua própria pilha de execução e *program counter*.

A comunicação entre as threads pode dar-se através de variáveis globais compartilhadas e a utilização destas variáveis pode ser controlada através de estruturas como monitores, semáforos, primitivas de exclusão mútua, variáveis condicionais e construções similares.

Estas características facilitam a implementação de um algoritmo de balanceamento de carga do tipo *sender-initiated*, onde um dos nós inicia o balanceamento transferindo parte da sua carga para outros nós.

6.1.2 Message Passing Interface - MPI

A MPI é um conjunto de padrões para a programação por troca de mensagens e foi desenvolvido por um fórum (*Message Passing Interface Forum*) que envolveu representantes de diversas áreas, tais como: acadêmica, empresarial e governamental.

Apesar da implementação do programa poder utilizar os protocolos UDP e TCP para a comunicação e transferência de tarefas [45, 88], respectivamente, o padrão MPI possui características que possibilitou aos desenvolvedores de programas paralelos a criarem bibliotecas eficientes e portáteis entre diferentes plataformas. Desta forma, o padrão MPI, ou simplesmente MPI, não é uma nova linguagem de programação, mas sim um padrão para bibliotecas de definições e funções que podem ser utilizadas em conjunto com linguagens como C e Fortran para alcançar o paralelismo em programas computacionais.

Existem diversas distribuições de bibliotecas MPI, como: MPICH, LAM/MPI, Open MPI, OpenMP, e outras.

Dentre as facilidades oferecidas pelas implementações, temos diretivas de comunicação de envio (MPI_Send) e recepção (MPI_Receive) de mensagens, e de broadcast (MPI_Bcast), onde um nó distribui uma determinada mensagem para todos os outros.

6.1.3 MPI e threads

Uma das dificuldades encontradas na abordagem de programas *multithread* utilizando uma biblioteca MPI reside na garantia de que diversas threads poderão realizar chamadas às funções MPI simultaneamente sem que haja problemas de sincronismo e comunicação [89]. As chamadas *thread-safe*.

O padrão para a implementação de distribuições MPI não requer um ambiente thread-safe, ficando à cargo das distribuições existentes implementar as premissas necessárias.

Para utilizarmos um programa com threads e chamadas à funções MPI, é necessário substituir a função de inicialização do MPI: MPI_Init, pela função MPI_Init_threads que tem como argumento de entrada o parâmetro *required*, que define qual será o suporte a threads necessários à biblioteca MPI, e o argumento de saída é o endereço da variável *provided* que será utilizada pela biblioteca MPI para dizer à aplicação qual suporte à thread foi concedido.

O suporte a threads disponível recai um uma das quatro categorias mostradas a seguir:

- MPI_THREAD_SINGLE que significa que nenhum suporte a threads é disponibilizado e a utilização desta categoria é equivalente à utilização do comando MPI_Init;

- MPI_THREAD_FUNNELED no qual apenas a thread principal pode fazer chamadas à rotinas MPI. Nesta categoria as outras threads podem executar outras tarefas enquanto a thread principal faz chamadas a funções MPI;

- MPI_THREAD_SERIALIZED na qual múltiplas threads podem fazer chamadas a rotinas MPI, mas apenas uma thread pode executar uma rotina MPI a cada momento;

- MPI_THREAD_MULTIPLE na qual threads podem fazer chamadas a rotinas MPI sem qualquer restrição.

Em um programa MPI, todas as threads existentes compartilham o mesmo comunicador MPI. Assim, uma mensagem destinada a uma thread estará acessível também às demais threads, e é impossível endereçar uma thread específica para receber uma determinada mensagem. Desta forma, a thread que realizar uma chamada a uma função de recepção de mensagem será a thread destinatária da referida mensagem. Também em um programa MPI, quando uma thread é bloqueada pela chamada a alguma função MPI, somente a thread que realizou a chamada permanecerá bloqueada até a finalização da requisição, as outras threads poderão continuar com o seu processamento.

O processo principal do programa LoadBalancer cria uma thread de leitura do tamanho da fila a cada 1 ms.

A tarefa consiste em uma variável do tipo inteiro longo (*longint*) que corresponde a 4 bytes em C. Esta variável, por sua vez, corresponde ao expoente de uma matriz 10×10 do tipo real (*float*) que corresponde também a 4 bytes na linguagem C.

6.2 Setup Experimental e Parâmetros do Modelo de Rede Neural

Nesta seção, o levantamento dos parâmetros [46] do Modelo de Rede Neural é apresentado.

Para os experimentos, utilizou-se o cluster homogêneo do Instituto de Pesquisas da Marinha (IPqM) - Figura 1.3 - formado por 11 nós Pentium 4HT de 2.6GHz cada processador e com 11 Gb de RAM e rodando LINUX distribuição Kurumin 4.1 em todos os nós.

Experimentos foram realizados a fim de determinar os valores de y_{max}, que é uma parâmetro diretamente relacionado à largura de banda da rede [45]. A Figura (6.1) apresenta os resultados experimentais do tempo de transferência de tarefas (atraso de transferência) entre nós em função do número de bytes.

A Tabela 6.1 apresenta os valores obtidos experimentalmente para o atraso de transferência em função do número de bytes. Um algoritmo foi criado e implementado em C e MPI para determinar estes valores.

A Figura 6.2 apresenta o gráfico formado pelos valores da Tabela 6.1.

Para calcularmos a largura de banda da rede, vamos tomar o tempo utilizado para transmitir 32768 bytes, que foi de 0,90 ms. Como o *round-trip time* é o tempo necessário para um pacote ir e voltar, então temos:

$$\text{Largura (Mps)} = 32768 \times 8 \quad \text{bits}/0,45 \times 10^{-3} \approx 582 \quad \text{Mbps} \tag{6.1}$$

Esta informação será necessária para calcularmos o parâmetro y_{max} mais adiante que, por sua vez, é utilizado na função $uhsat(y_i(t))$ nas simulações.

Na implementação, a função $uhsat(y_i(t))$ foi redefinida como:

$$uhlin(y_i(t)) = \begin{cases} y_i(t) & \text{se} \quad y_i(t) > 0 \\ 0 & \text{se} \quad y_i(t) \leq 0 \end{cases} \tag{6.2}$$

O parâmetro y_{max} representa a taxa de redução máxima que poderá ser aplicada à fila. Com uma tarefa de 4 bytes (32 bits/tarefa) a ser transmitida, o parâmetro y_{max} é definido por:

N^o Bytes	h	N^o Bytes	h
100	1,30 μs	600	7,82 μs
200	2,41 μs	700	9,22 μs
300	3,94 μs	800	10,32 μs
400	5,22 μs	900	11,86 μs
500	6,63 μs	1000	12,89 μs

Tabela 6.1: Atraso de transferência h em função do número de bytes transferidos

$$y_{max} = \frac{\text{largura de banda máxima [bps]}}{\text{tamanho da tarefa [bits/tarefa]}} \times t_{p_i} \tag{6.3}$$

Substituindo os valores em (6.3), temos:

$$y_{max} \approx \frac{596 \times 10^6}{32} \times 10 \times 10^{-6} \approx 167,6 \tag{6.4}$$

Ou seja, o limite máximo da função não-linear $\phi(y_i(t)) = uhsat(y_i(t)$ é de $167,6$.

As subseções (6.2.1) e (6.2.2) apresentam maiores detalhes das estimativas experimentais realizadas.

6.2.1 Estimativa do Atraso na Rede Local do Cluster

Para realizar a caracterização do atraso na rede interna ao cluster do IPqM, foi implementado um programa na linguagem C com a biblioteca MPI *mpich-1.2.6* o qual envia um número pré-determinado de tarefas de um nó a outro. Utilizando a função *MPI_Wtime* no início do loop que envia as tarefas, e realizando uma nova chamada após o término deste, é possível avaliar o atraso de transferência médio h.

O teste foi realizado através da média das medidas de atraso de 1.000 pacotes de 100 a 1.000 bytes cada, entre as combinações de pares possíveis de nós.

A Tabela 6.1 apresenta o atraso de transferência médio h em função da quantidade de bytes a serem transferidos, e a Figura 6.2 apresenta o seu respectivo gráfico. Podemos notar nesta figura que o atraso de transferência possui uma tendência linear.

O mesmo procedimento foi utilizado para encontrarmos o atraso de comunicação - parâmetro τ. A mensagem típica contém o tamanho das filas de todos os nós do sistema, o que significa que a mensagem carrega n bytes (tipo inteiro) de informação. A medida do parâmetro τ encontrada para os experimentos aqui presentes foi $\tau \approx 0,17\mu s$ (mensagem

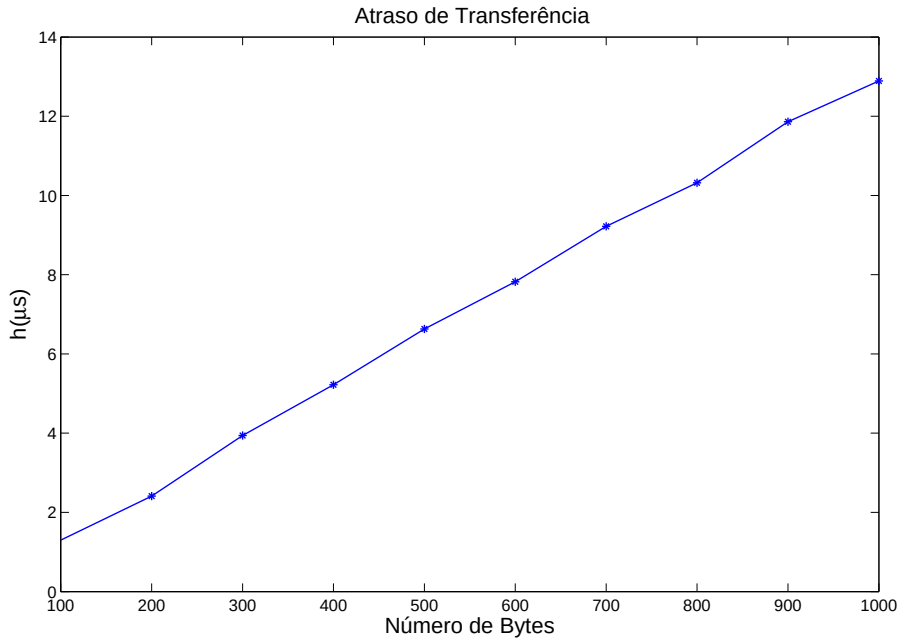

Figura 6.2: Atraso de transferência h em função do número de bytes a serem transmitidos.

contendo a informação de $q_1(t)$, $q_2(t)$ e $q_3(t)$).

6.2.2 Estimativa do Tempo Médio para processamento de uma tarefa em um dado processador

Um dos parâmetros necessários no **MB** que necessita de determinação experimental é o tempo médio para processamento de uma tarefa, ou t_{p_i}.

Este parâmetro representa a capacidade de processamento de um determinado nó, ou seja, quanto maior o parâmetro t_{p_i}, menor a velocidade de processamento do i-ésimo nó. E vice-versa.

Para realizar o levantamento experimental, um algoritmo foi monitorado a fim de verificar o tempo necessário para que ele realizasse as operações necessárias sobre a tarefa, que é elevar a matriz 10×10 à um determinado expoente (tarefa).

O tempo médio de execução de uma tarefa t_{p_i} em função da variável é de aproximadamente $9\mu s$

O algoritmo foi executada 1.000 vezes e a média calculada para obtermos t_{p_i}. Como os três processadores são idênticos, o tempo médio de execução de cada um foi praticamente o

Norma	Definição
1	Maior soma por coluna da matriz Π
2	Maior valor singular da matriz Π
Infinita	Maior soma por linha da matriz Π

Tabela 6.2: Definição de normas de uma matriz quadrada Π qualquer.

mesmo.

6.2.3 Implementação da Lei de Controle Discreta

Sendo um cluster um sistema naturalmente discreto e, em relação ao que já foi exposto na subseção 2.2.3, é necessário utilizarmos a relação de transformação da matriz de ganhos K_i para a matriz de ganhos K_z a fim de implementar a lei de controle discreta 2.25.

Uma grande diferença entre o trabalho de CHIASSON *et al.* e o desta tese deve ser destacado: Enquanto CHIASSON *et al.* realizaram um trabalho focado na modelagem de um sistema discreto por um modelo contínuo, interessados em estudar os efeitos do atraso e com menor foco em controle, esta tese foca na síntese de controladores sobre um modelo contínuo para ser implementado em um modelo discreto, seguindo o caminho inverso dos autores.

Isto significa que CHIASSON *et al.* executavam um experimento para um determinado valor de ganho K_z, mediam os tempos Δt_i e, a partir da média destes tempos, encontravam um ganho K_i pela equação 2.25 e o utilizam para realizar as simulações.

O caminho inverso demonstrou-se bem mais desafiador pois, uma vez que tenhamos determinado os ganhos K_{ij}, a discretização auxiliada pela relação 2.25 depende do tempo Δt_i, o qual, por sua vez, depende do próprio ganho, recaindo assim em uma equação aberta.

Para solucionar este problema, voltou-se a atenção sobre o significado do ganho K_z variar no intervalo de $(0, 1)$. Um ganho $K_z = 0,7$ significa na prática que 70% do excesso de tarefas será removida da fila e transferida para os outros nós.

Desta forma, a fim de manter a coerência de K_z, a divisão de cada elemento da matriz de ganhos não-diagonal K por diversas normas de matrizes foram experimentadas, a fim de verificar a que dava melhores resultados. A Tabela 6.2 apresenta as normas testadas.

Outra tentativa foi a de dividir cada elemento do i-ésima linha pela soma dos módulos dos elementos da própria linha. Ao final de vários testes, a norma 2 mostrou-se a mais adequada.

Desta maneira, a matriz de ganhos $K^1_{[04]}$ (5.61) toma a seguinte forma:

$$\overline{K}^1_{[04]} = \frac{K^1_{[04]}}{||K^1_{[04]}||_2} = \begin{bmatrix} 0,8813 & -0,1187 & -0,1187 \\ -0,1187 & 0,8813 & -0,1187 \\ -0,1187 & -0,1187 & 0,8813 \end{bmatrix} \tag{6.5}$$

onde $\overline{K}^1_{[04]}$ é a matriz $K^1_{[04]}$ após ter sido normalizada.

6.3 Experimentos em um Cluster Homogêneo

Para demonstrar a eficácia da metodologia sistemática para a síntese de controladores para o problema de balanceamento de carga no caso homogêneo, foi utilizado o cluster do *Instituto de Pesquisas da Marinha* (IPqM) para a realização de diversos experimentos. Os experimentos foram numerados em ordem seqüencial e o objetivo de cada um é apresentado no início da cada subseção.

6.3.1 Experimento [01]: Desempenho com a matriz de ganhos diagonal

O objetivo deste experimento é o de verificar se o modelo (2.1) representa de forma adequada, dentro de determinados limites, o problema de balanceamento de carga entre processadores.

Para alcançarmos este objetivo, vamos apresentar algumas simulações, bem como os respectivos resultados experimentais, de um processo de balanceamento de carga entre três nós em um cluster homogêneo para valores de K_z iguais a 0,8, 0,9 e 1,0.

Em relação à simulação, o procedimento adotado pode ser enumerado da seguinte forma:

1. Escolher um valor para K_z;

2. Inserir este valor no algoritmo de balanceamento de carga e executar no cluster;

3. Gravar em arquivo o atraso de transferência $h_i(t)$ e o intervalo de balanceamento de carga $\Delta t_i(t)$ utilizado por cada nó, e a cada execução do algoritmo até que o sistema tenha atingido um critério de parada;

4. Calcular as médias de h_i e Δt_i do arquivo;

5. Calcular os ganhos K_i conforme a relação (2.25);

6. Inserir os valores médios de h_i e Δt_i na simulação e executar.

As Figuras 6.3 e 6.5 mostram os resultados experimentais para alguns valores escolhidos de K_z. É possível notar que à medida que aumentamos o valor do ganho K_z, o sistema vai tornando-se mais rápido, porém mais oscilatório. Para $K_z = 1,0$ - Figura 6.5, por exemplo, o sistema torna-se bastante oscilatório.

Diferentemente do capítulo 5 onde os gráficos destacados eram o do erro $y(t)$, neste capítulo será dado maior ênfase ao gráficos na variável x (tempo de espera estimado) por este apresentar a informação do valor do ponto de equilíbrio a variável $x(t)$. Este ponto de equilíbrio é o tempo de espera estimado que uma tarefa que esteja chegando à uma das filas, deverá esperar para ter o seu processamento iniciado.

Para a simulação apresentada no gráfico 6.3c, temos que Δt médio é 0,48 ms, obtido através do experimento [01] para $K_z = 0,8$. Com isto, temos:

$$K_i = 0,8/0,48ms = 1667,7, \quad i = 1,2,\ldots,n \tag{6.6}$$

Em seguida, utilizou-se a matriz de ganhos K como sendo:

$$K = \begin{bmatrix} 1667,7 & 0 & 0 \\ 0 & 1667,7 & 0 \\ 0 & 0 & 1667,7 \end{bmatrix} \tag{6.7}$$

O mesmo procedimento foi adotado na simulações para outros valore de K_z.

A Tabela 6.3 fornece a informação à respeito do Maximum Overshoot, Maximum Overshoot Instant e Settling Time em cada um dos resultados experimentais.

	Simulação			**Experimental**		
K_z	M_p	M_t	t_s	M_p	M_t	t_s
0,8	0,4181	1,21 ms	8,1 ms	0,4800	1,0 ms	16,0 ms
0,9	0,4144	1,20 ms	6,4 ms	0,4900	1,0 ms	13,0 ms
1,0	0,4109	1,18 ms	6,3 ms	0,5000	1,0 ms	12,0 ms

Tabela 6.3: Parâmetros de desempenho para o experimento [01].

As Figuras 6.3 e 6.5 apresentam os resultados experimentais e simulados para valores

de 0,8 a 1,0 de K_z. Na Figura, é possível verificar que os sistema vai tornando-se mais oscilatório conforme K_z vai aumentando.

Na Figura 6.3, o gráfico 6.3c apresenta uma diferença de amplitude do experimento para a simulação. Esta diferença pode ser explicada pelo fato do gráfico 6.3c apresentar um vale em sua amplitude (0,5ms) (lembrando que h médio utilizado foi de 0,5ms) antes do pico (1,2ms). Este vale é devido ao efeito do atraso e é, na verdade, o primeiro ciclo de bursting. O gráfico 6.3a, não apresenta este vale pois como o algoritmo foi implementado com comunicação síncrona, os efeitos do bursting não aparece. A medida que o ganho vai aumentando, os efeitos vão aumentando também, tornando a amplitude das curvas mais distantes dos resultados experimentais, porém, sem invalidar a tendência delas - Figura 6.4.

Outro fato responsável pelas diferenças entre simulações e experimentos é o de que as tarefas são consideradas como algo contínuo no modelo (2.1) e não quantizado como no problema real, onde cada tarefa possui um tamanho fixo dado em bytes.

De uma forma geral, os resultados experimentais estão de acordo com os resultados teóricos, corroborando assim os resultados apresentados em [45].

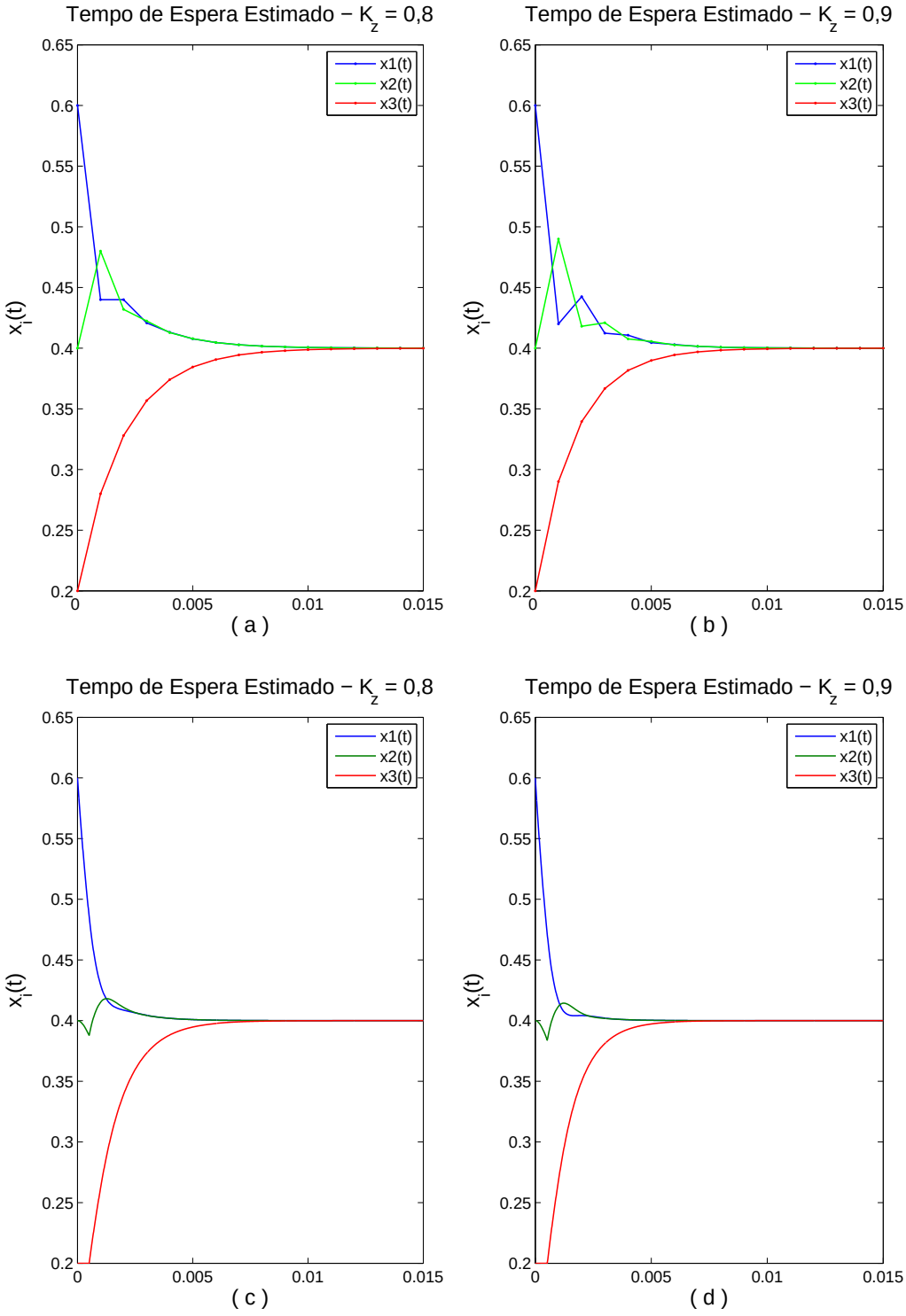

Figura 6.3: Experiência [01]: O gráfico (a) apresenta o tempo de espera estimado para $K_z = 0,9$ e o gráfico (b) apresenta a respectiva simulação com h médio de $532\mu s$ e Δ_t médio de 0,35ms.

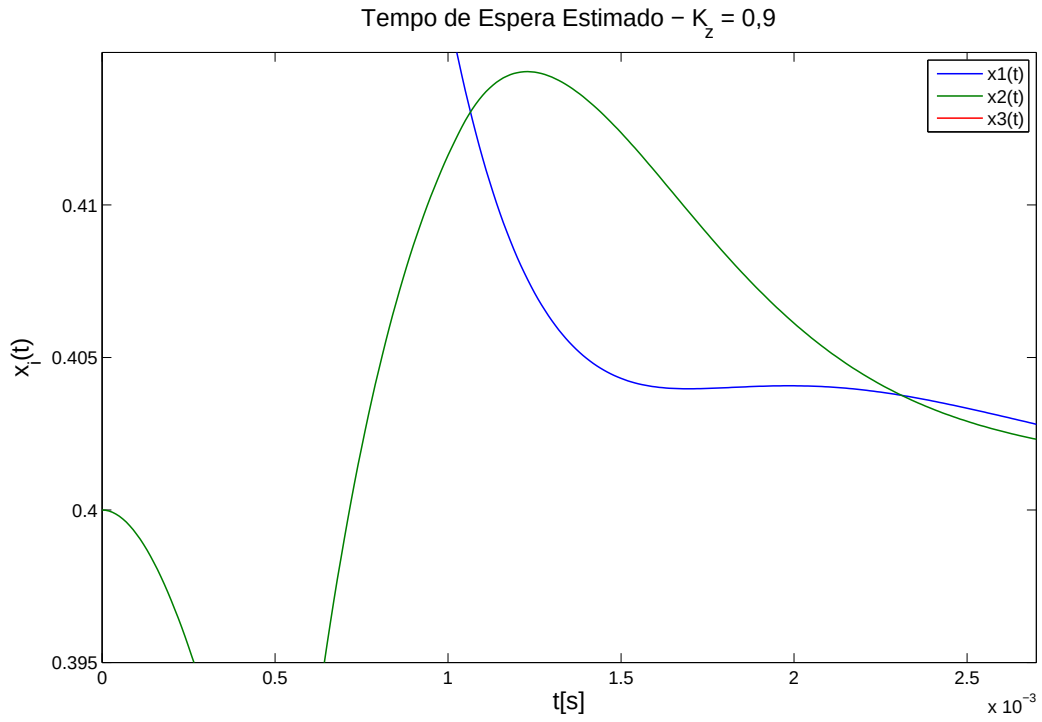

Figura 6.4: Experiência [01]: Ampliação de parte da Figura 6.3 detalhando a troca simulada de tarefas entre os nós 1 e 2 para $K_z = 0,9$.

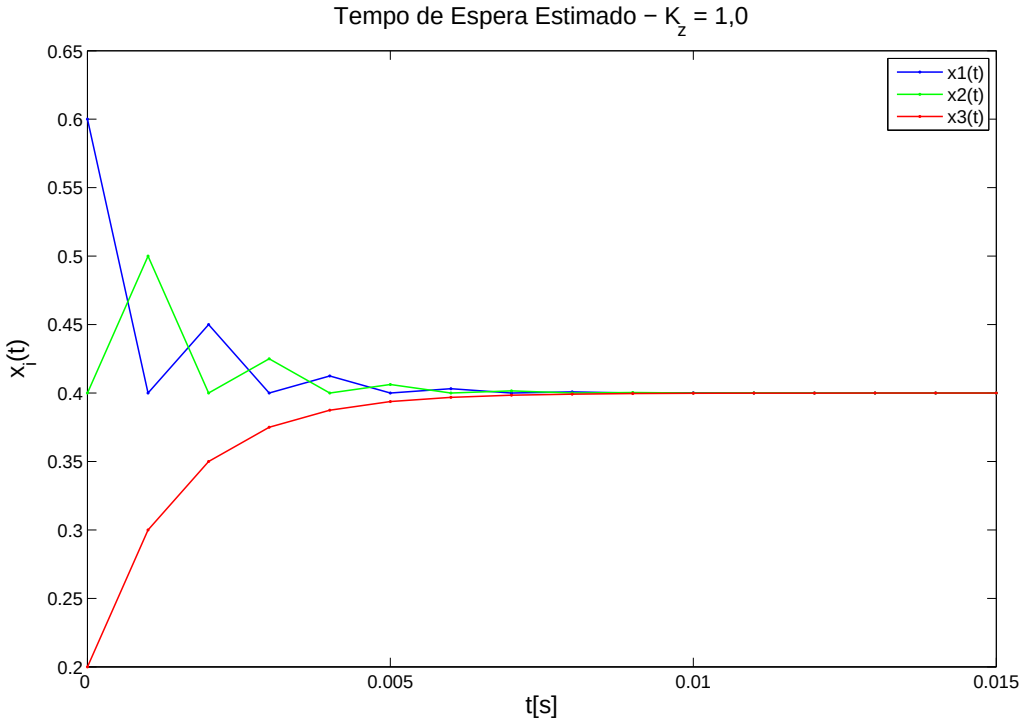

Figura 6.5: Experiência [01]: Resposta do modelo (2.1) - Tempo de espera estimado para $K_z = 1,0$.

6.3.2 Experimento [02]: Desempenho com a matriz de ganhos dada pela síntese por função custo quadrático

O objetivo deste experimento é o de apresentar os resultados obtidos quando utilizado a matriz de ganhos $K_{[04]}$ (5.61) sob as mesmas condições do experimento anterior, validando assim a metodologia proposta nesta tese.

Reescrevendo a matriz de ganhos $K_{[04]}$ aqui por conveniência, temos:

$$K_{[04]} = \begin{bmatrix} 3415,6 & -460,1 & -460,1 \\ -460,1 & 3415,6 & -460,1 \\ -460,1 & -460,1 & 3415,6 \end{bmatrix} \tag{6.8}$$

Normalizando a matriz, temos:

$$\overline{K}_{[04]} = \frac{K_{[04]}}{||K_{[04]}||_2} = \begin{bmatrix} 0,8813 & -0,1187 & -0,1187 \\ -0,1187 & 0,8813 & -0,1187 \\ -0,1187 & -0,1187 & 0,8813 \end{bmatrix} \tag{6.9}$$

A simulação neste experimento, ao contrário da subseção anterior, foi realizada antes do experimento, utilizando para isto os dados experimentais obtidos previamente no setup do ambiente computacional. Para esta simulação, utilizou-se um atraso de transferência médio de $481\mu s$ e $t_{p_1} = t_{p_2} = t_{p_3} = 9\mu s$.

A Figura 6.6 apresenta os resultados experimentais e simulado para o experimento [02]. Podemos verificar que os resultados experimentais apresentam melhoria no desempenho reduzindo as oscilações. Apesar do Maximum Overshoot ter aumentado em relação ao experimento [01] com $K_z = 1,0$ ($M_p = 0,4141$ no gráfico 6.6a), as oscilações foram reduzidas e o settling time obtido é de $t_s = 4ms$.

Isto comprova que os ganhos fora da diagonal principal da matriz de ganhos K podem contribuir para a melhoria do desempenho.

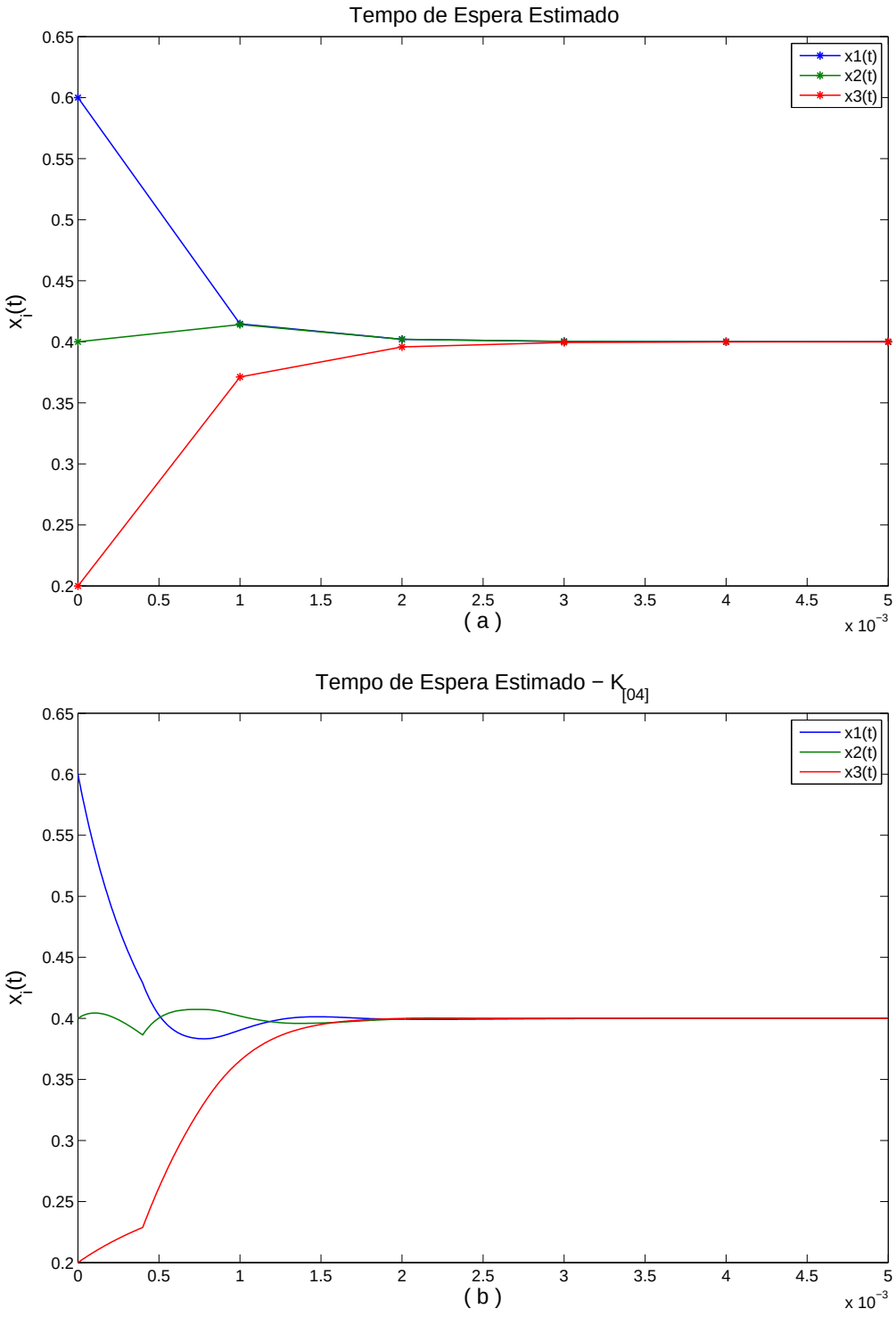

Figura 6.6: Experiência [02]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação para $K_{[04]}$ com h médio de $481\mu s$.

6.4 Experimentos em um Cluster Heterogêneo

Nesta seção, será demonstrado os experimentos realizados em um cluster heterogêneo. Para realizar os experimentos, o segundo e o terceiro nó foram substituídos por placas padrão PC-104 devido à disponibilidade no Instituto de Pesquisas da Marinha. Os modelos utilizados foram a Jaguar(Processador Intel Pentium III de 800MHz com 128Mb de memória) e a Bobcat (Processador Intel 586 de 133MHz com 128Mb de memória), ambas da Versalogic Corporation.

6.4.1 Experimento [03]

O objetivo deste experimento é o de apresentar os resultados obtidos para a execução do programa LoadBalancer para o caso heterogêneo. Novamente, os resultados obtidos utilizando-se uma matriz de ganhos diagonal e uma matriz de ganhos não diagonal são apresentados.

A determinação dos parâmetros t_{p_i} foi refeita pois as placas padrão PC-104 possuem freqüência de clock diferentes. Os valores medidos foram, aproximadamente, $t_{p_1} = 9\mu s$, $t_{p_2} = 65\mu s$ e $t_{p_3} = 93\mu s$.

Para os resultados apresentados na Figura 6.7, temos que o ganho discreto K_z utilizado foi $K_z = 0,8$ (resultado experimental) e a partir daí o Δt médio medido experimentalmente foi de aproximadamente 0,82 ms. Para calcularmos os ganhos para simulação, utilizou-se a seguinte forma:

$$K_i = 0,8/0,82ms = 975,6, \quad i = 1, 2, \ldots, n \tag{6.10}$$

Em seguida, utilizou-se a matriz de ganhos $K^1_{[03]}$ como sendo:

$$K^1_{[03]} = \begin{bmatrix} 975,6 & 0 & 0 \\ 0 & 975,6 & 0 \\ 0 & 0 & 975,6 \end{bmatrix} \tag{6.11}$$

A Figura 6.7a apresenta o resultado experimental para $K_z = 0,8$ e a Figura 6.7b apresenta a sua respectiva simulação. O settling time medido na Figura 6.7a é de 17ms.

Utilizando os parâmetros $\varepsilon = 10^{-14}$, $\Gamma = diag(10^{-10})$ e $\Omega = diag(15 \times 10^{-10})$, a solução

do problema dado pela Definição 5.3 fornece a seguinte matriz de ganhos $K^2_{[03]}$:

$$K^2_{[03]} = \begin{bmatrix} 747,2 & -861,9 & -440,0 \\ -876,4 & 6247,2 & -714,3 \\ -447,4 & -714,5 & 6366,5 \end{bmatrix} \tag{6.12}$$

A Figura 6.8 apresenta o resultado experimental e simulado para a matriz de ganhos normalizada $\overline{K}^2_{[03]}$. O settling time para o gráfico 6.8b é 7,2ms.

Para mostrar que os elementos fora da diagonal principal auxiliam na melhora do desempenho também para o caso heterogêneo, a Figura 6.9 apresenta o resultado experimental e simulado para a matriz $K^2_{[03]}$ com os elementos fora da diagonal principal nulos. Comparando a Figura 6.8 com a Figura 6.9, é possível verificar que existe uma diferença acentuada entre os gráficos 6.8b e 6.9b. O settling time no gráfico 6.9b é 8,8ms.

Apesar da diferença entre os resultados experimentais e seus respectivos resultados simulados, a comparação entre as Figuras 6.8a e 6.9a corroboram a observação anterior, mostrando que a proposta de um controlador com uma matriz de ganhos não-diagonal resulta em melhor desempenho no problema de balanceamento de carga.

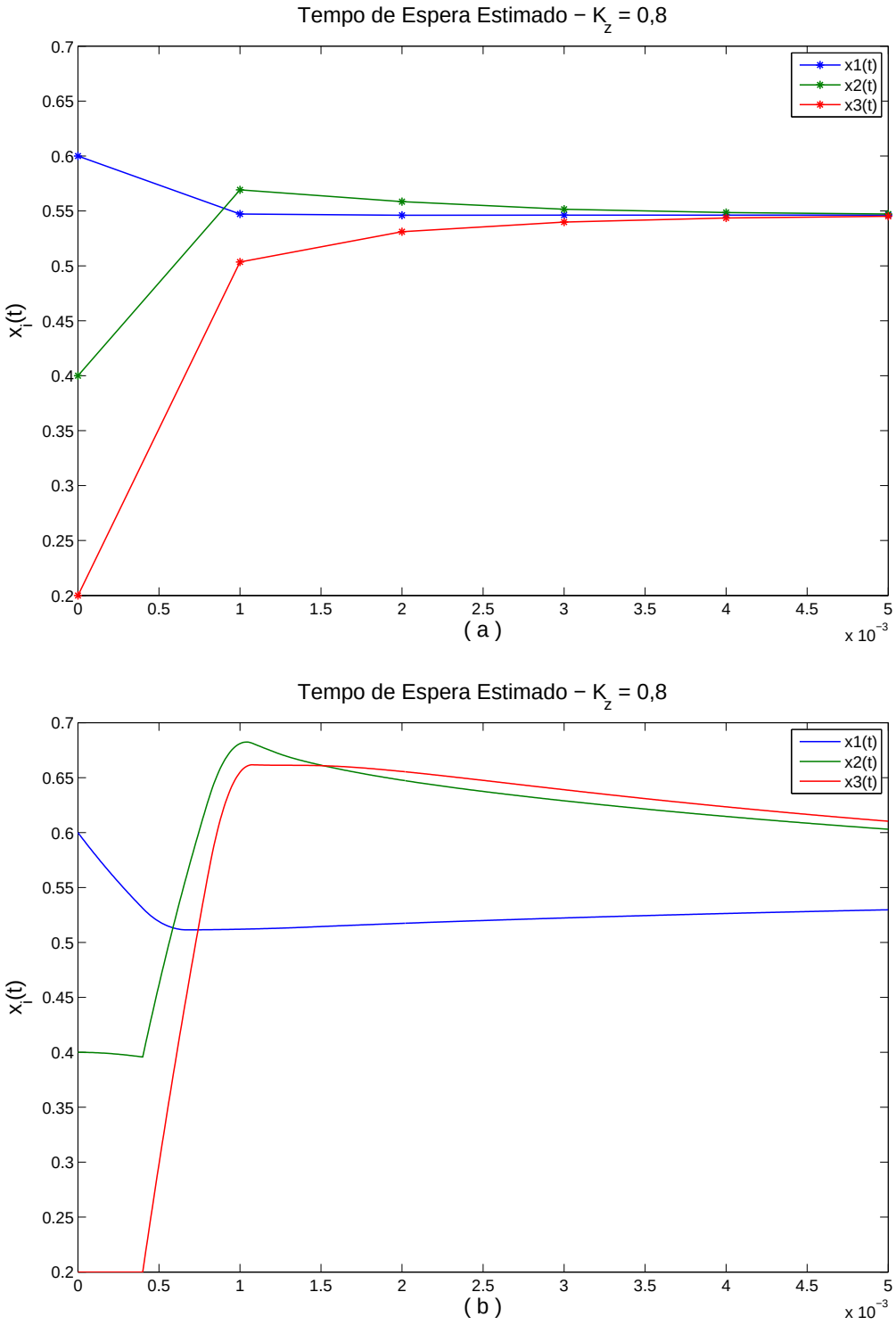

Figura 6.7: Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $K^1_{[03]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,82ms.

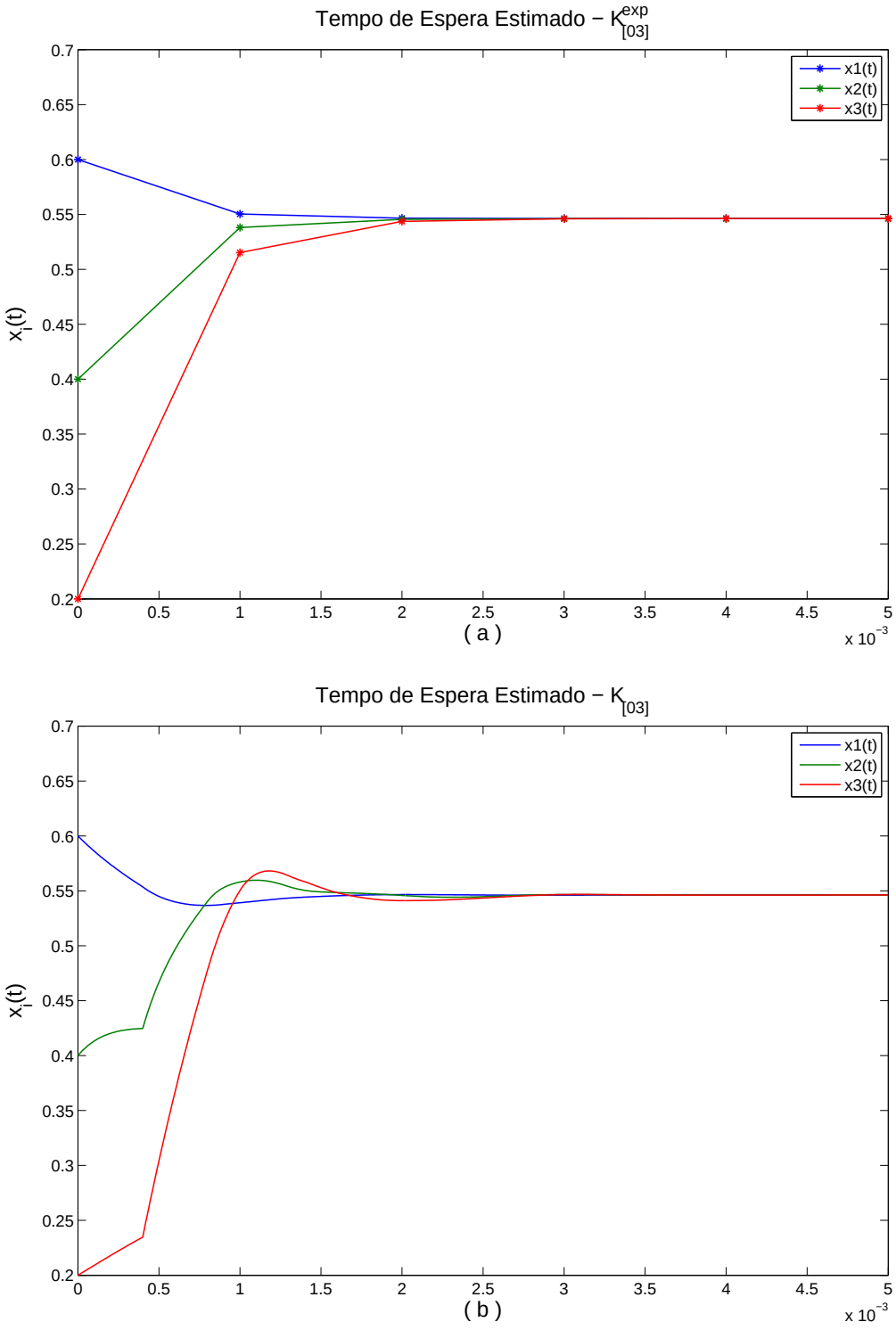

Figura 6.8: Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,28ms.

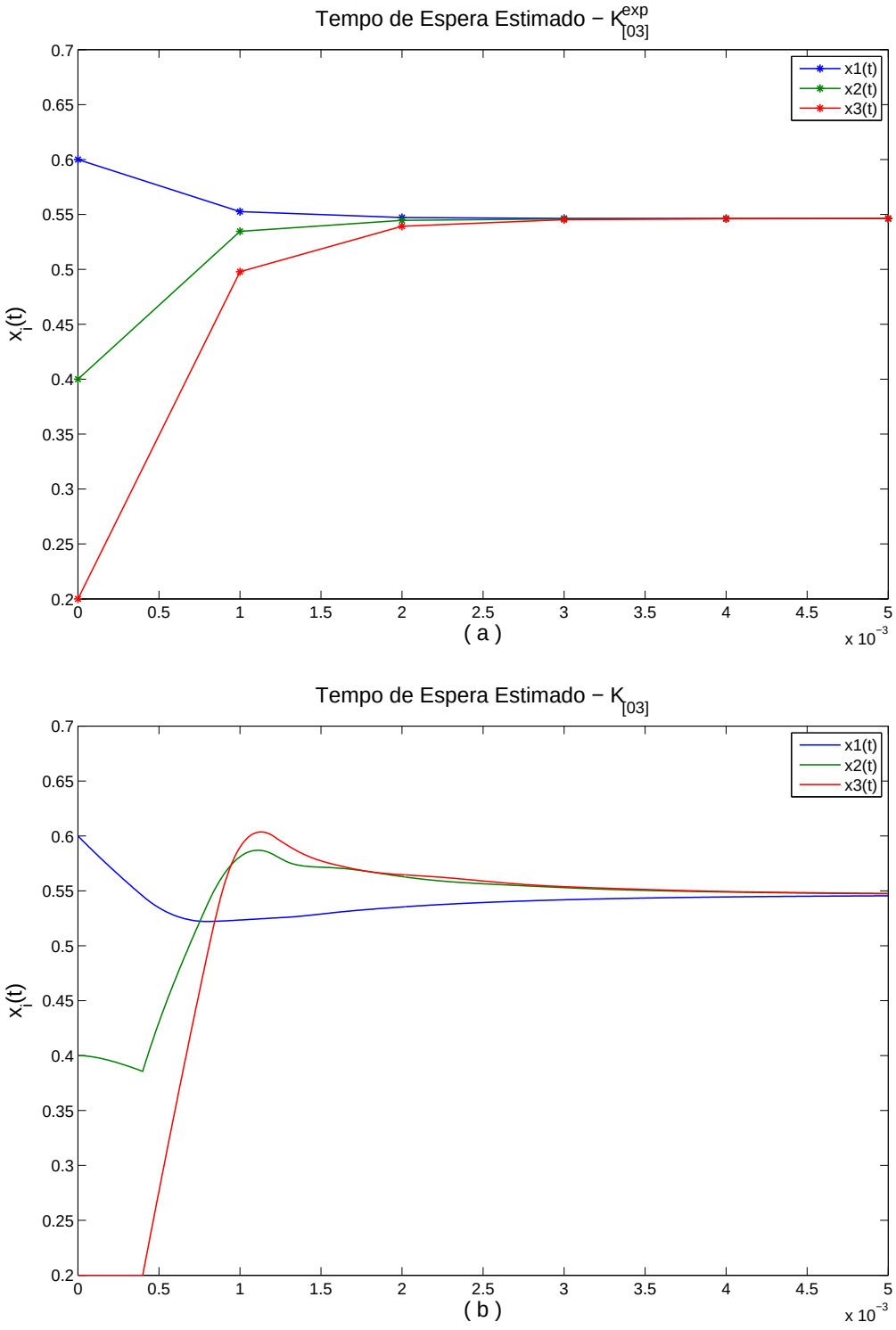

Figura 6.9: Experiência [03]: O gráfico (a) apresenta o tempo de espera estimado para $\overline{K}_{[04]}$ e o gráfico (b) apresenta a respectiva simulação com h médio de $481\mu s$ e Δ_t médio de 0,28ms.

6.5 Análise dos Resultados

Através das simulações e resultados experimentais, podemos verificar que aumentando os ganhos o sistema torna-se mais rápido, porém mais oscilatório também.

Um outro fator que agrava as oscilações é a dependência do atraso em função do número de tarefas a serem transmitidas. Como conseqüência, o algoritmo de balanceamento de carga é fortemente influenciado pela quantidade de carga. Por exemplo, se o desequilíbrio de carga for muito grande em algum momento, pode ser tentador distribuir a maior quantidade de tarefas possível para balancear o sistema rapidamente. Porém, como a largura de banda da rede é finita, existe aqui um ponto de inflexão onde uma maior quantidade de carga a ser distribuída leva mais tempo para alcançar o nó de destino e, desta forma, aumentando o tempo de processamento computacional. Tais resultados estão condizentes com os resultados encontrados por CHIASSON *et. al.* [33, 45, 46].

Outro item observado durante os experimentos é que as oscilações são maiores para tarefas com uma número maior de bytes. Durante os primeiros experimentos, adotou-se uma tarefa como tendo 400 bytes (era a própria matriz e não o expoente). A redução do tamanho da tarefa de 400 bytes para 4 bytes permitiu obter oscilações menores durante os experimentos.

Como explicado no quinto parágrafo da subseção 6.2.3, é fácil imaginar que é possível obter o melhor desempenho para o balanceamento de carga fazendo com que $K_z = 0,9$, ou até mesmo $K_z = 1,0$. Isto deve remover o maior número possível de tarefas excedentes da fila e redistribuí-las. Isto seria verdade se a rede possuísse largura de banda infinita, fazendo com que o atraso de transferência fosse nulo. Na verdade, para grandes valores do ganho discreto K_z, o atraso acaba reduzindo o desempenho, principalmente quando o atraso é grande e não são homogêneos.

Em [45], é mostrado que as melhores escolhas para a matriz de ganhos K_z, quando aplicado à grids, varia entre 0,5 e 0,6. Para cluster, este valor é de cerca de 0,7 a 0,8. A parti daí, o efeito de bursting ou ringing como é chamado por GHANEM [45], torna a convergência imprevisível.

6.6 Resumo do Capítulo

Neste capítulo, uma introdução à arquitetura de implementação, processos concorrentes (threads), a biblioteca MPI e o seu papel na implementação do algoritmo de balanceamento de carga. Em seguida, o setup experimental e a medida dos parâmetros experimentais foram apresentados.

A discretização da lei de controle para o caso da matriz de ganhos K não-diagonal foi apresentada e experimentos para o cluster homogêneo foram apresentados, primeiro para validar a metodologia publicada em CHIASSON *et al.* [33], e em segundo para validar a metodologia proposta nesta tese.

No caso da metodologia proposta por CHIASSON *et al.* [33], iniciou-se primeiro com o experimento para um determinado valor de K_z, gravando em arquivo e a cada iteração, os valores do intervalo de tempo entre sucessivas execuções do balanceamento de carga. A partir deste dado, e da estimativa de um h médio, os ganhos K_i foram encontrados e substituídos na simulação. Esta forma de realizar a simulação resultou em simulações mais fidedignas, já que o programa de balanceamento de carga desenvolvido (LoadBalancer) possui comunicação síncrona e não é possível estipular intervalos de tempo fixo para cada execução dele e em cada um dos nós, em outras palavras, o programa não é em tempo real.

A substituição de dois nós do cluster por placas padrão PC-104 mostraram-se boas opções devido à disponibilidade destas, facilidade no manuseio e compatibilidade com o sistema operacional LINUX instalados no cluster, não necessitando de praticamente nenhuma reconfiguração.

Experimentos para o caso homogêneo e heterogêneo foram apresentados e comparados a exemplos para matrizes de ganho diagonal, validando o resultado teórico apresentado nesta tese.

Capítulo 7

Conclusões e Trabalhos Futuros

Este trabalho teve por objetivo criar uma metodologia para determinar-se os melhores ganhos para o problema do balanceamento de carga em clusters heterogêneos, explorando e preenchendo uma lacuna nos artigos publicados por CHIASSON *et al.* [30, 33, 45, 90], o da síntese de controladores.

Nesta tese foram apresentados dois modelos matemáticos para o problema do balanceamento de carga entre processadores heterogêneos e suas similaridades com um terceiro modelo, o de rede neurais com atraso.

Através de uma mudanças de variáveis, foi mostrado que é possível transformar o modelo (2.1) publicado por CHIASSON *et al.* em um caso particular de uma rede neural de Hopfield com atraso e, a partir daí, utilizar a teoria de redes neurais para garantir a estabilidade e convergência na forma de uma LMI. Baseado nos trabalhos de SINGH [54, 70], um função de Lyapunov foi proposta e um critério de estabilidade para o modelo (3.30), na forma de uma LMI (5.21), foi obtido.

Diferentes versões foram analisadas, buscando-se aumentar o conhecimento e o comportamento do modelo (2.1), e uma função de custo quadrática foi utilizada de forma a garantir desempenho. Tal função foi expressa na forma de uma segunda LMI (5.55).

A solução numérica de ambas as LMIs, simultaneamente, permite encontrar a matriz de ganhos K (5.20).

Simulações e resultados experimentais puderam comprovar a eficácia do método proposto nesta tese.

Dentre as contribuições geradas pelo presente trabalho, podem ser mencionadas:

1. Formulação do problema de balanceamento de carga como um caso particular de uma rede neural com controles;

2. Melhoria no desempenho do processo de balanceamento de carga incluindo redução do efeito do bursting ou ringing;

3. Proposta de uma metodologia para síntese de controladores para o problema de balanceamento de carga;

4. Prova de estabilidade para a classe de sistemas Persidskii com atraso;

5. Abordagem computacional eficiente do problema de balanceamento de através do uso de LMI e do Método dos Pontos Interiores;

6. Extensão do modelo (2.1) utilizando uma matriz de ganhos K não-diagonal;

7.1 Trabalhos Futuros

Alguns tópicos que merecem investigação posterior e futuras pesquisas são:

1. Uso de uma função de custo quadrática que inclua o atraso de transferência, visando adequar esses resultados ao modelo, quando os atrasos tiverem valores significativos;

2. Transformação do problema de viabilidade da Definição (5.3) em um problema de otimização, a fim de eliminar o passo de ajustes nas matrizes Ω e Γ;

3. A busca de modificações no problema dado pela Definição (5.3) de forma que os elementos da matriz de ganhos K já estivessem contidos no intervalo $0 < K_{z_i} < 1$ contornaria o o passo de normalização da matriz de ganhos K;

4. Paralelização do Método dos Pontos Interiores, aumentando o *speed-up* da solução numérica do problema;

5. Incluir no modelo os atrasos de comunicãção (τ_{ij}) que foram desconsiderados neste trabalho;

6. Considerar atrasos não-homogêneos (h_{ij}) para o caso dos grids;

7. Utilizar modelo de balanceamento de carga simultaneamente ao processamento de tarefas (caso dos processadores com tecnologia *Hyper-Threading*;

8. Investigar a descentralização do modelo através do particionamento do cluster em subclusters e avaliar o desempenho;

9. Testes extensivos em outros ambientes computacionais.

Apêndice A

Prova de Conservação do Número Total de Tarefas

Conforme dito anteriormente no capítulo 2, na subseção 2.2.1, a proposição desta tese em utilizarmos uma matriz de ganhos mais geral necessita de uma análise quanto à conservação do número total de tarefas. Para isto, vamos analisar a conservação para o caso $n = 2$. Em seguida, faremos a mesma análise na forma matricial visando facilitar o entendimento. Depois a análise da conservação na forma matricial será realizada para $n = 3$, onde aproveitaremos para verificar a possibilidade de termos duas matrizes de ganho diferentes no modelo **MRN** (3.30), K e K^{τ}. O interesse em duas matrizes de ganhos diferentes é flexibilizar a ação de controle, aumentando o grau de liberdade dos controladores.

Por fim, estenderemos a mesma análise para o caso n qualquer e concluiremos que a conservação sempre se mantém.

Seja o modelo dado em (2.1) para $n = 2$:

$$\begin{cases} \dot{x}_1(t) = & -K_{11}\phi(y_1(t)) + p\frac{t_{p_1}}{t_{p_2}}K_{22}\phi(y_2(t-h)) \\ \dot{x}_2(t) = & -K_{22}\phi(y_2(t)) + p\frac{t_{p_2}}{t_{p_1}}K_{11}\phi(y_1(t-h)) \end{cases} \tag{A.1}$$

Dividindo a primeira equação de (A.1) por t_{p_1} e a segunda equação de de (A.1) por t_{p_2}, teremos as equações dinâmicas do tamanho das filas dada em número de tarefas:

$$\begin{aligned}
\dot{q}_1(t) &= \underbrace{-\frac{K_{11}}{t_{p_1}}\phi(y_1(t))}_{B_1} + \underbrace{p\frac{K_{22}}{t_{p_2}}\phi(y_2(t-h))}_{A_4} \\
\dot{q}_2(t) &= \underbrace{-\frac{K_{22}}{t_{p_2}}\phi(y_2(t))}_{A_1} + \underbrace{p\frac{K_{11}}{t_{p_1}}\phi(y_1(t-h))}_{B_4}
\end{aligned} \tag{A.2}$$

onde $q_1(t)$ e $q_2(t)$ correspondem ao número de tarefas nas filas 1 e 2, respectivamente.

Segundo Chiasson [33, 65] o modelo dinâmico do comprimento das filas na rede é dado por:

$$\dot{q}_{net_i}(t) = -\sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}K_{jj}\phi(y_j(t-h)) + \sum_{j=1}^{n}\frac{p_{ij}}{t_{p_j}}K_{jj}\phi(y_j(t)) \tag{A.3}$$

Utilizando a equação acima para o exemplo, temos as equações dinâmicas das filas na rede:

$$\begin{aligned}
\dot{q}_{net_1}(t) &= \underbrace{-p\frac{K_{22}}{t_{p_2}}\phi(y_2(t-h))}_{A_3} + \underbrace{p\frac{K_{22}}{t_{p_2}}\phi(y_2(t))}_{A_2} \\
\dot{q}_{net_2}(t) &= \underbrace{-p\frac{K_{11}}{t_{p_1}}\phi(y_1(t-h))}_{B_3} + \underbrace{p\frac{K_{11}}{t_{p_1}}\phi(y_1(t))}_{B_2}
\end{aligned} \tag{A.4}$$

A equação (A.3) nos diz que q_{net_i} é o número de tarefas inseridas na rede com destino ao *i-ésimo* nó, ou seja, o j-ésimo nó está inserindo tarefas na rede com destino ao *i-ésimo* nó na taxa de $(p_{ij}t_{p_i}/t_{p_j}K_{jj}\phi(y_j(t)))$, enquanto o *i-ésimo* nó está retirando estas mesmas tarefas da rede a uma taxa de $(p_{ij}t_{p_i}/t_{p_j}K_{jj}\phi(y_j(t-h)))$. De uma forma mais simples: Tudo o que entra em direção ao *i-ésimo* nó, será retirado por este mesmo nó com um atraso de h segundos. Este é o modelo dinâmico da rede.

As funções $\phi(y_i(t))$ e $\phi(y_i(t-h))$ atuam como chaves "liga-desliga", selecionando que termos serão utilizados para ceder e que termos serão utilizados para receber.

A figura (A.1) ilustra a troca de tarefas entre as equações (A.2) e (A.4) para o caso onde $q_2(t) > q_1(t)$. A troca poderá ser acompanhada pela seqüência de termos $A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4$, quando $q_2(t) > q_1(t)$, e pela seqüência $B_1 \rightarrow B_2 \rightarrow B_3 \rightarrow B_4$, quando $q_1(t) > q_2(t)$.

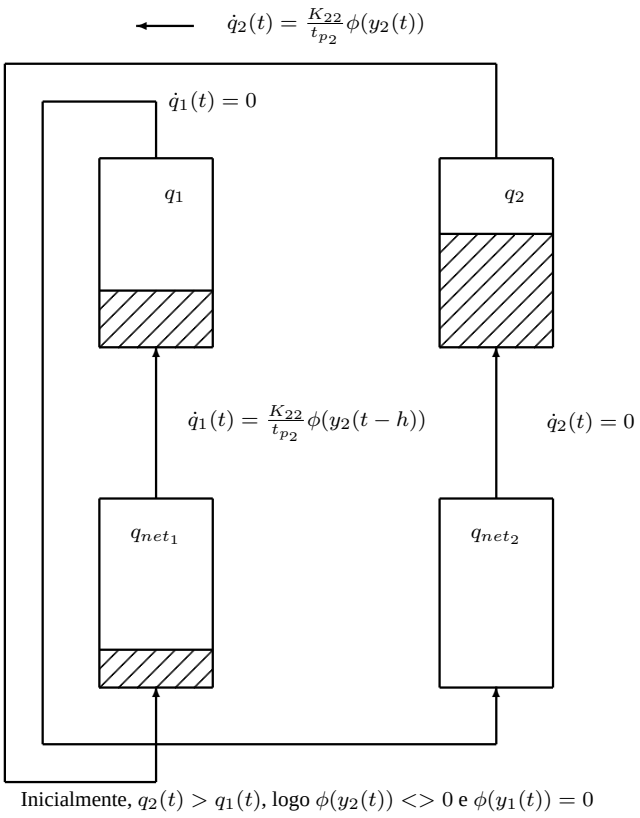

Figura A.1: Diagrama do balanceamento de carga entre dois nós com atraso

Neste contexto, associaremos a palavra “ceder” a um termo negativo, enquanto a palavra “receber” será associada a um termo positivo.

Segundo Chiasson [65], para que haja conservação do número total de tarefas, é necessário e suficiente que a seguinte restrição seja atendida:

$$\sum_{i=1}^{n}(\dot{q}_i(t) + \dot{q}_{net_i}(t)) = 0 \tag{A.5}$$

Logo, de (A.5), a conservação de tarefas corresponde a uma soma de parcelas carregando seus respectivos sinais. E lembrando que $p = 1$ para este caso, temos:

$$(A_1 + A_2) + (A_3 + A_4) + (B_1 + B_2) + (B_3 + B_4) = 0 \tag{A.6}$$

Substituindo as parcelas $A_1, \ldots, A_4$ e $B_1, \ldots, B_4$, definidas nas equações (A.2) e (A.4), podemos verificar que a soma realmente se anula, mostrando o princípio de funcionamento da troca e da conservação de tarefas no modelo de CHIASSON *et al.* [65].

No entanto, seria interessante repetirmos esta mesma análise na forma matricial. Isto

ajuda na análise do caso para n qualquer.

Colocando (A.1) na forma matricial, teremos:

$$\begin{aligned}\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= -\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ &+\begin{bmatrix} 0 & p\frac{t_{p_1}}{t_{p_2}} \\ p\frac{t_{p_2}}{t_{p_1}} & 0 \end{bmatrix}\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix}\end{aligned} \tag{A.7}$$

Multiplicando os dois termos da equação (A.7) à esquerda por

$$\Upsilon_2 = \begin{bmatrix} \frac{1}{t_{p_1}} & 0 \\ 0 & \frac{1}{t_{p_2}} \end{bmatrix} \tag{A.8}$$

temos a taxa de variação da quantidade de tarefas em cada uma das filas, já que $x_i(t)/t_{p_i} = q_i(t)$.

$$\begin{aligned}\begin{bmatrix} \dot{q}_1(t) \\ \dot{q}_2(t) \end{bmatrix} &= -\begin{bmatrix} \frac{1}{t_{p_1}} & 0 \\ 0 & \frac{1}{t_{p_2}} \end{bmatrix}\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ &+\begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix}\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix}\end{aligned} \tag{A.9}$$

O primeiro termo da equação (A.9) é subtraído do segundo termo, significando “cessão” de tarefas e “recepção” de tarefas, respectivamente.

Reescrevendo a equação (A.4) na forma matricial, teremos:

$$\begin{aligned}\begin{bmatrix} \dot{q}_{net_1}(t) \\ \dot{q}_{net_2}(t) \end{bmatrix} &= \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix}\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ &-\begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix}\begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix}\begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix}\end{aligned} \tag{A.10}$$

Para haver conservação de tarefas, é necessário que a equação (A.5) seja satisfeita. Para isto, a soma das equações de (A.9) e das equações de (A.10) deve ser nula.

Podemos notar que o termo com atraso de (A.9) se cancelará com o termo com atraso de (A.10) pois elas são iguais e com sinais diferentes.

Logo, apenas os termos sem atraso contribuem para a soma:

$$\sum_{i=1}^{2}(\dot{q}_i(t)+\dot{q}_{net_i}(t)) = \left(\begin{bmatrix} 0 & \frac{1}{t_{p_2}} \\ \frac{1}{t_{p_1}} & 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{t_{p_1}} & 0 \\ 0 & \frac{1}{t_{p_2}} \end{bmatrix} \right) \begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} \tag{A.11}$$

$$\sum_{i=1}^{2}(\dot{q}_i(t)+\dot{q}_{net_i}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}} & \frac{1}{t_{p_2}} \\ \frac{1}{t_{p_1}} & -\frac{1}{t_{p_2}} \end{bmatrix} \begin{bmatrix} K_{11} & 0 \\ 0 & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} \tag{A.12}$$

Efetuando os produtos matriciais em (A.18), temos:

$$\sum(\dot{q}(t)+\dot{q}_{net}(t)) = \begin{bmatrix} -\frac{K_{11}}{t_{p_1}}\phi(y_1(t)) + \frac{K_{22}}{t_{p_2}}\phi(y_2(t)) \\ \frac{K_{11}}{t_{p_1}}\phi(y_1(t)) - \frac{K_{22}}{t_{p_2}}\phi(y_2(t)) \end{bmatrix} \tag{A.13}$$

A soma das duas componentes do vetor acima é nula, mostrando que há conservação de tarefas.

Propondo duas novas matrizes genéricas de ganho no lugar da matriz de ganhos diagonal, K e K^τ, e substituindo-as na equação (A.7), teremos o novo modelo simplificado na forma matricial:

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= -\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ &+ \begin{bmatrix} 0 & p\frac{t_{p_1}}{t_{p_2}} \\ p\frac{t_{p_2}}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K^\tau_{11} & K^\tau_{12} \\ K^\tau_{21} & K^\tau_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix} \end{aligned} \tag{A.14}$$

Vamos verificar se:

1. Se a conservação de tarefas ainda se mantém para uma matriz de ganhos não-diagonal;
2. Se a utilização de duas matrizes de ganhos K e K^τ ainda permite a conservação de tarefas.

Caso um item ou mais se mostre verdadeiro, isto ajudará na síntese de controladores menos restritivos.

Multiplicando a equação (A.14) à esquerda pela matriz (A.8), teremos a taxa de variação do comprimento das filas:

$$\begin{bmatrix} \dot{q}_1(t) \\ \dot{q}_2(t) \end{bmatrix} = -\begin{bmatrix} \frac{1}{t_{p_1}} & 0 \\ 0 & \frac{1}{t_{p_2}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ + \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K_{11}^{\tau} & K_{12}^{\tau} \\ K_{21}^{\tau} & K_{22}^{\tau} \end{bmatrix} \begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix} \tag{A.15}$$

O modelo de filas na rede é obtido substituindo-se K e K^{τ} na equação (A.10), obtendo:

$$\begin{bmatrix} \dot{q}_{net_1}(t) \\ \dot{q}_{net_2}(t) \end{bmatrix} = \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + \\ - \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K_{11}^{\tau} & K_{12}^{\tau} \\ K_{21}^{\tau} & K_{22}^{\tau} \end{bmatrix} \begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix} \tag{A.16}$$

Podemos verificar que, para que a restrição (A.5) seja satisfeita e, de acordo com a figura, é necessário que o termo positivo da equação (A.16) torne-se o termo negativo dela mesmo após h segundos (ou seja, o que entra na fila da rede, deverá ser o mesmo que sai defasado de h segundos no tempo).

Desta forma, se entrarmos com um valor multiplicado por K na rede, deverá sair K e não K^{τ}, violando com isto a equação (A.5).

Logo, para mantermos a conservação de tarefas, é necessário primeiro que $K = K^{\tau}$, ou seja, o segundo item da lista não se verifica.

Já o primeiro item da lista é possível, pois a equação (A.16) torna-se:

$$\begin{bmatrix} \dot{q}_{net_1}(t) \\ \dot{q}_{net_2}(t) \end{bmatrix} = \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} + - \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} \\ p\frac{1}{t_{p_1}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t-h)) \\ \phi(y_2(t-h)) \end{bmatrix} \quad \text{(A.17)}$$

que, ao ser somada à equação (A.14), já com $K = K^\tau$, resulta em

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}} & \frac{1}{t_{p_2}} \\ \frac{1}{t_{p_1}} & -\frac{1}{t_{p_2}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} \phi(y_1(t)) \\ \phi(y_2(t)) \end{bmatrix} = 0 \quad \text{(A.18)}$$

Efetuando as multiplicações matriciais acima, temos:

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} (-\frac{K_{11}}{t_{p_1}} + \frac{K_{21}}{t_{p_2}})\phi(y_1(t)) + (-\frac{K_{12}}{t_{p_1}} + \frac{K_{22}}{t_{p_2}})\phi(y_2(t)) \\ (\frac{K_{11}}{t_{p_1}} - \frac{K_{21}}{t_{p_2}})\phi(y_1(t)) + (\frac{K_{12}}{t_{p_1}} - \frac{K_{22}}{t_{p_2}})\phi(y_2(t)) \end{bmatrix} \quad \text{(A.19)}$$

Somando as duas linhas, do vetor acima, o resultado é nulo, o que nos mostra que ainda há conservação para uma matriz K genérica.

Vamos analisar a proposta para o caso onde $n = 3$, a fim de generalizar para n qualquer.

Desenvolvendo o modelo base (3.1) para $n = 3$ e colocando-o já na forma matricial, teremos:

$$\dot{x}(t) = - \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t)) + + \begin{bmatrix} 0 & p\frac{t_{p_1}}{t_{p_2}} & \frac{t_{p_1}}{t_{p_3}} \\ p\frac{t_{p_2}}{t_{p_1}} & 0 & p\frac{t_{p_2}}{t_{p_3}} \\ p\frac{t_{p_3}}{t_{p_1}} & p\frac{t_{p_3}}{t_{p_2}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t-h)) \quad \text{(A.20)}$$

Multiplicando a equação (A.20) acima por $\Upsilon_3 = \textit{diag}(1/t_{p_1}, 1/t_{p_2}, 1/t_{p_3})$:

$$\begin{aligned}\dot{q}(t) &= -\begin{bmatrix} \frac{1}{t_{p_1}} & 0 & 0 \\ 0 & \frac{1}{t_{p_2}} & 0 \\ 0 & 0 & \frac{1}{t_{p_3}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t)) + \\ &+ \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & 0 & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t-h)) \end{aligned} \quad \text{(A.21)}$$

As equações dinâmicas das filas na rede são dadas por:

$$\begin{aligned}\dot{q}_{net}(t) &= \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & 0 & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t)) + \\ &- \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & 0 & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t-h)) \end{aligned} \quad \text{(A.22)}$$

A soma de $\dot{q}(t)$ e $\dot{q}_{net}(t)$ resume-se a:

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & -\frac{1}{t_{p_2}} & p\frac{1}{t_{p_3}} \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & -\frac{1}{t_{p_3}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \phi(y(t)) \quad \text{(A.23)}$$

Como neste caso, $p = 0,5$, então a soma acima também é nula, o que prova a conservação de tarefas.

Podemos, por indução a partir daí, generalizar para um n qualquer.

$$\dot{x}(t) = -\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t)) + \\ + \begin{bmatrix} 0 & p\frac{t_{p_1}}{t_{p_2}} & \cdots & \frac{t_{p_1}}{t_{p_n}} \\ p\frac{t_{p_2}}{t_{p_1}} & 0 & \cdots & p\frac{t_{p_2}}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{t_{p_n}}{t_{p_1}} & p\frac{t_{p_n}}{t_{p_2}} & \cdots & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t-h)) \quad \text{(A.24)}$$

Multiplicando a equação (A.24) acima por $\Upsilon_n = \mathit{diag}(1/t_{p_1}, 1/t_{p_2}, \ldots, 1/t_{p_n})$:

$$\dot{q}(t) = -\begin{bmatrix} \frac{1}{t_{p_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{t_{p_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{t_{p_n}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t)) + \\ + \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & \cdots & \frac{1}{t_{p_n}} \\ p\frac{1}{t_{p_1}} & 0 & \cdots & p\frac{1}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t-h)) \quad \text{(A.25)}$$

As equações dinâmicas das filas na rede é dada por:

$$\dot{q}_{net}(t) = \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & \cdots & \frac{1}{t_{p_n}} \\ p\frac{1}{t_{p_1}} & 0 & \cdots & p\frac{1}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t)) + - \begin{bmatrix} 0 & p\frac{1}{t_{p_2}} & \cdots & \frac{1}{t_{p_n}} \\ p\frac{1}{t_{p_1}} & 0 & \cdots & p\frac{1}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t-h)) \quad \text{(A.26)}$$

Somando as equações (A.25) e (A.26), de acordo com a equação (A.5), teremos:

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & p\frac{1}{t_{p_n}} \\ p\frac{1}{t_{p_1}} & -\frac{1}{t_{p_2}} & \cdots & p\frac{1}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & -\frac{1}{t_{p_n}} \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \phi(y(t)) \quad \text{(A.27)}$$

Se multiplicarmos a matriz de ganhos K pela função vetorial $\phi(y(t))$, teremos que os vetores $\phi(y(t))$ serão os coeficientes de combinações lineares da matriz K:

$$\begin{bmatrix} \varphi_1(K,\phi) \\ \varphi_2(K,\phi) \\ \vdots \\ \varphi_n(K,\phi) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} K_{1i}\phi(y_i(t)) \\ \sum_{i=1}^{n} K_{2i}\phi(y_i(t)) \\ \vdots \\ \sum_{i=1}^{n} K_{ni}\phi(y_i(t)) \end{bmatrix} \quad \text{(A.28)}$$

Substituindo a equação matricial (A.28) em (A.27), teremos:

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & p\frac{1}{t_{p_n}} \\ p\frac{1}{t_{p_1}} & -\frac{1}{t_{p_2}} & \cdots & p\frac{1}{t_{p_n}} \\ \vdots & \vdots & \ddots & \vdots \\ p\frac{1}{t_{p_1}} & p\frac{1}{t_{p_2}} & \cdots & -\frac{1}{t_{p_n}} \end{bmatrix} \begin{bmatrix} \varphi_1(K,\phi) \\ \varphi_2(K,\phi) \\ \vdots \\ \varphi_n(K,\phi) \end{bmatrix} \quad \text{(A.29)}$$

Multiplicando a matriz quadrada pelo vetor $\varphi(K,\phi)$, temos:

$$\sum(\dot{q}(t) + \dot{q}_{net}(t)) = \begin{bmatrix} -\frac{1}{t_{p_1}}\varphi_1 + p\frac{1}{t_{p_2}}\varphi_2 + \cdots + p\frac{1}{t_{p_n}}\varphi_n \\ p\frac{1}{t_{p_1}}\varphi_1 + -\frac{1}{t_{p_2}}\varphi_2 + \cdots + p\frac{1}{t_{p_n}}\varphi_n \\ \vdots \\ p\frac{1}{t_{p_1}}\varphi_1 + p\frac{1}{t_{p_2}}\varphi_2 + \cdots + -\frac{1}{t_{p_n}}\varphi_n \end{bmatrix} \tag{A.30}$$

Como devemos somar as componentes dos vetores $q(t)$ e q_{net}, isto significa que devemos somar as componentes do vetor acima. Lembrando que $p = 1/(n-1)$ e, por conseqüência, $\sum_{i=1}^{n-1} p = 1$, então a soma é nula, provando assim que há conservação de tarefas para uma ordem n qualquer.

Podemos observar também que a *conservação do número total de tarefas* independe das funções não-lineares.

Apêndice B

Algoritmos Genéticos

Algoritmos Genéticos são métodos de otimização heurísticos baseados no sucesso do processo da evolução biológica das espécies através dos mecanismos de seleção natural, recombinação genética e de mutação [91].

Na natureza, os indivíduos de uma mesma espécie são forçados a competir por recursos limitados, tais como alimento, espaço e parceiros. Em geral, os indivíduos melhor adaptados às condições hostis de seu meio ambiente tendem a sobreviver por tempo suficiente para se reproduzir e deixar um número maior de descendentes, enquanto que outros indivíduos menos aptos morrem sem deixar descendentes.

As diversas características que constituem cada indivíduo são determinadas por seu conteúdo genético. O conjunto destas características determina uma aptidão maior ou menor para a sobrevivência no meio ambiente onde o indivíduo está inserido, resultando em uma probabilidade maior ou menor de este indivíduo sobreviver e deixar descendentes em gerações posteriores.

No processo de reprodução sexuada, o material genético de dois indivíduos é recombinado, gerando uma nova combinação dos genes destes dois indivíduos. Esta nova combinação de genes determinará as características do filhote e sua conseqüente aptidão face ao meio ambiente em que deverá viver e se reproduzir.

Assim, os genes presentes nos indivíduos mais aptos tendem a continuar existindo na população, replicando-se nos descendentes destes indivíduos, enquanto que os genes presentes em indivíduos menos aptos tendem a desaparecer.

O processo de recombinação de genes é imperfeito, na medida em que uma pequena par-

cela dos genes herdados diretamente dos pais pode sofer mutações aleatórias, conduzindo ao surgimento de novos genes, diferentes daqueles herdados dos pais, possibilitando a geração eventual de filhotes que possuam características novas, inexistentes nos indivíduos das gerações anteriores.

Grande parte destas mutações conduz a características indesejadas, que podem dificultar ou inviabilizar a sobrevivência e reprodução do filhote, de modo que estes genes mutantes tendem a ser naturalmente descartados da população.

Eventualmente, algumas mutações oportunas podem produzir características que tendem a contribuir para melhorar a aptidão do indivíduo, aumentando sua possibilidade de sobreviver e de gerar um maior número de descendentes. Nestes casos, os genes mutantes tendem a se replicar nos descendentes do indivíduo favorecido, aumentando sua incidência nas gerações subseqüentes e podendo, a longo prazo, tornar-se predominantes na população.

Este processo cíclico de seleção, recombinação e mutação de genes promove a evolução natural do conteúdo genético de uma população em direção a indivíduos cada vez mais aptos a sobreviver e se reproduzir.

Podem ser facilmente observadas na natureza um grande número de soluções complexas e altamente engenhosas para o problema de otimização da aptidão dos seres vivos ao seu meio ambiente. Um dos exemplos mais notáveis é o sentido da visão, que evoluiu em grande parte das espécies mais complexas de seres vivos.

Inspirado neste processo de evolução biológica das espécies, Holland [91] propôs em 1975 algoritmos computacionais que imitam o processo evolutivo da natureza, com o objetivo de solucionar problemas de otimização complexos e dificilmente tratáveis por outros métodos. Por utilizarem uma técnica semelhante àquela adotada pela natureza para otimizar a aptidão dos seres vivos para sobreviver e procriar, tais algoritmos são usualmente denominados algoritmos genéticos.

Os algoritmos genéticos manipulam uma população de potenciais soluções para um problema de otimização.

Tal como na natureza, os parâmetros que definem cada solução particular em um algoritmo genético são codificados em genes.

Enquanto na natureza os genes de cada indivíduo particular são codificados por seqüências de nucleotídeos em uma molécula de DNA, nos algoritmos genéticos os parâmetros de cada solução-candidata particular são geralmente codificados por meio de cadeias de dígitos

binários (ou outro tipo de representação numérica), também denominadas genes.

Tal como na natureza, os algoritmos genéticos provêem um mecanismo de recombinação de genes, que possibilita a construção de uma terceira solução-candidata (solução-filhote) a partir da recombinação das seqüências genéticas de duas soluçõescandidatas pré-existentes.

Tal como na natureza, os algoritmos genéticos também provêem um mecanismo de mutação aleatória, que possibilita a produção eventual de genes mutantes por meio da introdução de pequenas alterações aleatórias nos genes das soluções-filhotes.

Tal como na natureza, os algoritmos genéticos provêem um mecanismo de seleção que favorece as soluções-candidatas mais aptas a otimizar a função-objetivo do problema, aumentando as possibilidades de estas gerarem um maior número de descendentes na geração subseqüente, e desfavorece as menos aptas, diminuindo as possibilidades de estas gerarem descendentes.

Isto é feito estabelecendo-se uma função de adequação, que atribui um valor de aptidão a cada solução-candidata com base nos parâmetros codificados em seus genes, e estabelecendo-se também uma relação apropriada entre o valor de aptidão de uma solução-candidata e a probabilidade de esta solução-candidata contribuir com um descendente para a geração seguinte.

As principais características que distinguem os algoritmos genéticos das técnicas convencionais de otimização são:

- O processo de busca de uma solução opera com uma população de soluções candidatas, e não apenas com uma solução de cada vez;
- São processadas apenas representações codificadas (genótipos) das soluções candidatas, e não as as próprias soluções (fenótipos);
- As regras de transição de um conjunto de soluções para outro são probabilísticas, e não determinísticas;
- O único requisito exigido para a função-objetivo é que ela possa ser calculada a partir da representação codificada de uma solução, não sendo requeridas propriedades especiais tais como convexidade e diferenciabilidade.

Os ingredientes necessários para se formular um algoritmo genético são:

- Uma população inicial de soluções-candidatas;
- Um mecanismo de codificação;

- Uma função de adequação;
- Um mecanismo de seleção;
- Um mecanismo de recombinação;
- Um mecanismo de mutação;
- Um critério de terminação.

B.1 População Inicial

A população inicial é geralmente escolhida de forma aleatória, atribuindo-se a cada parâmetro de cada solução-candidata um valor aleatório uniformemente distribuído dentro de seu domínio de valores permitido.

Em problemas específicos, pode-se manipular as probabilidades de escolha de modo a que a população inicial apresente maior incidência de parâmetros em faixas de valores apropriados previamente conhecidos. Pode-se também incluir soluções de boa qualidade previamente conhecidas, ou obtidas por outros métodos.

A dimensão da população (número de indivíduos) é um parâmetro importante na elaboração de um algoritmo genético. Uma população pequena apresenta a vantagem de possibilitar uma convergência mais rápida do algoritmo, ao custo de um maior risco de convergência prematura para soluções sub-ótimas de baixa qualidade. Por outro lado, uma população grande provê maior diversidade genética e, conseqüentemente, maior chance de convergência para a solução ótima ou para soluções sub-ótimas de boa qualidade. A contrapartida é um maior esforço computacional, devido tanto ao maior número de soluções-candidatas a serem processadas em cada geração como ao maior número de gerações necessárias para convergência.

É importante observar que, ao contrário do que ocorre com populações de seres vivos na natureza, o número de indivíduos da população é geralmente mantido constante nos algoritmos genéticos, por razões de conveniência computacional.

B.2 Mecanismo de Codificação

O mecanismo de codificação determina a forma pela qual cada possível solução candidata é mapeada por uma seqüência de genes.

O mecanismo de codificação das soluções-candidatas depende da natureza das variáveis do problema. Para problemas combinatórios, como, por exemplo, o clássico problema do caixeiro viajante, os genes podem ser representados por dígitos binários significando a inclusão ou não de uma borda no circuito hamiltoniano. Para problemas contínuos, os genes podem assumir valores contínuos, como, por exemplo, a intensidade do fluxo em diversos canais em problemas de otimização de fluxo em sistemas de transporte.

O requisito fundamental de um mecanismo de codificação é que cada possível solução candidata seja mapeada por uma única combinação de genes.

B.3 Função de Adequação

A função de adequação determina o valor de aptidão atribuído a cada solução-candidata, calculado com base em seus parâmetros, codificados nos genes.

O valor de aptidão deve medir o potencial de uma solução-candidata para ser a solução ótima do problema.

Freqüentemente, a função de adequação é definida como sendo a própria função-objetivo do problema de otimização que se deseja resolver, ou alguma versão apropriadamente transformada desta função-objetivo.

Entretanto, a função de adequação deve ser definida para toda e qualquer combinação possível de parâmetros de solução-candidata, de modo que seu valor deve diferir da função-objetivo no caso de soluções que não satisfaçam as restrições do problema (soluções inviáveis).

Em geral, a função de adequação deve ser formulada de modo que soluções inviáveis possuam um valor de aptidão inferior a qualquer solução viável. Adicionalmente, é desejável que este valor seja maior para soluções inviáveis próximas ao limiar de viabilidade e menor para soluções inviáveis afastadas deste limiar, visando prover seletividade entre soluções inviáveis no sentido de conduzir o deslocamento da população rumo à região viável.

B.4 Mecanismo de Seleção

O mecanismo de seleção deve ser elaborado de modo que soluções-candidatas com maior valor de aptidão tenham uma maior probabilidade de contribuir com um descendentes para a

geração seguinte que soluções-candidatas com menor valor de aptidão.

Três mecanismos de seleção freqüentemente utilizados são:

- Seleção proporcional;
- Seleção por torneio;
- Seleção por truncamento.

No mecanismo de seleção proporcional, os dois progenitores de cada solução-filhote a ser inserida na geração subseqüente são escolhidos dentre toda a população com uma probabilidade dada pela razão entre seu valor de aptidão e a soma dos valores de aptidão de toda a população.

Ou seja, na seleção proporcional, a probabilidade de uma determinada solução-candidata ser escolhida para ser um dos progenitores de uma nova solução-filhote é diretamente proporcional ao seu valor de aptidão.

Um grave problema da seleção proporcional é que, na medida em que a população converge para as proximidades de uma solução final, os valores de aptidão das soluções-candidatas tendem a se equiparar, conduzindo a probabilidades semelhantes para todos os indivíduos e prejudicando a seletividade das melhores soluções. Este problema é geralmente contornado fazendo-se ajustes apropriados do valor de aptidão na medida em que a população converge.

Uma forma mais adequada de se evitar o problema da baixa seletividade nas populações finais é a utilização de mecanismos de solução que se baseiam apenas na comparação entre as aptidões de diferentes indivíduos (e não no valor da aptidão propriamente dita), tais como os mecanismos de seleção por torneio e por truncamento.

O mecanismo de solução por torneio se processa da seguinte forma:

1. São escolhidos aleatoriamente dois grupos de K soluções candidatas para cada nova solução a ser gerada;
2. Em cada grupo de soluções, a solução com maior valor de aptidão é marcada como a solução "vencedora"do torneio;
3. A nova geração de soluções candidatas é formada a partir da recombinação da seqüência genética das soluções "vencedoras"dos dois torneios.

Utiliza-se tipicamente o valor K = 2 (torneios entre pares de soluções). A intensidade da seleção pode ser aumentada utilizando-se valores maiores para K.

A seleção por truncamento se processa da seguinte forma:

1. As soluções candidatas são ordenadas em ordem decrescente de valor de aptidão, formando um rank;
2. A nova geração de soluções candidatas é formada a partir da recombinação aleatória entre as primeiras T% soluções candidatas do rank, onde T é um percentual que regula a intensidade da seleção (tipicamente 20% a 50%).

Um aumento na intensidade de seleção (maior K ou menor T) tende a acelerar a convergência do algoritmo genético, ao custo de se aumentar o risco de convergência prematura para soluções sub-ótimas de baixa qualidade.

Outros mecanismos de seleção são descritos na referência [91].

B.5 Mecanismo de Recombinação

O mecanismo de recombinação é elaborado de forma a possibilitar a construção de uma nova solução (solução-filhote) a partir dos genes de duas soluções pré-existentes.

Três mecanismos de recombinação freqüentemente utilizados são:

- Recombinação em 1 ponto;
- Recombinação em 2 pontos;
- Recombinação uniforme.

No mecanismo de recombinação em 1 ponto, escolhe-se aleatoriamente um número inteiro n entre 1 e g 1, onde g é o número de genes de cada solução-candidata. Em seguida, as seqüências de genes de cada um dos progenitores são seccionadas em 2 segmentos após o n-ésimo gene de cada progenitor. A sequência de genes da soluçãofilhote é então construída concatenando-se o primeiro segmento de um dos progenitores com o segundo segmento do outro progenitor.

No mecanismo de recombinação em 2 pontos, são escolhidos aleatoriamente dois números inteiros n1 e n2 entre 1 e g, tais que n1 n2 . Em seguida, as seqüências de genes de cada um dos progenitores são seccionadas em 3 segmentos (1 a n1, n1 1 a n2 e n2 1 a g). A

sequência de genes da solução-filhote é então constituída pelos genes de 1 a n1 e de n2 1 a g de um dos progenitores e pelos genes de n1 1 a n2 do outro progenitor.

Essencialmente, o mecanismo de recombinação em 1 ponto é o caso particular da recombinação em 2 pontos em que n1 n e n2 g. O principal objetivo da recombinação em 2 pontos é eliminar a polarização estatística existente na recombinação em 1 ponto, que decorre do fato de que um dos cortes da seqüência de genes ocorre sempre no gene g, escolhido arbitrariamente para ser o último da seqüência.

No mecanismo de recombinação uniforme, cada gene da seqüência da solução-filhote é herdado aleatóriamente de um dos progenitores, com 50% de probabilidade para cada progenitor.

Em muitos casos, determinadas características das soluções-candidatas estão associadas à correlação entre um ou mais genes. Nestes casos, se os genes correlacionados forem posicionados próximos uns aos outros, a probabilidade de tais características serem herdadas diretamente dos progenitores para o filhote torna-se maior se forem utilizados mecanismos de recombinação em 1 ou 2 pontos, em comparação com a utilização de mecanismos de recombinação uniforme.

Assim, no caso dos mecanismos de recombinação em 1 ou 2 pontos, características benéficas relacionadas à correlação de genes vizinhos têm maior probabilidade de serem preservadas nas gerações subseqüentes.

Por outro lado, no mecanismo de recombinação uniforme, o posicionamento escolhido para os genes no mecanismo de codificação não exerce influência estatística no processo. Em contrapartida à desvantagem de uma maior probabilidade de ruptura de seqüências benéficas, a recombinação uniforme tende a produzir uma maior diversidade de seqüências de genes correlacionados, aumentando assim a probabilidade de que novas características favoráveis surjam na população.

Uma estratégia interessante é utilizar recombinação uniforme nas primeiras gerações, quando as soluções-candidatas ainda são de baixa qualidade e a exploração de uma grande diversidade de seqüências genéticas é desejável, e recombinação em 2 pontos nas últimas gerações, quando maioria das soluções-candidatas são de alta qualidade e se deseja evitar que seqüências benéficas sejam rompidas freqüentemente.

Outros mecanismos de recombinação são descritos na referência [54].

B.6 Mecanismo de Mutação

Se fosse utilizado isoladamente, o mecanismo de recombinação conduziria à convergência para uma solução composta exclusivamente pelos mesmos genes que existiam na população inicial. A menos que todos os genes da solução ótima estivessem presentes na população inicial (situação improvável em problemas razoavelmente complexos), a solução encontrada seria necessáriamente sub-ótima (e possivelmente de baixa qualidade).

Por esta razão, o mecanismo de mutação é um componente essencial para o algoritmo genético, sendo o único mecanismo capaz de gerar novos genes ausentes da população inicial, propiciando, assim, uma exploração abrangente do espaço de busca.

Em um mecanismo de mutação típico, cada gene da solução-filhote é submetido à uma probabilidade de sofrer uma mutação. Esta probabilidade, denominada taxa de mutação, se situa tipicamente na faixa de 0,0001 a 0,01.

Nos casos em que os genes são representados por cadeias de dígitos binários, a mutação é geralmente efetuada alterando-se o valor de um dígito binário escolhido aleatoriamente. Nos casos de genes representados por variáveis contínuas, o valor do gene pode ser alterado de diversas maneiras, podendo receber um novo valor escolhido aleatoriamente com probabilidade uniforme dentro da faixa de variação permitida para aquele gene, ou com um padrão de probabilidade específico em torno de seu valor original [40].

Uma taxa de mutação elevada é geralmente interessante nas primeiras gerações, quando as soluções-candidatas ainda são de baixa qualidade e a exploração de uma grande diversidade genes é desejável. Por outro lado, na medida em que a população evolui e passa a conter soluções-candidatas de alta qualidade, uma taxa alta de mutação deixa de ser interessante, visto que, em uma população de soluções de alta qualidade, as mutações aleatórias passam a ter probabilidade muito maior de prejudicar o indivíduo, em vez de aprimorá-lo.

Portanto, uma estratégia interessante é diminuir gradativamente a taxa de mutação na medida em que a população evolui.

B.7 Critério de Terminação

Ao longo de sucessivas gerações de soluções-candidatas, as seqüências de genes destas tenderão a convergir para a seqüência de genes correspondente a uma solução final.

Na medida em que ocorre a convergência, o mecanismo de recombinação genética torna-se ineficaz, pois todos os potenciais progenitores de novas soluções-filhotes tendem a ter conteúdo genético muito semelhante. O mecanismo de mutação também se torna ineficaz, pois, na medida em que toda a população converge para uma solução ótima local, mutações aleatórias tendem a produzir predominantemente indivíduos menos aptos que a média da população, que tendem a ser descartados.

Assim, é conveniente se estabelecer um critério de terminação visando abortar a execução do algoritmo genético tão logo se perceba que haverá pouco ou nenhum ganho adicional na continuação do processo (a expectativa é que isto se deva ao fato de a população ter convergido para uma solução ótima).

Uma vez terminado o algoritmo, toma-se a melhor solução-candidata encontrada como solução final do problema.

Critérios de terminação freqüentemente utilizados geralmente se baseiam na ocorrência de um dentre os eventos listados a seguir, ou na ocorrência simultânea de mais de um deles:

- Quando o valor de aptidão do indivíduo mais apto da população tiver aumentado menos de X% nas últimas Y gerações;
- Quando os valores dos parâmetros codificados em todos os genes dos N indivíduos mais aptos apresentarem desvio padrão menor que X%.

Diversos critérios de terminação podem ser utilizados simultaneamente, podendo-se requerer, para a terminação do algoritmo, a satisfação simultânea de todos os critérios ou simplesmente a satisfação de qualquer um dentre os critérios utilizados.

B.8 Descrição Genérica de um Algoritmo Genético

É apresentada a seguir uma descrição genérica de um algoritmo genético:

1. Gerar aleatoriamente uma população inicial de soluções candidatas;
2. Calcular a função de adequação (fitness function) para todas as candidatas;
3. Enquanto um critério de terminação não se cumprir:

    3.1 Selecionar um grupo de soluções de acordo com o valor de adequação (fitness);

    3.2 Criar uma nova geração recombinando os genes das soluções selecionadas;

3.3 Sujeitar as soluções candidatas recém-geradas a mutação aleatória;

3.4 Calcular a função de adequação (fitness function) para todas as candidatas.

4. Tomar como solução final a candidata com maior valor de adequação (fitness).

O processo se inicia com uma população gerada aleatoriamente. Em cada iteração, uma nova geração é criada. Sobre cada nova geração atuam os mecanismos de seleção, recombinação e mutação.

Estes mecanismos trabalham cooperativamente no sentido de aumentar o valor médio de adequação de modo a conduzir a população rumo ao valor ótimo de adequação. Este processo deve ser conduzido sem abrir mão prematuramente da diversidade genética, visando evitar a convergência prematura para uma solução ótima de qualidade muito aquém da solução ótima global.

Diversos parâmetros do algoritmo, tais como a dimensão da população, a intensidade da seleção e a taxa de mutação, devem ser adequadamente ajustados para que a evolução se dê no ritmo adequado, ou seja, nem demasiadamente lenta (ineficiente) e nem demasiadamente rápida (convergência prematura).

Ao final do processo, o indivíduo com o melhor grau de adequação é tomado como a solução final do problema.

Apêndice C

Inequações Matriciais Lineares

Diversos problemas de otimização relacionados à teoria de controle envolvem desigualdades matriciais lineares, freqüentemente referenciadas pela sigla LMI, originada da expressão inglesa Linear Matrix Inequality [92].

Desigualdades matriciais lineares são restrições convexas que podem ser expressas na forma geral:

$$F_0 + y_1 F_1 + y_2 F_2 + \cdots + y_m F_m > 0 \tag{C.1}$$

onde $F_i = F_i^T \in \mathbb{R}^{n \times n}$ $i = 0, 1, \ldots, m$ são matrizes conhecidas e $y_1, y_2, \ldots, y_m \in \mathbb{R}$.

Restrições do tipo

$$M^T P + PM > 0 \tag{C.2}$$

onde $M, P \in \mathbb{R}^{n \times n}$, $P = P^T$, sendo M uma matriz constante, são LMIs, visto que se pode mapear (C.2) em (C.1) associando-se os elementos p_{ij}, $i, j = 1, 2, \ldots, n$, $i \leq j$ às incógnitas y_k, $k = 1, 2, \ldots, n(n+1)/2$, e atribuindo às matrizes F_k correspondentes a cada elemento p_{ij} o valor $M^T E_{ij} + E_{ij} M$, onde E_{ij} são matrizes cujos elementos "ij" e "ji" têm valor 1 e todos os demais elementos têm valor zero.

Problemas de otimização envolvendo restrições na forma de LMI são comumente denominados *problemas de programação semidefinida* [74].

São definidas a seguir algumas classes de problemas de programação semidefinida, utilizando a nomenclatura da referência [74].

Classe **LMIP** (*LMI Problem*)

encontrar $y \in \mathbb{R}^m$

sujeito a

$$F_0 + y_1 F_1 + y_2 F_2 + \cdots + y_m F_m > 0 \tag{C.3}$$

Classe **EVP** (*Eigenvalue Problem*)

minimizar $c^T y$

sujeito a

$$F_0 + y_1 F_1 + y_2 F_2 + \cdots + y_m F_m > 0 \tag{C.4}$$

Classe **GEVP** (*Generalized Eigenvalue Problem*)

minimizar λ

sujeito a

$$\begin{aligned} \lambda(B_0 + y_1 B_1 + \cdots + y_m B_m) - (F_0 + y_1 F_1 + \cdots + y_m F_m) > 0 \\ B_0 + y_1 B_1 + \cdots + y_m B_m > 0 \\ C_0 + y_1 C_1 + \cdots + y_m C_m > 0 \end{aligned} \tag{C.5}$$

Estas três classes de problemas de programação semidefinida podem ser solucionadas eficientemente por métodos de pontos interiores especializados.

Estes métodos têm sido intensivamente investigados e aperfeiçoados na última década. Muitos deles apresentam convergência com complexidade polinomial, em alguns casos comprovada teoricamente e, em outros, experimentalmente.

É importante esclarecer que, neste contexto, solucionar o problema significa determinar se o problema é viável ou não e, caso seja, obter uma solução viável cujo valor objetivo exceda o mínimo global por uma diferença menor que uma tolerância pré-especificada.

A resolução de problemas da classe LMIP geralmente é efetuada adicionando-se uma variável de decisão extra, θ, de modo a permitir que, para um valor suficientemente alto de θ o problema modificado tenha uma solução inicial viável conhecida. Este problema modificado é da classe EVP e geralmente tem a seguinte forma (embora diversas variantes sejam freqüentemente utilizadas):

minimizar θ

sujeito a

$$F_0 + y_1 F_1 + \cdots + y_m F_m + \theta I > 0 \tag{C.6}$$

O problema (C.6) equivale a maximizar o menor autovalor da expressão $F_0 + y_1 F_1 + \cdots + y_m F_m$.

Se o valor ótimo θ^* for não-positivo ($\theta^* \leq 0$), obtém-se a confirmação de que o problema original é viável e que $y_1^*, y_2^*, \ldots, y_m^*$ é uma solução viável deste problema, a qual maximiza o menor autovalor do resultado da expressão matricial.

Nos casos em que o único interesse é comprovar a existência de uma solução viável (não necessariamente ótima), pode-se interromper o processo de otimização tão logo seja obtida uma solução viável, ou seja, uma solução para a qual $\theta^* \leq 0$.

Se, por outro lado, o valor ótimo θ^* for positivo, isto significará que o problema não possui uma solução viável, ou seja, que não existe solução $y_1, y_2, \ldots, y_m$ capaz de evitar com que a expressão $F_0 + y_1 F_1 + \cdots + y_m F_m$ tenha pelo menos um autovalor negativo.

Para um melhor entendimento de LMIs e suas aplicações, pode-se verificar o texto de BOYD [74].

A solução de LMIs é normalmente realizada através do método dos pontos interiores de NESTEROV e NEMIROVSKY. Para um descrição completar do método, verificar [76].

Apêndice D

Fluxograma Simplificado do Algoritmo de Balanceamento de Carga

A figura D.1 apresenta um fluxograma simplificado para o algoritmo de balanceamento de carga.

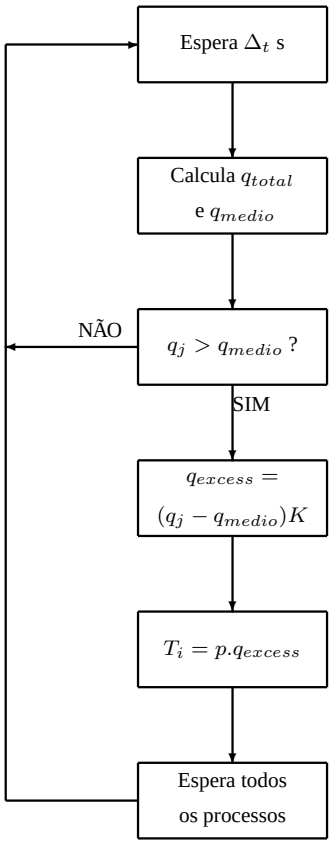

Figura D.1: Fluxograma simplificado do algoritmo de balanceamento de carga

Cada nó estima a quantidade de tarefas acima da média global. Se a estimativa do j-ésimo nó for positiva, o j-ésimo nó enviará então uma fração dada por pq_{excess} para cada nó, onde $p = 1/(n-1)$.

A implementação do algoritmo, através do programa *LoadBalancer* foi feita de forma síncrona, onde após as transferências de carga ($T_i = pq_{excess}$), a função *MPI_Barrier* estabelece um ponto de espera para todos os nós envolvidos no processo de balanceamento.

Referências

[1] RIVELLO, M., SILVA, R., AMORIM, R., "Curso de montagem de clusters linux". *Laboratório Nacional de Computação Científica* v. 1, n. 1, pp. 1–50, 2003.

[2] CASADO, J. M., "Workload balance approaches for branch and bound algorithms on distributed systems". *Proceedings of the 7th. Euromicro Workshop on Parallel and Distributed Processing* v. 1, pp. 389–400, February 1999.

[3] ROTITHOR, H. G., "Taxonomy of dynamic task scheduling schemes in distributed computing systems". In: *IEE Proc. in Comput. Digit. Tech.*, v. 141, pp. 1–10, January 1994.

[4] CORRADI, A., LEONARDI, L., ZAMBONELLI, F., "Diffusive load-balancing polices for dynamic applications". *IEEE Concurrency* v. 22, n. 1, pp. 979–993, Jan.-Fev. 1999.

[5] WILLEBEEK-LEMAIR, M. H., REEVES, A. P., "Strategies for dynamic load balancing on highly parallel computers". *IEEE Transactions on Parallel and Distributed Systems* v. 4, n. 9, pp. 979–993, 1993.

[6] PARK, B. J., CHOI, H. R., KIM, H. S., "A hybrid genetic algorithm for the job shop scheduling problems". *Computers & Industrial Engineering* v. 45, pp. 597–613, 2003.

[7] KASAHARA, H., NARITA, S., "Practical multiprocessor scheduling algorithm for efficient parallel processing". *IEEE Transactions on Computation* v. 33, pp. 1023–029, 1979.

[8] BOKHARI, S. H., "Dual processor scheduling with dynamic reassignment". *IEEE Transactions on Software Engineering* v. 1, July 1979.

[9] YI, P., MA, R., LEE, E., TSUCHIYA, M., "A task allocation model for distributed computing systems". *IEEE Transactions on Computers* v. 1, 1982.

[10] SHEN, C., TSAI, W., "A graph matching approach to optimal task assignment in distributed systems using a minimax criterion". *IEEE Transactions on Computers* v. 1, 1985.

[11] SEREDYNSKI, F., "Distributed scheduling using simple learning machines". *European Journal of Operational Research* v. 107, pp. 401–413, 1998.

[12] PRICE, C. C., SALAMA, M. A., "Scheduling of precedence-constrained tasks on multiprocessors". *Computer Journal* v. 3, n. 33, , 1990.

[13] SEREDYNSKI, F., "Dynamic mapping and load balancing with parallel genetic algorithms". *Proceedings of the 1st. IEEE Conference on Evolutionary Computation* v. 2, pp. 834–839, June 1994.

[14] DHODHI, M. K., AHMAD, I., AHMAD, I., "A multiprocessor scheduling scheme using problem-space genetic algorithms". *IEEE International Conference on Evolutionary Computation* v. 1, November 1995.

[15] AHMAD, I., DHODHI, M. K., "Multiprocessor scheduling in a genetic paradigm". *Parallel Processing* v. 22, pp. 395–406, 1996.

[16] SEREDYNSKI, F., "Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems". *Journal of Parallel and Distributed Computing* v. 47, pp. 39–57, 1997.

[17] SHUM, K. H., "Effective fault tolerance for agent-based cluster computing". *The Journal of Systems and Software* v. 48, pp. 189–196, 1998.

[18] KRAUS, S., PLOTKIN, T., "Algorithms of distributed task allocation for cooperative agents". *Theoretical Computer Sciences* v. 242, pp. 1–27, 2000.

[19] CAO, J., SPOONER, D. P., JARVIS, S. A., *ET AL.*, "Agent-based grid load balancing using performance-driven task scheduling". *Proceedings of the International Parallel and Distributed Processing Symposium* v. 1, 2003.

[20] MONG, K., SUN, W. H., "Ant colony optimization for routing and load balancing - survey and new directions". *IEEE Transactions On Systems, Man and Cybernetics - Part A: Systems and Humans* v. 33, n. 5, , September 2003.

[21] RIECHMANN, T., "Genetic algorithm learning and evolutionary games". *Journal of Economic Dynamics & Control* v. 25, pp. 1019–1037, 2001.

[22] ALTMAN, E., KAMEDA, H., HOSOKAWA, Y., "Nash equilibria in load balancing in distributed computer systems". *International Game Theory Review* v. 4, n. 2, pp. 1–10, July 2002.

[23] GROSU, D., CHRONOPOULOS, A. T., LEUNG, M.-Y., "Load balancing in distributed systems: An approach using cooperative games". *Proceedings of the International Parallel and Distributed Processing Symposium* v. 1, 2002.

[24] ZHU, W., LIANG, T.-Y., SHIEH, C.-K., "A hopfield neural network based task mapping method". *Computer Communications* v. 22, pp. 1068–1079, 1999.

[25] CHEN, W., LIN, C., "A hybrid heuristic to solve a task allocation problem". *Computers & Operational Research* v. 27, pp. 287–303, 2000.

[26] ZHOU, D. N., CHERKASSKY, V., BALDWIN, T. R., OLSON, D. E., "A neural network approach to job-shop scheduling". *IEEE Transactions on Neural Networks* v. 2, n. 1, pp. 175–179, January 1991.

[27] SABUNCUOGLU, I., GURGUN, B., "A neural network model for scheduling problems". *European Journal of Operational Research* v. 93, pp. 288–299, 1996.

[28] SILVA, M. P., CARDEIRA, C., MAMMERI, Z., "Solving real-time scheduling problems with hopfield-type neural networks". *New Frontiers of Information Technology - Proceedings of the 23rd EUROMICRO Conference* v. 1, pp. 671–678, 1997.

[29] CHEN, R.-M., HUANG, Y.-M., "Competitive neural network to solve scheduling problems". *Neurocomputing* v. 37, pp. 177–196, 2001.

[30] GHANEM, J., ABDALLAH, C. T., HAYAT, M., *ET AL.*, "Implementation of the load balancing algorithm over a local area network and the internet". *43rd. IEEE Conference on Decision and Control* v. 1, pp. 4199–4204, December 2004.

[31] YAÏCHE, H., MAZUMDAR, R. R., ROSENBERG, C., "A game theoretic framework for bandwidth allocation and princing in broadband networks". *IEE/ACM Transactions on Networking* v. 8, n. 5, pp. 667–678, October 2000.

[32] DHAKAL, S., PASKALEVA, B. S., HAYAT, M. M., *ET AL.*, "Dynamical discrete-time load balancing in distributed systems in the presence of time delays". *Proceedings of the 42nd. IEEE Conference on Decision and Control* v. 1, 2003.

[33] CHIASSON, J., TANG, Z., GHANEM, J., *ET AL.*, "The effect of time delays on the stability of load balancing algorithms for parallel computations". *IEEE Transactions on Control Systems Technology* v. 13, n. 6, pp. 932–942, November 2005.

[34] EL-ABD, A. E., EL-BENDARY, M. I., "A neural network approach for dynamic load balancing in homogeneous distributed systems". *Neurocomputing* v. 2, pp. 628–629, 1997.

[35] CARNEIRO, M., "A 3,6 ghz chegarás. daí não passarás". *Revista Veja - Editora Abril* v. 1, n. 1878, , 3 de novembro 2004.

[36] GIBBS, W. W., "Racha no núcleo". *Revista Scientific American Brasil* v. 3, n. 31, pp. 20–23, dezembro 2004. Editora Duetto.

[37] CRIVELLI, S., HEAD-GORDON, T., "A new load balancing strategy for the solution of dynamical large-tree-search problems using a hierarchical approach". *Journal of Research & Development - IBM* v. 48, n. 2, , 2004.

[38] MARTINO, V. D., MILILOTTI, M., "Suboptimal scheduling in a grid using genetic algorithms". *Parallel Computing* v. 30, pp. 553–565, 2004.

[39] DUSSA-ZIEGER, K., SCHWEHM, M., "Scheduling of parallel programs on configurable multiprocessors by genetic algorithms". *International Journal of Approximate Reasoning* v. 19, pp. 23–38, 1998.

[40] HUI, C.-C., CHANSON, S. T., "Hydrodynamic load balancing". *IEEE Transactions on Parallel and Distributed Systems* v. 10, n. 11, pp. 1118–1137, November 1999.

[41] KANG, W., KELLY, F. P., LEE, N. H., WILLIAMS, R. J., "Fluid and brownian approximations for an internet congestion control model". *43rd. IEEE Conference on Decision and Control* v. 1, pp. 3938–3943, December 2004.

[42] HUI, C.-C., CHANSON, S. T., "A hydrodynamic approach to heterogeneous dynamic load-balancing in a network of computers". *International Conference on Parallel Processing* v. 1, pp. III–140 – III–147, 1996.

[43] HUI, C.-C., CHANSON, S. T., "Theoretical analysis of the heterogeneous dynamic load-balancing problem using a hydrodynamic approach". *Journal of Parallel and Distributed Computing* v. 43, pp. 139–146, 1997.

[44] DHAKAL, S., *Load Balancing in Delay-Limited Distributed Systems*. Master's Thesis, The University of New Mexico, December 2003.

[45] GHANEM, J., *Implementation of Load Balance Policies in Distributed Systems*. Master's Thesis, University of New Mexico, Albuquerque - New Mexico, June 2004.

[46] TANG, Z., BIRDWELL, J. D., CHIASSON, J., ABDALLAH, C. T., HAYAT, M. M., "A time delay model for load balancing with processor resource constraints". In: *43rd. IEEE Conference on Decision and Control*, pp. 4193–4198. IEEE CDC, December 2004.

[47] LOH, P. K. K., HSU, W. J., WENTONG, C., SRISKANTHAN, N., "How network topology affects dynamic load balancing". *IEEE Concurrency* v. 4, n. 3, pp. 25–35, 1996.

[48] EM KARMIADAKIS, G., II, R. M. K., *Parallel Scientific Computing in C++ and MPI: A Seamless Approach to Parallel Algorithms and Their Implementations*. Cambridge University Press, 2003.

[49] HOPFIELD, J. J., "Neuronal networks and physical systems with emergent collective computation abilities". *Proceedings of the National Academy of Sciences* v. 79, pp. 2554–2558, 1982.

[50] HOPFIELD, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons". *Proceedings of the National Academy of Sciences* v. 81, pp. 3088–3092, 1984.

[51] TANK, D. W., HOPFIELD, J. J., "Simple neural optimization networks: An a/d converter, signal decision circuit, and a linear programming network". *IEEE Transactions on Circuits and Systems* v. 33, pp. 533–541, 1986.

[52] KASZKUREWICZ, E., BHAYA, A., *Matrix Diagonal Stability in Systems and Computation*. Birkhäuser, 2000.

[53] FORTI, M., GRAZZINI, M., NISTRI, P., PANCIONI, L., "A result on global convergence in finite time for nonsmooth neural networks". *Proceedings of the IEEE International Symposium on Circuit and Systems* v. 1, pp. 759–762, May 2006.

[54] SINGH, V., "A generalized lmi-based approach to the global asymptotic stability of delayed cellular neural networks". *IEEE Transactions on Neural Networks* v. 15, n. 1, pp. 223–225, January 2004.

[55] PERSIDSKII, S. K., "Problem of abosulte stability". *Automation and Remote Control* v. 12, pp. 1889–1895, 1969.

[56] CHEN, C.-T., *Linear System - Theory and Design*. 3rd ed., Oxford University Press, 1999.

[57] KHARITONOV, V. L., "Robust stability analisys of time delay systems: A survey". *Annual Reviews in Control* v. 23, pp. 185–196, 1999.

[58] GARCIA, H., KHARITONOV, V. L., "Lyapunov matrices for time delay systems". *2nd. International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005)* v. 1, September 2005.

[59] KOKAME, H., MORI, T., "A tutorial on stability and stabilization of delay differential systems". *SICE - Annual Conference in Fukui - Fukui University - Japan* v. 1, pp. 1785–1791, August 2003.

[60] NIAN, X., "Stability of linear systems with time-varying delays: An lyapunov functional approach". *Proceedings of the American Control Conference - Denver - Colorado* v. 1, pp. 883–884, June 2003.

[61] SLOTINE, J.-J. E., LI, W., *Applied Nonlinear Control*. Prentice Hall, 1991.

[62] STRANG, G., *Linear Algebra and Its Applications*. 3rd ed., Thomson Learning Inc., 1988.

[63] FORTI, M., GRAZZINI, M., NISTRI, P., PANCIONI, L., "Generalized lyapunov approach for convergence of neural networks with discontinuous or non-lipschitz activations". *Physica D* v. 214, pp. 88–99, 2006.

[64] FORTI, M., NISTRI, P., "Global convergence of neural networks with discontinuous neuron activations". *IEEE Transactions on Circuit and Systems - I* v. 50, n. 11, pp. 1421–1435, November 2003.

[65] CHIASSON, J., "A method for computing the inetrval of delay values for which a differential-delay system is stable". *IEEE Transactions on Automatic Control* v. 33, n. 12, pp. 1176–1178, December 1988.

[66] YU, L., CHU, J., "An lmi approach to guaranteed cost control of linear uncertain time-delay systems". *Automatica* v. 35, pp. 1155–1159, 1999.

[67] GOUAISBAUT, F., DAMBRINE, M., RICHARD, J. P., "Robust control of delay systems: A sliding mode control design via lmi". *Systems & Control Letters* v. 46, pp. 219–230, 2002.

[68] KWON, W. H., LEE, Y. S., HAN, S. H., "General receding horizon control for linear time-delay systems". *Automatica* v. 40, pp. 1603–1611, 2004.

[69] PHAT, V. N., NIAMSUP, P., "Stability of linear time-varying delay systems and applications to control problems". *Journal of Computational and Applied Mathematics* v. 194, pp. 343–256, 2006.

[70] SINGH, V., "Global robust stability of delayed neural networks: An lmi approach". *IEEE Transactoins on Circuitis and Systems-II* v. 52, n. 1, pp. 33–36, janeiro 2005.

[71] LIAO, X., WING, K., Z. WU, G. C., "Novel robust stability criteria for interval-delayed hopfield neural networks". *IEEE Transactions Circuits System I - Fundamental os Theory Application* v. 48, n. 11, pp. 1355–1359, November 2001.

[72] XU, B., LIU, X., LIAO, X., "Global asymptotic stability of high-order hopfield type neural networks with time delay". *Computers and Mathematics with Applications* v. 45, pp. 1729–1737, 2003.

[73] ZHANG, H., LI, C., LIAO, X., "A note on the robust stability of neural networks with time delay". *Chaos, Solitons and Fractals* v. 25, pp. 357–360, 2005.

[74] BOYD, S., GHAOUI, L. E., FERRON, E., *ET AL.*, *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1994.

[75] GAHINET, P., NEMIROVSKI, A., LAUB, A. J., CHILALI, M., "*LMI Control Toolbox For Use with Matlab - User's Guide - version 1*". The Mathworks, www.mathworks.com, 1995.

[76] NESTEROV, Y., NEMIROVSKY, A., *A general approach to polynomial-time algorithms design for convex programming*. Centr. Econ. and Math. Inst., , USSR Acad. Sci., Moscow, 1988.

[77] DORF, R. C., BISHOP, R. H., *Sistemas de Controle Modernos*. 8a ed., LTC - Livros Técnicos e Científicos Editora S.A., 1998.

[78] MUKAIDANI, H., "An lmi approach to decentralized guaranteed cost control for a class of uncertain nonlinear large-scale delay systems". *Journal of Mathematical Analysis and Applications* v. 300, pp. 17–29, october 2004.

[79] LIEN, C.-H., "Delay-dependent and delay-independent guaranteed cost control for uncertain neutral systems with time-varying delays via lmi approach". *Chaos, Solitons and Fractals* v. 1, 2007.

[80] KOSMIDOU, O. I., BOUTALIS, Y. S., "A linear matrix inequality approach for guaranteed cost control of systems with state and input delays". *IEEE Transactions on Sytems, Man and Cybernetics - Part A: Systems and Humans* v. 5, n. 36, pp. 936–942, September 2006.

[81] KWON, O. M., PARK, J. H., "Decentralized guaranteed cost control for uncertain large-scale systems using delayed feedback: Lmi optimization approach". *Journal of Optimization Theory and Applications* v. 129, n. 3, pp. 391–414, june 2006.

[82] RAMI, M. A., ZHOU, X. Y., "Linear matrix inequalities, riccati equations and indefinite stochastic linear quadratic controls". *IEEE Transactions on Automatic Control* v. 6, n. 45, pp. 1131–1143, june 2000.

[83] TEWARI, A., *Modern Control Design with Matlab and Simulink*. John Wiley & Sons, Ltd., 2002.

[84] ENNE, A. J. F., *Frame Relay - Redes, Protocolos e Serviços*. Axel Books, 1998.

[85] KARYPIS, G., GUPTA, A., KUMAR, V., "A parallel formulation of interior point algorithms". In: *Conference on High Performance Networking and Computing*, v. 1, pp. 204–213. ACM New York, NY, USA, 1994.

[86] LANGSAM, Y., AUGENSTEIN, M. J., TENEMBAUM, A. M., *Data Structures Using C and C++*. 2nd ed., Prentice Hall, 1996.

[87] RIBEIRO, U., *Sistemas Distribuídos - Desenvolvendo Aplicações de Alta Performance no Linux*. Axcel Books do Brasil Editora, 2005.

[88] COMER, D. E., STEVENS, D. L., *Interligação em Rede com TCP/IP - Projeto, Implementação e Detalhes Internos*, v. 1. 3a ed., Editora Campus, 1998.

[89] FILHO, S. L. M., *Uma Abordagem Multithread em Aplicações Paralelas Utilizando MPI*. Master's Thesis, Programa de Pós-graduação em Informática - Universidade Federal do Paraná, 2005.

[90] ZHANG, Y., HAKOZAKI, K., KAMEDA, H., SHIMIZU, K., "A performance comparison of adaptive and static load balancing in heterogeneous distributed systems". *Proceedings of the 28th. Annual Simulation Symposium* v. 1, pp. 332–340, April 1995.

[91] HOLLAND, J. H., *Adaptation in Natural and Artifical Systems*. University of Michigan Press, Ann Arbor, 1975.

[92] DE ALMEIDA, H. L. S., *Otimização de Controladores Robustos Utilizando Funções de Lyapunov Quadráticas-Por-Partes*. Ph.D. dissertation, COPPE/UFRJ, Fevereiro 2003.