UTILIZAÇÃO DE SISTEMAS GRADIENTES PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM COMPUTADORES PARALELOS

Leonardo Valente Ferreira

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. Liu Hsu, Dr. État

Prof. Marcelo Carvalho Minhoto Teixeira, D.Sc

Prof. Paulo Augusto Valente Ferreira, D.Sc

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2006

FERREIRA, LEONARDO VALENTE

Utilização de Sistemas Gradientes para Resolução de Problemas de Otimização em Computadores Paralelos [Rio de Janeiro] 2006

VIII, 201p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2006)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. sistemas gradientes
2. processamento paralelo
3. otimização
4. support vector machines
5. restauração de imagens
6. sistemas de estrutura variável

I. COPPE/UFRJ II. Título (série)

Agradecimentos

À minha família, em especial ao meu pai, que sempre foi uma fonte de inspiração e referência em minha vida, por todo o apoio, amor e amizade. À minha mãe, que sempre incentivou com grande entusiasmo os meus projetos de vida.

Às minhas irmãs Nancy e Elaine e às minhas tias Elvira e Rosa, pelo incentivo e apoio.

Aos meus avós, Hilda e Waldemar, que sempre tiveram participação decisiva na minha formação, e onde quer que estejam, me inspiram e apóiam.

Aos meus amigos, em especial os do NACAD, que sempre foram fonte de incentivo e apoio nos últimos seis anos.

Aos membros da banca examinadora, pelas valiosas sugestões e críticas construtivas, que foram decisivas para melhorar a qualidade deste trabalho.

Ao prof. Amit Bhaya, pelo grande apoio, entusiasmo, pela confiança, amizade e pelas horas de trabalho e dedicação.

Ao meu orientador, prof. Eugenius Kaszkurewicz, pelo inestimável apoio, entusiasmo, amizade e pelas longas horas de trabalho e dedicação ao longo dos últimos seis anos, que foram determinantes para o sucesso deste trabalho.

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

UTILIZAÇÃO DE SISTEMAS GRADIENTES PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO EM COMPUTADORES PARALELOS

Leonardo Valente Ferreira

Agosto/2006

Orientador: Eugenius Kaszkurewicz

Programa: Engenharia Elétrica

Sistemas gradientes que resolvem problemas de otimização são propostos e analisados. Estes sistemas são obtidos a partir de um método de penalização exata, resultando em sistemas dinâmicos com segundo membro descontínuo, que podem ser interpretados como modelos de redes neurais com função de ativação descontínua. As análises de convergência, feitas através da forma Persidskii dos sistemas gradientes, utilizando funções de Lyapunov não suaves do tipo Lure-Persidskii, são simples e resultam em condições de convergência global tratáveis.

Os sistemas propostos, por serem facilmente paralelizáveis, são adequados à resolução de problemas de otimização grande porte, utilizando computadores paralelos. Três aplicações são consideradas: o problema *k*-winners-take-all, formulado através de um problema de programação linear, treinamento de *support vector machines* (SVMs) e reconstrução de imagens. O treinamento de SVMs, com grandes bases de dados, e o problema de reconstrução de imagens são de grande porte, exigindo a paralelização dos sistemas gradientes.

As soluções são obtidas através de integração numérica em paralelo dos sistemas gradientes, utilizando algoritmos de integração com passo adaptativo. Os resultados mostram que sistemas gradientes são métodos competitivos para a resolução de problemas de otimização de grande porte, quando implementados em paralelo e/ou resolvidos numericamente através técnicas de integração eficientes.

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

ON USING GRADIENT SYSTEMS FOR SOLVING OPTIMIZATION PROBLEMS IN PARALLEL COMPUTERS

Leonardo Valente Ferreira

August/2006

Advisor: Eugenius Kaszkurewicz

Department: Electrical Engineering

Gradient systems designed to solve optimization problems are proposed and analysed. These systems are obtained by means of an exact penalty method, resulting in systems of ordinary differential equations with discontinuous righthand sides, which are neural network models with discontinuous activation functions. The global convergence of the proposed systems are proved by means of a Persidskii form of the gradient systems, and non-smooth Lure-Persidskii or diagonal type Lyapunov functions.

The proposed gradient systems can be easily parallelized, which makes them suitable to solve large scale problems. Three applications are considered: the *k*-winners-take-all problem, formulated by means of a linear programming problem, training of support vector machines (SVMs) and image restoration. SVM training with large data sets and image restoration are usually large scale problems, and the gradient systems that solve those problems need to be parallelized.

The gradient systems are solved by means of parallel numerical integration of these systems, and to use efficient to use efficient numerical integration methods with adaptive stepsize control, the nonlinearities are smoothed in a neighborhood of the corresponding discontinuity surface by means of the *boundary layer technique*. It is shown that gradient systems solve efficienty large scale optimization problems, when are implemented in parallel computers and/or solved by means of efficient numerical integration techniques.

Sumário

Notação

Vetores são representados por caracteres minúsculos em negrito, como $\mathbf{b}$ ou por letras gregas minúsculas em negrito, como $\boldsymbol{\beta}$.

Matrizes são representadas por caracteres maiúsculos também em negrito, como $\mathbf{A}$ ou por letras gregas maiúsculas em negrito, como $\boldsymbol{\Theta}$.

Funções escalares de várias variáveis, como $f(x_1, \ldots, x_n)$, são representadas por $f(\mathbf{x})$ ou $f(\cdot)$, com $\mathbf{x} = (x_1, \ldots, x_n)$.

Funções vetoriais do tipo diagonal, como $(f_1(x_1), \ldots, f_n(x_n))$, são representadas por $\mathbf{f}(\mathbf{x})$.

Escalares são representados por caracteres minúsculos em itálico, como por exemplo a ou por letras gregas minúsculas, como ξ.

Desigualdades vetoriais. Dados dois vetores $\mathbf{u}$ e $\mathbf{v}$ em $\mathbb{R}^n$, a desigualdade $\mathbf{u} \geq \mathbf{v}$ significa que $u_i \geq v_i$, para todo i. De modo análogo define-se as desigualdades $\mathbf{u} \leq \mathbf{v}$, $\mathbf{u} > \mathbf{v}$ e $\mathbf{u} < \mathbf{v}$.

Normas vetoriais e matriciais. Dado um vetor $\mathbf{x}$ em $\mathbb{R}^n$, denotamos por $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$ e $\|\mathbf{x}\|_\infty$ as normas L_1, L_2 ou euclideana e L_∞ ou infinito de $\mathbf{x}$ em $\mathbb{R}^n$, respectivamente. Denotamos pot $\|\mathbf{x}\|$ a norma euclideana de $\mathbf{x}$ em $\mathbb{R}^n$. As normas matriciais são denotadas de forma análoga.

Capítulo 1

Introdução

A utilização de circuitos analógicos, formulados matematicamente através de sistemas gradientes, para a resolução de problemas de otimização foi primeiramente abordada por Pyne [1], que apresentou um método para resolução de problemas de programação linear utilizando computadores analógicos. O trabalho de Pyne foi motivado pela necessidade de resolver problemas de otimização em tempo real, e é viável porque a implementação de sistemas gradientes sob a forma de circuitos é simples e barata.

Nos anos que seguiram, houve considerável desenvolvimento na aplicação desta classe de sistemas à resolução de problemas mais gerais de otimização. Rybashov [2] propôs sistemas gradientes para resolução de problemas de programação linear e quadrática através de computadores analógicos. A resolução de problemas gerais de otimização convexa através de sistemas gradientes foi estudada em [3], apresentando uma rigorosa análise de convergência utilizando o método direto de Lyapunov. Em [4], a estabilidade de sistemas gradientes gerais para a minimização de funções suaves foi investigada.

A conexão entre o método proposto por Pyne [1] e o método da função de penalidade foi estabelecida por Karpinskaya [5], utilizando uma função de penalidade exata [6]. O uso de funções de penalidade não-suaves foi discutido por Utkin [7] e Korovin e Utkin [8], que obtiveram condições de existência destas funções de penalidade, e a convergência dos sistemas foi analisada utilizando a teoria de sistemas de estrutura variável.

No contexto de redes neurais, no ano de 1984, Hopfield [9] introduziu a rede que hoje leva seu nome, direcionada à implementação de sistemas contínuos através de circuitos. Dois anos depois, Tank e Hopfield [10] mostraram como a rede de Hopfield pode ser utilizada

para resolver de problemas de otimização, e uma implementação da rede através de um circuito, foi apresentada para resolver problemas de programação linear. A utilização de funções de ativação descontínuas na rede de Hopfield foi considerada por Forti e Nistri [11], que obtiveram condições de convergência em tempo finito através de uma rigorosa análise de convergência. O modelo de Hopfield é muito popular e diversas modificações foram propostas [12].

Nos anos 90, Rodríguez-Vázquez et al. [13] apresentaram métodos para resolução de problemas de otimização não-linear por meio de circuitos integrados SC (switched capacitor), utilizando funções de penalidade não-suaves. Utkin [7] apresentou uma análise de sistemas gradientes descontínuos aplicados à resolução de problemas gerais de programação convexa, apresentando uma análise de convergência dos sistemas apresentados. O uso de sistemas gradientes com segundo membro descontínuo para resolver problemas de otimização convexa também foi considerado por Glazos et al. [14], que utilizaram a teoria de sistemas de estrutura variável para analisar a convergência de um sistema gradiente para a resolução de problemas de programação convexa.

Uma classe de sistemas gradientes com segundo membro descontínuo que resolve problemas de programação linear foi analisada por Chong et al. [15]. O método da função de penalidade foi utilizado com duas funções de penalidade exatas e condições convergência explícitas, dependentes dos estados do sistema, foram obtidas.

Kennedy e Chua [16] analisaram a estabilidade de uma rede neural para resolução de problemas de programação não-linear, assumindo que a função de restrição e a função objetivo do problema são de classe C^2. Estas hipóteses foram relaxadas por Forti et al. [17], que consideraram a função objetivo e a função de restrição do problema como regulares, permitindo que estas funções sejam descontínuas; para esta classe de problemas foram obtidas condições de convergência global em tempo finito, dependentes da norma euclideana do vetor de estados da rede. Recentemente, a convergência de uma classe de redes neurais com funções de ativação descontínuas ou não-Lipschitz foi analisada [18], utilizando funções de Lyapunov não diferenciáveis, obtendo condições de convergência exponencial ou em tempo finito.

Análise de convergência através de sistemas Persidskii e funções tipo diagonal

Verificou-se que é possível escrever os sistemas gradientes que modelam as redes estudadas na forma Persidskii. O uso da formulação de Persidskii é justificado pelo fato de que as funções de Lyapunov associadas a estes sistemas serem bastante conhecidas. Persidskii [19] estudou a estabilidade absoluta do modelo que leva o seu nome através de uma função de Lyapunov do tipo diagonal.

Este método de análise foi utilizado em [20] e [21], em que sistemas gradientes que resolvem três variantes de problemas de programação linear foram analisados, utilizando sistemas Persidskii com segundo membro descontínuo e funções de Lyapunov do tipo diagonal não-suaves. A estratégia de análise é baseada na estratégia adotada em [15], resultando em condições de convergência baseadas em parâmetros de penalidade constantes, e portanto mais tratáveis que as obtidas por Chong et al. [15] e Forti et al. [17], que dependem dos estados do sistema dinâmico.

Sistemas Persidskii e funções de Lyapunov tipo diagonal também foram utilizados na análise de convergência de um sistema gradiente para resolver sistemas subdeterminados de equações lineares da forma $\mathbf{Ax} = \mathbf{b}$ [22]. Através desta análise, foi possível mostrar a convergência em tempo finito das trajetórias do sistema gradiente para o conjunto solução do sistema de equações lineares.

Método de penalidades

O desenvolvimento de métodos alternativos para a resolução de problemas de otimização, ganha grande impulso quando os métodos tradicionais não são adequados à resolução de problemas de grande porte. Um exemplo disto são métodos como os quase-Newton, que estão entre os mais eficientes, entretanto são inadequados à implementação sob a forma de circuitos VLSI, além de serem de difícil paralelização, pois estes métodos exigem a estimação de uma matriz hessiana em cada iteração.

Por esta razão, tornam-se necessárias abordagens alternativas, que contemplem tanto a implementação via hardware como a implementação numérica em computadores paralelos, e para este fim os sistemas tipo gradiente, associados ao método da função de penalidade constituem uma poderosa ferramenta.

O método da função de penalidade exata é particularmente adequado, pois as derivadas

destas funções são facilmente obtidas analiticamente e suas expressões são simples, tornando os sistemas gradientes também simples para fins de implementação, além de não precisarem de uma condição inicial no interior do conjunto viável do problema. Outra vantagem da utilização deste tipo de função de penalidade é que a implementação através de circuitos utilizando tecnologia VLSI pode ser feita utilizando-se apenas resistores, capacitores, amplificadores e relés, o que torna este tipo de implementação simples e barata [12]. Outros métodos podem ser utilizados para a formação do sistema gradiente, como a minimização da função lagrangeana. No entanto, este método exige a determinação dos multiplicadores de Lagrange, o que provoca um aumento considerável na dimensão do problema. Os métodos de barreiras, por outro lado, precisam ser iniciados em um ponto no interior da região viável, e funcionam evitando que as soluções saiam deste conjunto. A dificuldade em obter-se um ponto inicial no interior do conjunto viável, especialmente para problemas de dimensão elevada, é uma desvantagem do método de barreiras, dificultando o seu uso para resolver problemas de grande porte.

Além da adequabilidade dos sistemas gradiente a aplicações em tempo real utilizando circuitos, sistemas gradientes podem ser interpretados como modelos de redes neurais recorrentes [12, 23]. Estes sistemas podem ser paralelizados, sendo adequados à resolução de problemas de grande porte, através da implementação destes em computadores paralelos.

Aplicações

Um circuito k-winners-take-all (KWTA), modelado a partir de um problema de programação linear com restrições em caixa, foi analisado por Ferreira et al. [24] através dos mesmos métodos de Ferreira et al. [21]. Estes mesmos métodos foram aplicados ao treinamento de *support vector machines* (SVMs) [25]. Neste caso as condições de convergência são independentes dos parâmetros de penalidade e os métodos podem ser estendidos a problemas gerais de programação quadrática com restrições lineares. As contribuições teóricas obtidas pela utilização da forma Persidskii dos sistemas gradientes são: i) os resultados de convergência obtidos fornecem uma forma explícita para ajustar os parâmetros de penalidade apenas em função dos dados do problema de programação linear; ii) para problemas de programação quadrática, a convergência é garantida para quaisquer valores dos parâmetros de penalidade.

Aplicações como reconstrução de imagens e treinamento de SVMs utilizando um espaço de amostras muito grandes são aplicações que exigem muito poder computacional. Estes dois problemas são modelados através de problemas de otimização, que podem ser resolvidos pelos sistemas propostos por Ferreira et al. [22, 25, 26] e, como requerem grande capacidade de processamento, é necessária a paralelização dos sistemas gradientes. Os principais desafios a serem enfrentados no desenvolvimento deste trabalho envolvem: i) desenvolver programas paralelos para resolver o sistemas de equações diferenciais com o segundo membro descontínuo; ii) desenvolver algoritmos eficientes e precisos de integração paralela de sistemas de equações diferenciais com segundo membro descontínuo.

Experimentos numéricos e ambiente computacional

Os experimentos computacionais foram realizados na estrutura disponível no Núcleo de Atendimento em Computação de Alto Desempenho (NACAD), da COPPE, que conta de um cluster Itautec com 16 nós duais, cuja configuração é mostrada na tabela 1.1, e uma maquina paralela SGI Altix, com 12 processadores, utilizando arquitetura de memória compartilhada e um cluster Itanium com 4 processadores, viabilizando a implementação de aplicações paralelas de grande porte. As ferramentas de software utilizadas são de código livre e estão disponíveis gratuitamente na Internet.

Tabela 1.1: Características gerais do Cluster Itautec. Fonte: http://www.nacad.ufrj.br

Cluster InfoServer Itautec	
Número de nós	16 nós duais (2 processadores por nó)
Memória RAM	512 MB / nó
Memória Cache	256 KB / CPU
Hard Disk	18 GB
Processadores	Pentium III 1GHZ
Sist. Operacional	Red Hat Linux 7.3
Compiladores	C / C++, Fortran 77/90

A principal plataforma de desenvolvimento adotada neste trabalho é o SGI Altix. Esta máquina, dispõe de software de desenvolvimento de aplicações paralelas de código livre (compiladores e as bibliotecas MPI), e portanto é mais acessível à comunidade científica. As características gerais desta máquina são exibidas na tabela 1.2. A linguagem de programação utilizada é o FORTRAN 90 utilizando o MPI (Message Passing Interface) como APIs de programação paralela.

Tabela 1.2: Características gerais do SGI Altix. Fonte: http://www.nacad.ufrj.br

SGI Altix 350	
Número de processadores	12 procs
Memória RAM	28 GB (compartilhada)
Disco Rígido	360 GB
Sistema Operacional	Linux
Compiladores	Fortran 90 e C/C++
Arquitetura CPU Pico de performance	Itanium 2 de 64 bits 6 GFLOP/S por CPU

As características gerais do SGI Altix são mostradas na tabela 1.2. Esta máquina possui as mesmas ferramentas de desenvolvimento de aplicações paralelas que o cluster InforServer, porém é uma máquina mais poderosa, possui maior quantidade de memória, e por isso é mais adequada aos experimentos realizados neste trabalho.

Este trabalho é estruturado como segue. No capítulo 2 são apresentadas as ferramentas matemáticas e conceitos utilizados na análise teórica dos sistemas propostos. Em seguida, no capítulo 3 sistemas gradientes são introduzidos, e sua aplicação à resolução de problemas de otimização é apresentada. Os três capítulos seguintes são dedicados a três aplicações: no capítulo 4 um sistema gradiente obtido a partir de um problema de programação linear com variáveis em caixa é aplicado ao problema de encontrar os k maiores componentes de um dado vetor; o problema de separação de classes através de SVM e LS-SVM é discutido no capítulo 5; no capítulo 6 é apresentado o problema de restauração de imagens. Finalmente, no capítulo 7, são apresentadas as conclusões e perspectivas de trabalhos futuros. O apêndice A contém uma breve introdução aos conceitos básicos de computação paralela usados neste trabalho, e no apêndice B, listamos as ferramentas de software utilizadas no desenvolvimento das implementações dos algoritmos propostos.

Capítulo 2

Ferramentas matemáticas

2.1 Caracterização dos mínimos de funções

Seja $E : \mathbb{R}^n \to \mathbb{R}$ uma função escalar. Nesta seção, são apresentados resultados que permitem caracterizar os mínimos de uma função E através de suas derivadas parciais de primeira e segunda ordens.

Definição 2.1. Se uma função escalar E possui derivadas parciais contínuas até a n-ésima ordem em $\mathbb{R}^n$, diz-se que E é uma função de classe C^n.

Definição 2.2 (Ponto crítico). Um ponto $\mathbf{x}^* \in \mathbb{R}^n$ é um ponto crítico de E se

$$\nabla E(\mathbf{x}^*) = \mathbf{0},$$

em que $\nabla E(\mathbf{x}^*)$ é o gradiente de E em $\mathbf{x}^*$, e $\mathbf{0}$ é o vetor nulo em $\mathbb{R}^n$.

Definição 2.3 (Ponto de mínimo local). Um ponto $\mathbf{x}^* \in \mathbb{R}^n$ é um ponto de mínimo local ou um minimizador local de E se existe uma bola aberta $B(\mathbf{x}^*, \varepsilon)$, centrada em $\mathbf{x}^*$ de raio $\varepsilon > 0$, tal que, para todo $\bar{\mathbf{x}}$ em $B(\mathbf{x}^*, \varepsilon)$

$$E(\mathbf{x}^*) \leq E(\bar{\mathbf{x}}).$$

Se $E(\mathbf{x}^*) < E(\bar{\mathbf{x}})$ para todo $\bar{\mathbf{x}}$ em $B(\mathbf{x}^*, \varepsilon)$, então $\mathbf{x}^*$ é dito um ponto de *mínimo local estrito*.

Definição 2.4 (Ponto de mínimo global). Um ponto $\mathbf{x}^*$ é um ponto de mínimo global ou um minimizador global de E em $\mathbb{R}^n$, se para todo $\mathbf{x} \in \mathbb{R}^n$

$$E(\mathbf{x}^*) \leq E(\mathbf{x}).$$

Se $E(\mathbf{x}^*) < E(\mathbf{x})$ para todo $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^*$ é dito um ponto de *mínimo global estrito*.

O conceito de mínimo global pode ser também formulado em relação a um subconjunto $\mathcal{I}$ de $\mathbb{R}^n$. Neste caso, se para todo $\mathbf{x}^* \in \mathcal{I}$, tem-se que $E(\mathbf{x}^*) \leq E(\mathbf{x})$, então $\mathbf{x}^*$ é um ponto de mínimo global em $\mathcal{I}$. De modo análogo define-se pontos de *máximo local e global* de E.

Se a função E admitir pelo menos um mínimo local, este ponto não é necessariamente único. Neste caso, para cada mínimo local de E, existe um conjunto de minimizadores locais. Este conceito é formalizado na definição 2.5.

Definição 2.5 (Conjunto de minimizadores). Um conjunto $\mathcal{M} \subset \mathbb{R}^n$ é um conjunto de minimizadores locais de E se, para todo $\mathbf{x}^* \in \mathcal{M}$, existe uma bola aberta de raio ε centrada em $\mathbf{x}^*$, denotada por $B(\mathbf{x}^*, \varepsilon)$, tal que

$$E(\mathbf{x}^*) \leq E(\bar{\mathbf{x}}),$$

para todo $\bar{\mathbf{x}} \in B(\mathbf{x}^*, \varepsilon)$. Se $E(\mathbf{x}^*) < E(\bar{\mathbf{x}})$, para todo $\mathbf{x}^* \in \mathcal{M}$, então $\mathcal{M}$ é um conjunto de minimizadores locais estritos. Se $B(\mathbf{x}^*, \varepsilon) = \mathbb{R}^n$, então $\mathcal{M}$ é um *conjunto de minimizadores globais* de E.

Antes apresentamos teoremas que estabelecem condições que a função E deve satisfazer em um ponto $\mathbf{x}^*$ para que este seja um ponto de mínimo, definimos formalmente matrizes simétricas definidas positivas.

Definição 2.6 (Matriz definida positiva). Uma matriz simétrica $\mathbf{A}$ é definida positiva se e somente se a função quadrática $\mathbf{x}^T\mathbf{A}\mathbf{x}$ é positiva, para todo vetor $\mathbf{x}$ real não nulo.

Prova-se que $\mathbf{A}$ é definida positiva se e somente se todos os seus autovalores são positivos [27].

Teorema 2.1 (Condições para um mínimo local estrito). *Se E tem derivadas parciais de primeira e segunda ordens contínuas em um subconjunto D de $\mathbb{R}^n$, $\mathbf{x}^*$ é um ponto interior*

de D que é um ponto crítico de E, e a matriz hessiana $\nabla^2 E(\mathbf{x}^*)$ é definida positiva, então $\mathbf{x}^*$ *é um ponto de* mínimo local estrito *de E em D.* ■

No teorema 2.1, a condição $\nabla E(\mathbf{x}^*) = \mathbf{0}$ possibilita a identificação do ponto crítico. Por outro lado, a matriz hessiana de E em $\mathbf{x}^*$ fornece a curvatura da função E no ponto crítico $\mathbf{x}^*$.

Consideremos como exemplo a função $E(x) = x^3 - x^2$ e seja $D = \mathbb{R}$. A derivada de primeira ordem de E em relação a x é $dE/dx = 3x^2 - 2x$, daí temos que os pontos críticos de E são $x_1 = 2/3$ e $x_2 = 0$. A derivada de segunda ordem da função E é $d^2E/dx^2 = 6x - 2$, que nos pontos críticos x_1 e x_2 assume os valores

$$\frac{d^2E}{dx^2}(x_1) = 2 \quad e \quad \frac{d^2E}{dx^2}(x_2) = -2.$$

Como a derivada de segunda ordem no ponto crítico x_1 é positiva, a concavidade da função E neste ponto está voltada para cima, então x_1 é um ponto de mínimo local estrito em $\mathbb{R}$. Ao passo que no ponto crítico x_2, a concavidade está voltada para baixo, o que o caracteriza como um ponto de máximo local estrito em $\mathbb{R}$.

Teorema 2.2 (Condições para um mínimo global). *Se E possui derivadas parciais de primeira e segunda ordens contínuas em $\mathbb{R}^n$, $\mathbf{x}^*$ é um ponto crítico de E, e $\nabla^2 E(\mathbf{x})$ é semi-definida positiva em todo $\mathbb{R}^n$, então* $\mathbf{x}^*$ *é um ponto de* mínimo global *de E. Se $\nabla^2 E(\mathbf{x})$ for definida positiva em todo $\mathbb{R}^n$, então* $\mathbf{x}^*$ *é um ponto de* mínimo global estrito. ■

Consideremos como exemplo a função quadrática $E(x) = x^2$. A derivada de primeira ordem é $dE/dx = 2x$, e o ponto crítico correspondente é $x^* = 0$. A derivada de segunda ordem $d^2E/dx^2 = 2$ é positiva para todo $x \in \mathbb{R}$, logo a concavidade da parábola descrita por $E(x) = x^2$ está voltada para cima, o que permite concluir que o ponto crítico é de mínimo global estrito.

Para maiores detalhes sobre a teoria de máximos e mínimos de funções diferenciáveis de diversas variáveis, ver por exemplo [28, 29]. Caso a função E não possua derivadas parciais de primeira e segunda ordem contínuas, os teoremas 2.1 e 2.2 não podem ser utilizados para caracterizar os pontos de mínimo da função E.

2.2 Convexidade

Os problemas de otimização estudados neste trabalho consistem em minimizar uma função convexa em um conjunto convexo. Nesta seção são apresentadas definições, propriedades e teoremas sobre funções convexas e conjuntos convexos utilizados neste texto.

2.2.1 Conjuntos convexos

A idéia intuitiva de conjuntos convexos está ilustrada na figura 2.1, e é formalizada através da definição 2.7.

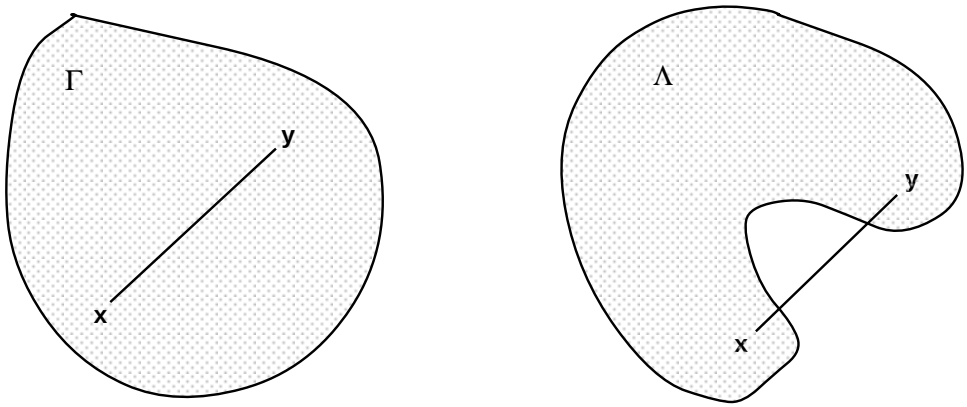


Figura 2.1: Γ é convexo e Λ é não convexo

Definição 2.7 (Conjunto convexo). O conjunto Γ é convexo se para quaisquer $\mathbf{x}, \mathbf{y} \in \Gamma$, o segmento de reta definido por

$$[\mathbf{x}, \mathbf{y}] = \{\mathbf{x}, \mathbf{y} \in \Gamma : \alpha\mathbf{x} + (1-\alpha)\mathbf{y},\ \alpha \in [0,1]\},$$

que une os pontos x e y está contido em Γ.

Teorema 2.3. *Seja Γ um conjunto convexo e $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)} \in \Gamma$. Dados $\alpha_1, \ldots, \alpha_p$ não negativos tais que $\sum_k \alpha_k = 1$, então*

$$\sum_{k=1}^{p} \alpha_k \mathbf{x}^{(k)} \in \Gamma,$$

em que a soma $\sum_{k=1}^{p} \alpha_k \mathbf{x}^k$ é denominada combinação convexa dos vetores $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(p)}$. ■

Definição 2.8 (Fecho convexo). Seja D um subconjunto de $\mathbb{R}^n$. O fecho convexo de D corresponde ao menor conjunto convexo que contém D. Este conjunto é denotado por $co(D)$.

O fecho convexo de um conjunto $D \subset \mathbb{R}^n$, corresponde à intersecção de todos os subconjuntos convexos de $\mathbb{R}^n$ que contém D [28]. O teorema 2.4, garante que o fecho convexo do conjunto D é o conjunto de todas as combinações convexas dos elementos de D.

Teorema 2.4. *Seja D qualquer subconjunto de $\mathbb{R}^n$, então o fecho convexo de D, denotado por $co(D)$, coincide com o conjunto de todas as combinações convexas de vetores em D, isto é, para um número natural arbitrário r, o conjunto*

$$co(D) = \left\{ \mathbf{x} = \sum_{i=1}^{r} \alpha_i \mathbf{x}_i : \mathbf{x}_i \in D, \alpha_i \geq 0, \sum_{i=1}^{r} \alpha_i = 1 \right\}$$

é o fecho convexo do conjunto D. ■

2.2.2 Funções convexas

Definição 2.9 (Função convexa). Seja $E : \Gamma \to \mathbb{R}$, tal que $\Gamma \subset \mathbb{R}^n$ é um conjunto convexo. A função E é convexa se,

$$E(\alpha \mathbf{x}^{(1)} + [1-\alpha]\mathbf{x}^{(2)}) \leq \alpha E(\mathbf{x}^{(1)}) + [1-\alpha]E(\mathbf{x}^{(2)}), \tag{2.1}$$

para todo $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in \Gamma$ e $\alpha \in [0,1]$.

O segundo membro da desigualdade (2.1), descreve o segmento de reta unindo os pontos $E(\mathbf{x}^{(1)})$ e $E(\mathbf{x}^{(2)})$, e o argumento da função E, no primeiro membro, descreve o segmento de reta unindo os pontos $\mathbf{x}^{(1)}$ e $\mathbf{x}^{(2)}$. Geometricamente, pela definição 2.9, E é convexa se o segmento de reta que une os pontos $E(\mathbf{x}^{(1)})$ e $E(\mathbf{x}^{(2)})$ está localizado acima ou sobre gráfico da função E. A igualdade em (2.1) é válida se a função E for linear.

Teorema 2.5. *Seja $E : \Gamma \to \mathbb{R}$, em que $\Gamma \subset \mathbb{R}^n$ é convexo e E tem derivadas parciais de primeira ordem contínuas em Γ. Então a função $E(\mathbf{x})$ é convexa em Γ se e somente se*

$$E(\mathbf{x}) - E(\mathbf{x}_0) \geq \nabla^T E(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0),$$

em que $\mathbf{x}, \mathbf{x}_0 \in \Gamma$. ■

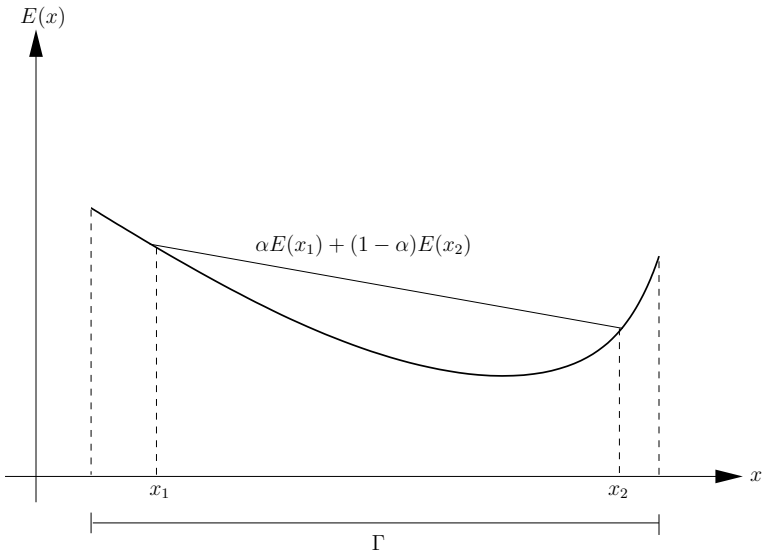


Figura 2.2: Interpretação geométrica da definição de função convexa para $E : \mathbb{R} \to \mathbb{R}$.

O teorema 2.6 garante que a soma de n funções convexas também é convexa. Este teorema pode ser provado utilizando diretamente a definição 2.9.

Teorema 2.6 (Soma de funções convexas). *Se $E_1(\mathbf{x}), \ldots, E_n(\mathbf{x})$ são funções convexas em um conjunto convexo $\Gamma \subset \mathbb{R}^n$, então*

$$E(\mathbf{x}) = E_1(\mathbf{x}) + E_2(\mathbf{x}) + \cdots + E_n(\mathbf{x})$$

também é convexa em Γ. ■

Teorema 2.7 (Condição de Lipschitz [30]). *Toda função convexa E satisfaz uma condição de Lipschitz em qualquer conjunto convexo limitado, isto é, para qualquer conjunto convexo limitado $\Gamma \subset \mathbb{R}^n$, existe um escalar L finito tal que*

$$|E(\mathbf{x}) - E(\mathbf{x}_0)| \leq L\|\mathbf{x} - \mathbf{x}_0\|, \quad \mathbf{x}, \mathbf{x}_0 \in \Gamma.$$

2.2.3 Minimização de funções convexas

As propriedades e a caracterização dos minimizadores de uma função convexa E, são formulados em relação a um subconjunto convexo Γ de $\mathbb{R}^n$, onde assume-se que E é convexa. Em particular, se E for convexa em todo $\mathbb{R}^n$, então $\Gamma = \mathbb{R}^n$.

Teorema 2.8. *Seja $E : \Gamma \to \mathbb{R}$ convexa em um conjunto convexo $\Gamma \subset \mathbb{R}^n$. Então qualquer minimizador local de E em Γ é também um minimizador global neste conjunto.* ■

O teorema anterior é provado em [28]. Se $E : \Gamma \rightarrow \mathbb{R}$ é uma função convexa de classe C^2 em Γ, então os teoremas 2.1 e 2.2, apresentados na seção 2.1, caracterizam os pontos de mínimo de E neste conjunto. Uma condição necessária e suficiente para que um ponto $\mathbf{x}^*$ em um conjunto Γ seja um minimizador global da função E em Γ, é dada pelo teorema a seguir [28].

Teorema 2.9 (Condição para um minimizador global de uma função convexa [28]). *Se $E : \Gamma \rightarrow \mathbb{R}$ é uma função convexa com derivadas parciais de primeira ordem contínuas em um conjunto convexo $\Gamma \subset \mathbb{R}^n$, então todo ponto crítico $\mathbf{x}^*$ de E em Γ é um minimizador global de E neste conjunto.* ■

O teorema 2.9 garante que todo $\mathbf{x}^*$ em um conjunto convexo $\Gamma \subset \mathbb{R}^n$, tal que $\nabla E(\mathbf{x}^*) = \mathbf{0}$, é um minimizador global de E no conjunto convexo Γ. Este resultado está baseado na hipótese da existência e da continuidade das derivadas parciais de primeira ordem da função E no conjunto Γ. Caso estas hipóteses não sejam satisfeitas, tem-se um problema de minimização de uma função convexa não-diferenciável.

2.2.4 Minimização de funções convexas não-diferenciáveis

Os teoremas 2.1 e 2.2, que possibilitam a identificação dos mínimos de funções através das suas derivadas de segunda ordem, não são aplicáveis quando as funções são não-diferenciáveis. Para ilustrar esta afirmação, considere o problema de encontrar o conjunto de minimizadores da função módulo de x, formulado como segue:

$$\text{minimizar}_x \, f(x) = |x|. \tag{2.2}$$

Utilizando a definição 2.9, pode-se provar que a função módulo, denotada por f em (2.2), é convexa em todo $\mathbb{R}$. Sejam $x, y \in \mathbb{R}$ e $\alpha \in [0, 1]$, então, pela desigualdade triangular, tem-se que $|x + (1 - \alpha)y| \leq |x| + (1 - \alpha)|y|$, para todo $x, y \in \mathbb{R}$, isto é,

$$f(x + (1 - \alpha)y) \leq f(x) + (1 - \alpha)f(y).$$

Como a desigualdade anterior é satisfeita para quaisquer $x, y \in \mathbb{R}$ e $\alpha \in [0, 1]$, então, pela definição 2.9, a função módulo é convexa em todo o conjunto $\mathbb{R}$. O gráfico de f é mostrado na figura 2.3.

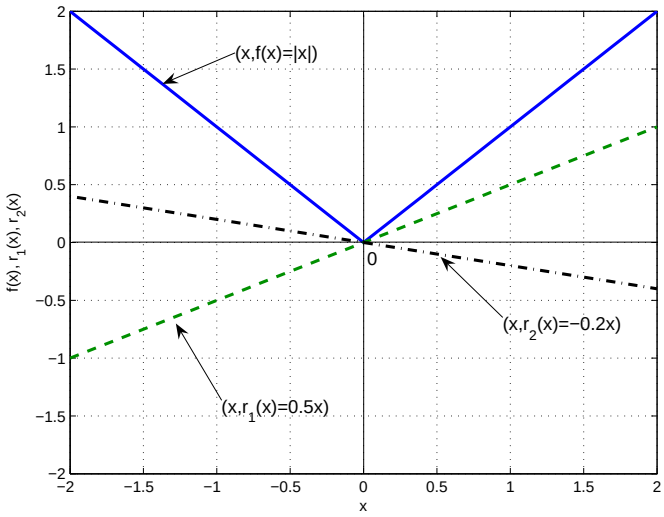


Figura 2.3: Gráfico da função $f(x) = |x|$ e das retas $r_1(x) = 0.5x$ e $r_2(x) = -0.2x$. A função módulo de x é convexa e não-diferenciável na origem, e o mínimo global desta função ocorre no ponto em que $f(x)$ é não-diferenciável. As retas $r_1(x)$ e $r_2(x)$ desempenham o papel de retas tangentes ao gráfico da função f na origem, mostrando que a derivada de f neste ponto não existe, no sentido usual

As derivadas de f são dadas como segue:

- Se $x > 0$, então a derivada de f em relação a x é

$$\frac{df}{dx^+} = +1. \tag{2.3}$$

- Se $x < 0$, então a derivada é

$$\frac{df}{dx^-} = -1. \tag{2.4}$$

Por (2.3) e (2.4), nota-se as derivadas de f à esquerda e à direita da origem são diferentes. Conseqüentemente, a derivada de f neste ponto não existe no sentido usual, o que a caracteriza como uma função não-diferenciável.

A derivada de uma função diferenciável de uma única variável em um ponto x_0, corresponde ao coeficiente angular da reta tangente ao gráfico da função no ponto de abscissa x_0. Logo, nos pontos em que f é diferenciável, temos que, para cada $x < 0$, o coeficiente angular da reta tangente é igual a -1, e para cada $x > 0$, é igual a $+1$.

Por outro lado, para $x = 0$, qualquer reta passando pela origem, com coeficiente angular no intervalo $[-1, 1]$, desempenha o papel de reta tangente ao gráfico de f no ponto de

abscissa $x = 0$ e, conseqüentemente, a derivada da função módulo neste ponto não existe no sentido usual. Logo, existe uma família de retas, com coeficiente angular no intervalo $[-1, 1]$, que desempenham o papel de retas tangentes ao gráfico de f no ponto de abscissa $x = 0$, sendo que duas destas retas, dadas por $r_1(x) = 0.5x$ e $r_2(x) = -0.2x$, com coeficientes angulares iguais a 0.5 e -0.2, respectivamente, são mostradas na figura 2.3.

Observe também que o mínimo global de f ocorre em $x^* = 0$, que é o ponto de não-diferenciabilidade da função módulo. Claramente, a condição que o ponto $\mathbf{x}^*$ tem que satisfazer para ser um ponto crítico de f, exigida na definição 2.2, não é satisfeita devido à inexistência, no sentido usual, da derivada de f na origem. Conseqüentemente, os teoremas 2.1, 2.2 e 2.9, que caracterizam os pontos de mínimo de uma função, também não são satisfeitos.

A caracterização dos mínimos desta classe de funções, é feita utilizando resultados da teoria de otimização de funções não-diferenciáveis [30, 31]. Estes resultados possibilitam a caracterização dos mínimos destas funções, mesmo que estes ocorram em pontos em que a função não é diferenciável.

Consideremos agora uma função $E : \Gamma \to \mathbb{R}^n$ convexa em um conjunto convexo $\Gamma \subset \mathbb{R}^n$, e possivelmente não-diferenciável em um conjunto $\Delta \subset \Gamma$. Os conceitos de subdiferencial de E em um ponto $\mathbf{x}_0 \in \Gamma$ e de subgradiente, ou gradiente generalizado, foram introduzidos por Clarke [31], e generalizam a definição do gradiente da função E.

Definição 2.10 (Subdiferencial e subgradiente de uma função convexa). Seja $E : \Gamma \to \mathbb{R}$ uma função convexa em um conjunto convexo $\Gamma \subset \mathbb{R}^n$. O subdiferencial de E em um ponto $\mathbf{x}_0 \in \Gamma$ é o conjunto descrito por

$$\partial E(\mathbf{x}_0) = \{\mathbf{e} : E(\mathbf{x}) - E(\mathbf{x}_0) \geq \mathbf{e}^T(\mathbf{x} - \mathbf{x}_0)\},$$

sendo que $\mathbf{x}, \mathbf{x}_0 \in \Gamma$. Um vetor $\mathbf{e} \in \partial E(\mathbf{x}_0)$ é denominado um subgradiente ou gradiente generalizado de E em $\mathbf{x}_0$.

De acordo com Dem'yanov e Vasil'ev [30, teorema 5.1], o conjunto $\partial E(\mathbf{x}_0)$ é não-vazio, convexo, e compacto. Se E for diferenciável em $\mathbf{x}_0$, então o conjunto $\partial E(\mathbf{x}_0)$ contém apenas um elemento, que é o próprio gradiente de E em $\mathbf{x}_0$.

Como E é convexa, então pelo teorema 2.7, E é localmente Lipschitz em uma vizinhança de $\mathbf{x}_0 \in \Gamma$, então, de acordo com Clarke [31, teorema 2.5.1], o subdiferencial de E em $\mathbf{x}_0$ é

caracterizado por

$$\partial E(\mathbf{x}_0) = co\{\lim_{i\to\infty} \nabla E(\mathbf{x}_i) : \mathbf{x}_i \to \mathbf{x}_0,\, \mathbf{x}_i \notin \Delta\}, \tag{2.5}$$

sendo que Δ é o conjunto em que E é não-diferenciável.

Utilizando os conceitos de subdiferencial e subgradiente, dados na definição 2.10, uma condição necessária e suficiente para que um ponto $\mathbf{x}^*$ seja um minimizador global da função convexa E, é dada pelo teorema 2.10.

Teorema 2.10 (Condição necessária e suficiente para um mínimo global de uma função convexa não-diferenciável [30]). *Um ponto $\mathbf{x}^*$ em $\Gamma \subset \mathbb{R}^n$ é um minimizador global da função convexa $E : \Gamma \to \mathbb{R}$ em Γ, se e somente se a inclusão* (2.6) *for satisfeita.*

$$\mathbf{0} \in \partial E(\mathbf{x}^*), \tag{2.6}$$

em que $\mathbf{0}$ é o vetor nulo em $\mathbb{R}^n$. ■

O teorema anterior garante uma condição necessária e suficiente para que o ponto $\mathbf{x}^*$ seja um mínimo global de uma função convexa E em um conjunto $\Gamma \subset \mathbb{R}^n$, é que o vetor nulo seja um subgradiente de E no ponto $\mathbf{x}^*$. Observe que se a função E for diferenciável, então o subdiferencial de E em $\mathbf{x}^*$ possui apenas um elemento, que é o próprio gradiente de E neste ponto. Neste caso a condição necessária e suficiente para que este ponto seja um minimizador global de E é simplesmente $\nabla E(\mathbf{x}^*) = \mathbf{0}$.

O teorema 2.10 é de fundamental importância, e a prova deste resultado pode ser encontrada em [30]. Para maiores detalhes sobre subdiferenciais e subgradientes, ver, por exemplo, [30] e [31]. Uma introdução à teoria de inclusões diferenciais é apresentada em [32].

Retornando ao problema (2.2) apresentado início desta seção, pode-se utilizar a definição 2.10 para determinar o subdiferencial da função $f(x) = |x|$ em um ponto x_0 e, utilizando o teorema 2.10, provar que a origem é um ponto de mínimo global. De fato, como a função módulo é convexa em todo $\mathbb{R}^n$, pela definição 2.10, o subdiferencial de f em um ponto x_0 é dado pelo conjunto:

$$\partial f(x_0) = \{\xi : |x| - |x_0| \geq \xi(x - x_0)\}, \tag{2.7}$$

em que ξ é uma derivada generalizada de f em x_0.

Precisamos obter os valores das derivadas generalizadas ξ, tais que, para cada escalar $x_0 \in \mathbb{R}$, a desigualdade $|x| - |x_0| \geq \xi(x - x_0)$ seja válida para todo $x \in \mathbb{R}$. Se $x_0 \neq 0$, duas possibilidades precisam ser analisadas:

- Se $x_0 > 0$, então $|x_0| = x_0$. Logo, a desigualdade em (2.7) pode ser escrita como

$$|x| - x_0 \geq \xi(x - x_0).$$

  Isolando o termo $|x|$ no primeiro membro:

$$|x| \geq \xi x + x_0(1 - \xi).$$

  Claramente, para que a desigualdade anterior seja válida para cada $x_0 \in \mathbb{R}$, e para todo $x \in \mathbb{R}$, o segundo termo no segundo membro precisa ser nulo, logo $\xi = +1$.

- Se $x_0 < 0$, então $|x_0| = -x_0$. Neste caso, a desigualdade em (2.7) toma a forma $|x| + x_0 \geq \xi(x - x_0)$, que pode ser reescrita como:

$$|x| \geq \xi x - x_0(\xi + 1).$$

  Utilizando argumentos análogos aos utilizados para o caso anterior, em que $x_0 > 0$, pode-se concluir que $\xi = -1$.

Os valores das derivadas generalizadas obtidos acima são, de fato, as derivadas da função módulo nos pontos em que esta é diferenciável. Neste caso, o subdiferencial de f em x_0 contém apenas um elemento, que é a própria derivada da função no ponto considerado. Estas derivadas também foram determinadas em (2.3) e (2.4).

Por outro lado, se $x_0 = 0$, então o subdiferencial de f é dado por

$$\partial f(0) = \{\xi : |x| \geq \xi x\}. \tag{2.8}$$

Se $x \geq 0$, então $|x| = x$, e a desigualdade em (2.8) pode ser reescrita como $x \geq \xi x$. Daí, tem-se que $(1 - \xi)x \geq 0$. Como $x \geq 0$, para que esta desigualdade seja satisfeita, é necessário que $(1 - \xi) \geq 0$, o que implica em $\xi \leq 1$ ou, equivalentemente, $\xi \in (-\infty, 1]$.

Por outro lado, se $x < 0$, então $|x| = -x$, e a desigualdade em (2.8) pode ser escrita como $x \leq -\xi x$. Esta desigualdade pode ser reescrita na forma $(1 + \xi)x \leq 0$. Como $x < 0$, para que esta desigualdade seja satisfeita, é necessário que $(1 + \xi) \geq 0$, o que implica em $\xi \in [-1, +\infty)$. Logo, o subdiferencial de f em $x_0 = 0$ é dado pela intersecção dos conjuntos descritos pelos intervalos $[-1, +\infty)$ e $(-\infty, 1]$:

$$\partial f(0) = [-1, +\infty) \cap (-\infty, 1] = [-1, 1].$$

Como $0 \in \partial f(0) = [-1, 1]$, então, pelo teorema 2.10, a origem é um ponto de mínimo global da função módulo. Obviamente, o subdiferencial de f em um ponto x_0 também pode ser obtido utilizando (2.5).

2.3 Sistemas dinâmicos

Considere o seguinte sistema de equações diferenciais ordinárias (EDOs):

$$\dot{\mathbf{x}}(t) = \mathbf{g}(\mathbf{x}(t)), \tag{2.9}$$

em que $\mathbf{x}(t) \in \mathbb{R}^n$, $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ é Lipschitz contínua em todo o seu domínio e $\dot{\mathbf{x}}(t) = d\mathbf{x}(t)/dt$ é a derivada no tempo de $\mathbf{x}(t)$. Este sistema é dito *autônomo* ou *invariante no tempo* pois a função $\mathbf{g}$ não depende explicitamente do tempo t [33].

Um conceito fundamental no estudo de equações diferenciais ordinárias é o conceito de *ponto de equilíbrio*, formalizado pela definição 2.11.

Definição 2.11 (Ponto de equilíbrio). Um ponto $\mathbf{x}^*$ é um ponto de equilíbrio se qualquer trajetória de (2.9) iniciando em $\mathbf{x}^*$ permanece neste ponto indefinidamente.

A definição 2.11 implica que um ponto de equilíbrio $\mathbf{x}^*$ da equação (2.9) é aquele que anula a derivada no tempo de $\mathbf{x}(t)$. Logo, um ponto de equilíbrio $\mathbf{x}^*$ é uma raíz real da equação

$$\mathbf{g}(\mathbf{x}^*) = \mathbf{0}.$$

A equação anterior pode admitir diversas raízes reais, podendo haver múltiplos pontos de equilíbrio, que podem ser isolados, se não existirem outros pontos de equilíbrio em uma

determinada vizinhança deste ponto, ou pode haver um contínuo de pontos de equilíbrio dentro desta vizinhança.

A análise da estabilidade do ponto de equilíbrio $\mathbf{x}^*$ da equação (2.9), é feita assumindo o ponto de equilíbrio na origem. A generalidade não é perdida, pois, caso $\mathbf{x}^* \neq \mathbf{0}$, definindo a variável $\mathbf{y}$ como

$$\mathbf{y} := \mathbf{x} - \mathbf{x}^*,$$

que é a própria variável $\mathbf{x}$ deslocada pelas coordenadas do ponto de equilíbrio $\mathbf{x}^*$, temos que

$$\dot{\mathbf{y}} = \dot{\mathbf{x}} = \mathbf{g}(\mathbf{y} + \mathbf{x}^*).$$

Como $\mathbf{x}^*$ é constante, define-se uma função $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, tal que $\mathbf{f}(\mathbf{y}) := \mathbf{g}(\mathbf{y} + \mathbf{x}^*)$. Deste modo, na variável $\mathbf{y}$, temos o seguinte sistema de equações diferenciais ordinárias

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \quad \mathbf{f}(\mathbf{0}) = \mathbf{g}(\mathbf{x}^*) = \mathbf{0},$$

que possui o ponto de equilíbrio na origem. Portanto, nas análises de estabilidade, pode-se sempre assumir que $\mathbf{x}^* = 0$ e $\mathbf{g}(\mathbf{0}) = \mathbf{0}$ [33].

Definição 2.12 (Estabilidade). O ponto de equilíbrio $\mathbf{x}^* = \mathbf{0}$ do sistema (2.9) é estável se para todo $\varepsilon > 0$ existe $\delta > 0$, dependente de ε, tal que

$$\|\mathbf{x}(0)\| < \delta \Rightarrow \|\mathbf{x}(t)\| < \varepsilon, \quad \forall\, t > 0.$$

Definição 2.13 (Estabilidade assintótica). O ponto de equilíbrio $\mathbf{x}^* = \mathbf{0}$ do sistema (2.9) é *assintoticamente estável* se:

- É estável;
- Existe $\delta > 0$ tal que $\|\mathbf{x}(0)\| < \delta \Rightarrow \mathbf{x}(t) \to \mathbf{0}$ quando $t \to \infty$.

Se a estabilidade assintótica é verificada para quaisquer condições iniciais, então o ponto de equilíbrio é *globalmente assintoticamente estável*.

A aplicação direta das definições 2.12 e 2.13 para verificar a estabilidade do ponto de equilíbrio $\mathbf{x}^* = \mathbf{0}$ do sistema (2.9), em geral, não é uma tarefa simples. Este problema é

resolvido através do *método direto de Lyapunov*, que é baseado na analogia que a energia de um sistema é dissipada até que este eventualmente atinja um ponto de equilíbrio estável. Os teoremas de estabilidade de Lyapunov garantem que outras funções, que não a função de energia do sistema físico, podem ser usadas para a verificação de estabilidade, desde que possuam certas características.

Seja $V : \mathbb{R}^n \to \mathbb{R}$ uma função continuamente diferenciável em todo $\mathbb{R}^n$, e $\dot{V}$ a derivada no tempo da função V ao longo das trajetórias do sistema (2.9), dada por

$$\dot{V}(\mathbf{x}(t)) = (\nabla V)^T \dot{\mathbf{x}} = (\nabla V)^T \mathbf{g}(\mathbf{x}).$$

A derivada no tempo de V ao longo das trajetórias do sistema (2.9) está relacionada diretamente aos estados deste sistema. Portanto, se $\bar{\mathbf{x}}(t)$ é uma solução de (2.9), iniciando em $t = 0$, e $\dot{V} < 0$, então a função V decresce monotonicamente ao longo da solução $\bar{\mathbf{x}}(t)$. Esta é a base da teoria de estabilidade de Lyapunov. Antes de enunciar os teoremas de Lyapunov, as definições a seguir são necessárias.

Definição 2.14 (Função definida positiva). Uma função $V : \mathbb{R}^n \to \mathbb{R}$, é localmente definida positiva em uma vizinhança $B(\mathbf{0}, \varepsilon)$, centrada na origem e de raio ε, se

- $V(\mathbf{x}) > 0$ quando $\mathbf{x} \neq 0$, $\mathbf{x} \in B(\mathbf{0}, \varepsilon)$;
- $V(\mathbf{0}) = 0$.

Caso V seja definida positiva em todo $\mathbb{R}^n$, então V é dita globalmente definida positiva.

Definição 2.15 (Função semi-definida positiva). Uma função $V : \mathbb{R}^n \to \mathbb{R}$, é localmente semi-definida positiva em uma vizinhança $B(\mathbf{0}, \varepsilon)$, centrada na origem e de raio ε, se

- $V(\mathbf{0}) = 0$;
- $V(\mathbf{x}) \geq 0$, para todo $\mathbf{x} \in B(\mathbf{0}, \varepsilon)$.

Caso V seja semi-definida positiva em todo $\mathbb{R}^n$, então V é dita globalmente semi-definida positiva.

Definição 2.16 (Função definida negativa e semi-definida negativa). Uma função $V : \mathbb{R}^n \to \mathbb{R}$ é *definida negativa* se $-V$ for definida-positiva. Por outro lado, se $-V$ for semi-definida positiva, então V é dita *semi-definida negativa*.

De posse das definições 2.14 e 2.16, podemos enunciar o teorema de estabilidade de Lyapunov [33].

Teorema 2.11 (Estabilidade no sentido de Lyapunov). *Sejam* $\mathbf{x}^* = \mathbf{0}$ *um ponto de equilíbrio do sistema* (2.9)*, e* $V : \mathbb{R}^n \to \mathbb{R}$ *uma função continuamente diferenciável em uma vizinhança centrada na origem de raio* ε*, denotada por* $B(\mathbf{0}, \varepsilon)$*, com as seguintes características:*

- $V(\mathbf{x})$ *é definida positiva em* $B(\mathbf{0}, \varepsilon)$*;*
- $\dot{V}(\mathbf{x})$ *é semi-definida negativa em* $B(\mathbf{0}, \varepsilon)$*,*

então a origem é estável. ■

Toda função V que satisfaz as condições do teorema anterior é dita uma *função de Lyapunov* associada ao sistema (2.9).

Teorema 2.12 (Estabilidade assintótica). *Seja* $V(\mathbf{x})$ *uma função continuamente diferenciável em uma vizinhança* $B(\mathbf{0}, \varepsilon)$ *da origem, e* $\mathbf{x}^* = \mathbf{0}$ *um ponto de equilíbrio do sistema* (2.9)*, tal que:*

- $\mathbf{x}^* = \mathbf{0}$ *é estável no sentido de Lyapunov;*
- $\dot{V}(\mathbf{x})$ *é definida negativa em* $B(\mathbf{0}, \varepsilon)$*.*

Então a origem é localmente assintoticamente estável. *Se* $B(\mathbf{0}, \varepsilon) = \mathbb{R}^n$ *e, adicionalmente,* V *é radialmente ilimitada, isto é,*

$$V(\mathbf{x}) \to \infty \text{ quando } \|\mathbf{x}\| \to \infty,$$

então a origem é globalmente assintoticamente estável. ■

Os teoremas enunciados acima fornecem condições suficientes para garantir a estabilidade do ponto de equilíbrio de um sistema. Se, para uma determinada função de Lyapunov, os teoremas não forem verificados, não significa que o sistema seja instável, mas apenas que a função de Lyapunov escolhida não permite tirar conclusões sobre a estabilidade do ponto de equilíbrio do sistema.

O Princípio da Invariância

A estabilidade assintótica do ponto de equilíbrio $\mathbf{x}^* = \mathbf{0}$ do sistema (2.9) é freqüentemente difícil de ser garantida utilizando os teoremas 2.11 e 2.12, pois há situações em que a derivada no tempo ao longo das trajetórias do sistema é semi-definida negativa. Ainda assim, é possível tirar conclusões sobre a estabilidade assintótica utilizando o princípio da invariância de LaSalle [34]. Antes de enunciar o teorema de LaSalle, é preciso introduzir o conceito de conjunto invariante.

Definição 2.17 (Conjunto invariante). Um conjunto M é dito invariante em relação ao sistema (2.9), se toda trajetória de (2.9) que inicia em um ponto de M, permanece neste conjunto indefinidamente.

Teorema 2.13 (LaSalle [34]). *Considere o sistema* (2.9) *e seja* $V(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *uma função continuamente diferenciável, tal que:*

- *O conjunto* $\Omega = \{\mathbf{x} : V(\mathbf{x}) < l\}$*, é limitado para algum* $l > 0$
- $\dot{V}(\mathbf{x}) \leq 0$ *para todo* $\mathbf{x} \in \Omega$

Seja $\Gamma = \{\mathbf{x} \in \Omega : \dot{V}(\mathbf{x}) = 0\}$ *e* M *o maior conjunto invariante contido em* Γ*. Então as trajetórias de* (2.9) *iniciadas em* Ω *convergem assintoticamente para o conjunto invariante* M*.* ■

Os resultados apresentados nesta seção são válidos para sistemas de equações diferenciais suaves. Entretanto, o teorema de LaSalle já foi estendido para sistemas de EDOs não-suaves [35], e a generalização do teorema de estabilidade de Lyapunov para sistemas de EDOs não-suaves foi provada em [36].

2.3.1 Sistemas dinâmicos com segundo membro descontínuo

Resultados sobre sistemas dinâmicos com segundo membro descontínuo foram apresentados primeiramente por Filippov em um artigo [37], e posteriormente sob a forma de livro [38]. Neste último trabalho, Filippov define soluções de equações diferenciais com segundo membro descontínuo, e desenvolve a teoria de estabilidade das mesmas.

Outros trabalhos acerca de sistemas dinâmicos com o segundo membro descontínuo e modos deslizantes foram publicados. Utkin [39] apresentou a teoria de sistemas de estrutura

variável e modos deslizantes. Em [7], a teoria de modos deslizantes foi desenvolvida e aplicada a uma série de problemas. A teoria de sistemas de estrutura variável também foi apresentada por De Carlo et al. [40] na forma de um tutorial. Chong et al. [15] desenvolveram uma teoria de convergência em tempo finito para sistemas de equações diferenciais com segundo membro descontínuo.

Nesta seção apresentamos as definições de soluções para esta classe de sistemas dinâmicos, seguindo a metodologia de Filippov [38]. Sejam os seguinte conjuntos:

$$\mathbb{R}^n := \bigcup_{i=1}^{k} \bar{\Omega}_i, \text{ em que cada } \Omega_i \text{ é um conjunto aberto e } \bar{\Omega}_i \text{ é o fecho de } \Omega_i; \tag{2.10}$$

$$\Delta := \bigcup_{i=1}^{k} \Delta_i, \text{ é de medida nula no sentido de Lebesgue, em que cada } \Delta_i \text{ é a fronteira de } \Omega_i.$$

Definição 2.18 (Função contínua por partes [38]). Uma função $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ é *contínua por partes* em $\mathbb{R}^n$, com a partição definida em (2.10), se for contínua em cada conjunto aberto Ω_i, $i = 1, \ldots, k$, e à medida que o valor argumento de $\mathbf{f}$ aproxima-se de um ponto da fronteira Δ_i deste conjunto, o valor desta função tende a um limite finito.

Considere o seguinte sistema de equações diferenciais ordinárias, escrito em notação vetorial:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \tag{2.11}$$

sendo que $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ é uma função vetorial contínua por partes, em que cada componente de f_i de $\mathbf{f}$, é uma função contínua no conjunto aberto Ω_i, e descontínua no conjunto de medida nula Δ_i, que consiste dos pontos de fronteira de Ω_i, para cada $i = 1, \ldots, k$. Filippov [38, pag. 50] define as soluções deste sistema como segue.

Definição 2.19 (Soluções segundo Filippov). Uma solução do sistema de equações diferenciais (2.11), é uma função vetorial $\mathbf{x}(t)$ absolutamente contínua, que satisfaz a inclusão diferencial (2.12), para quase todo $t \in [t_0, t_f]$.

$$\dot{\mathbf{x}}(t) \in F(\mathbf{x}(t)) = co(D), \tag{2.12}$$

sendo que o conjunto D é definido por

$$D = \left\{ \mathbf{f}^{(i)}(\mathbf{x}) = \lim_{\bar{\mathbf{x}} \to \mathbf{x}} \mathbf{f}(\bar{\mathbf{x}}) \,|\, \bar{\mathbf{x}} \in \Omega_i \right\}$$

Se $\mathbf{x}(t) \notin \Delta$ então o conjunto $F(\mathbf{x}(t))$ contém apenas um elemento, que satisfaz a equação (2.11) no sentido usual [38].

Sistemas dinâmicos com o segundo membro descontínuo não admitem pontos de equilíbrio no sentido usual, segundo a definição 2.11. Para esta classe de sistemas dinâmicos, são definidos os pontos de equilíbrio da inclusão diferencial (2.12).

Definição 2.20 (Ponto de equilíbrio de uma inclusão diferencial [32]). Um ponto $\mathbf{x}^*(t) \in \mathbb{R}^n$ que satisfaz

$$\mathbf{0} \in F(\mathbf{x}^*(t)),$$

é dito um ponto de equilíbrio da inclusão diferencial (2.12).

Pela definição 2.8 e pelo teorema 2.4, o fecho convexo do conjunto D, denotado por $co(D)$, é o conjunto de todas as combinações convexas dos vetores pertencentes a D, logo, o conjunto $F(\mathbf{x})$ é dado por:

$$F(\mathbf{x}(t)) = \left\{ \sum_{i=1}^{k} \alpha_i \mathbf{f}^{(i)}(\mathbf{x}(t)) : \; \alpha_i \geq 0, \; \sum_{i=1}^{k} \alpha_i = 1 \right\}. \tag{2.13}$$

Consideremos o caso em que o conjunto Δ consiste de um número finito de hipersuperfícies. Seja Δ_i definido como segue:

$$\Delta_i = \{\mathbf{x} : \varphi_i(\mathbf{x}) = 0\}, \quad (i = 1, \ldots, k)$$

em que cada $\varphi_i : \mathbb{R}^n \to \mathbb{R}$ é uma função de classe C^1. Estas superfícies são conhecidas como *superfícies de descontinuidade* e, neste caso, o segundo membro da equação (2.11) é descontínuo em uma intersecção S das superfícies de descontinuidade Δ_i, $i = 1, \ldots, k$, isto é,

$$S := \Delta_1 \cap \ldots \cap \Delta_k.$$

Neste caso, a superfície S divide o conjunto $\mathbb{R}^n$ em dois espaços, denotados por Ω^+ e

Ω^-, e os valores limites de $\mathbf{f}$ nestes conjuntos são

$$\mathbf{f}^-(\mathbf{x}) = \lim_{\bar{\mathbf{x}}\to\mathbf{x}} \mathbf{f}(\bar{\mathbf{x}}) \quad (\bar{\mathbf{x}} \in \Omega^-)$$
$$\mathbf{f}^+(\mathbf{x}) = \lim_{\bar{\mathbf{x}}\to\mathbf{x}} \mathbf{f}(\bar{\mathbf{x}}) \quad (\bar{\mathbf{x}} \in \Omega^+).$$

Por (2.13), o conjunto $F(\mathbf{x})$ é o conjunto de todos os vetores $\mathbf{v}$ na forma:

$$\mathbf{v} = \alpha\mathbf{f}^+ + \beta\mathbf{f}^-,\ \alpha + \beta = 1.$$

Observe que $\beta = 1 - \alpha$. Logo, conjunto $F(\mathbf{x})$ na definição 2.19 é

$$F(\mathbf{x}) = \{\alpha\mathbf{f}^+ + (1-\alpha)\mathbf{f}^-, 0 \leq \alpha \leq 1\}, \tag{2.14}$$

que é o segmento de reta unindo as extremidades dos vetores $\mathbf{f}^-$ e $\mathbf{f}^+$.

Seja $P(\mathbf{x})$ o hiperplano tangente à superfície S no ponto $\mathbf{x}$. Se para $t \in [t_0, t_f]$ as extremidades dos vetores $\mathbf{f}^-$ e $\mathbf{f}^+$ estiverem localizadas em lados diferentes do hiperplano $P(\mathbf{x})$, então $P(\mathbf{x}) \cap F(\mathbf{x}) \neq \varnothing$. O ponto de intersecção entre o segmento de reta $F(\mathbf{x})$, dado em (2.14), e o hiperplano $P(\mathbf{x})$ é a extremidade de um vetor $\mathbf{f}^0 \in F(\mathbf{x})$. Este vetor satisfaz a equação $\dot{\mathbf{x}} = \mathbf{f}^{(0)}(\mathbf{x})$, que determina a velocidade do movimento ao longo da superfície de descontinuidade S. Este tipo de movimento é chamado *movimento deslizante* ou diz-se que o sistema está em *modo deslizante*.

Por outro lado, se para $t \in [t_0, t_f]$ as extremidades dos vetores $\mathbf{f}^-$ e $\mathbf{f}^+$ estiverem localizadas no mesmo lado do hiperplano $P(\mathbf{x})$, então as trajetórias atravessam a superfície de descontinuidade S. Neste caso, as trajetórias não ficam confinadas à superfície S, e o movimento deslizante não ocorre.

Diversas técnicas estão disponíveis para determinar o conjunto $F(\mathbf{x}(t))$, em (2.12), algumas são discutidas em [38, cap. 2]. A seguir, apresentamos uma breve discussão sobre a definição de soluções através do método do controle equivalente.

2.3.2 Solução por controle equivalente

Esta discussão tem como base a apresentação em [38, pag. 54-55]. A definição de solução por controle equivalente é adequada a sistemas de equações diferenciais na forma

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u_1(\mathbf{x}(t)), \ldots, u_k(\mathbf{x}(t))), \tag{2.15}$$

em que $\mathbf{f}$ é contínua nos seus argumentos e os controles $u_1, \ldots, u_k$ são descontínuos na fronteira Δ_i do conjunto Ω_i, $i = 1, \ldots, k$.

Suponhamos que cada controle u_i é descontínuo apenas em uma superfície suave Δ_i, definida como segue:

$$\Delta_i := \{\mathbf{x}(t) : \varphi_i(\mathbf{x}(t)) = 0\} \qquad (i = 1, \ldots, k). \tag{2.16}$$

Intersecções e até mesmo a coincidência entre as superfícies definidas acima podem ocorrer. Para cada ponto de descontinuidade de cada uma das funções u_i, um conjunto convexo fechado U_i, que é o conjunto de todos os valores possíveis da função u_i, precisa ser definido. Cada um destes conjuntos precisa conter os pontos limites de qualquer seqüência da forma $v_p \in U_i(\mathbf{x})$, em que $\mathbf{x}_p \to \mathbf{x}$, $p = 1, 2, \ldots$. Nos pontos em que a função u_i é contínua, o conjunto U_i possui apenas um elemento, que é o próprio valor que u_i assume neste ponto. Então, a solução de (2.15) é uma função vetorial $\mathbf{x}(t)$ absolutamente contínua, que satisfaz a seguinte inclusão diferencial

$$\dot{\mathbf{x}}(t) \in K(\mathbf{x}(t)),$$

para quase todo t em um intervalo. O conjunto $K(\mathbf{x}(t))$, é definido como o conjunto dos valores da função $\mathbf{f}(\mathbf{x}, u_1(\mathbf{x}), \ldots, u_k(\mathbf{x}))$, em que $\mathbf{x}$ é constante e as funções escalares $u_1, \ldots, u_k$ variam independentemente nos conjuntos $U_1(\mathbf{x}), \ldots, U_k(\mathbf{x})$. Este conjunto de valores da função $\mathbf{f}$ é denotado por $\mathbf{f}(\mathbf{x}, U_1(\mathbf{x}), \ldots, U_k(\mathbf{x}))$, e o conjunto K é dado por

$$K(\mathbf{x}) := \mathbf{f}(\mathbf{x}, U_1(\mathbf{x}), \ldots, U_k(\mathbf{x})). \tag{2.17}$$

Método do controle equivalente

Sem perda de generalidade, consideremos que $\mathbf{x}$ pertence simultaneamente às r ($1 \leq r \leq k$) primeiras superfícies de descontinuidade, definidas em (2.16), isto é:

$$\mathbf{x} \in \Delta_1 \cap \Delta_2 \cap \ldots \cap \Delta_r.$$

Se a trajetória $\mathbf{x}$ não sai imediatamente da superfície de descontinuidade, definimos os controles equivalentes $u_i^{eq}(t, \mathbf{x})$, $j = 1, \ldots, r$, de tal modo que:

(i) o vetor $\mathbf{f}$ em (2.15) seja tangente a todas as k superfícies de descontinuidade definidas em (2.16);

(ii) os valores de cada controle equivalente variam no intervalo $[u_i^-, u_i^+]$, sendo que u_i^- e u_i^+ são os valores limites da função u_i em ambos os lados da superfície de descontinuidade Δ_i.

Substituindo os os r primeiros controles no argumento de $\mathbf{f}$ em (2.15) pelos controles equivalentes correspondentes obtemos:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u_1^{eq}, \ldots, u_r^{eq}, u_{r+1}(\mathbf{x}(t)), \ldots, u_k(\mathbf{x}(t))). \tag{2.18}$$

Pela definição dos controles equivalentes u_i^{eq}, $j = 1, \ldots, r$, o vetor $\mathbf{f}$ é tangente às r primeiras superfícies de descontinuidade, definidas em (2.16). Portanto, os controles equivalentes podem ser determinados através do seguinte sistema de equações:

$$\nabla^T \varphi_i(\mathbf{x}) \mathbf{f}(\mathbf{x}, u_1^{eq}, \ldots, u_r^{eq}, u_{r+1}, \ldots, u_k) = 0 \quad (j = 1, \ldots, r) \tag{2.19}$$

A definição de solução da equação (2.15), utilizando o método do controle equivalente, é formalizada a seguir.

Definição 2.21 (Solução por controle equivalente). Uma solução do sistema (2.15) é uma função vetorial $\mathbf{x} \in \mathbb{R}^n$ absolutamente contínua, que fora das superfícies de descontinuidade satisfaz (2.15), e nas superfícies de descontinuidade, ou na intersecção delas, satisfaz (2.18), para quase todo t em um intervalo.

O exemplo 2.3.1, considerado em [38, pag. 54-55], ilustra a formação do conjunto K, em (2.17).

Exemplo 2.3.1. Seja $r = 1$, então a equação (2.15) apresenta a forma

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u). \tag{2.20}$$

A equação (2.18), que descreve o movimento ao longo da superfície de descontinuidade $S := \{\mathbf{x} : \varphi(\mathbf{x}) = 0\}$, apresenta a forma

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u^{eq}). \tag{2.21}$$

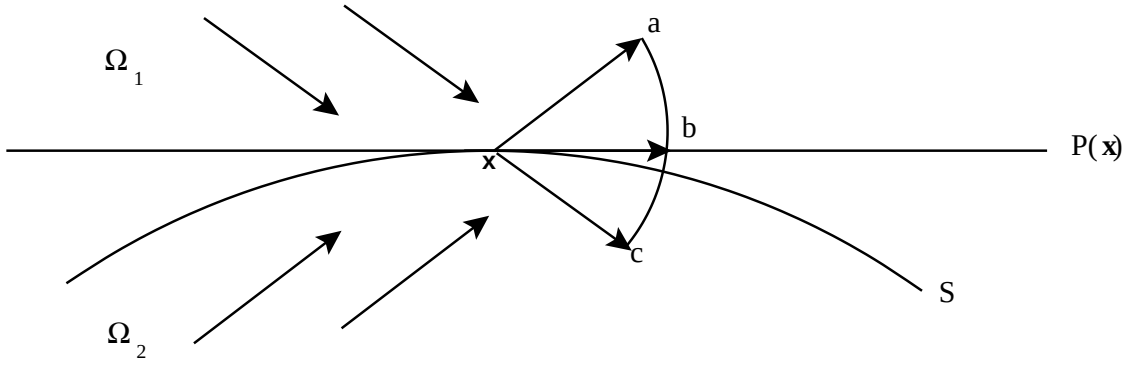


Figura 2.4: Soluções definidas por controle equivalente

Na figura 2.4, a dinâmica em modo deslizante, descrita pela equação (2.21), é ilustrada geometricamente. O domínio Ω é dividido em Ω_1 e Ω_2 pela superfície de descontinuidade S.

O conjunto K, definido em (2.17), é gerado pelas extremidades do vetor $\mathbf{f}(\mathbf{x}, u)$, quando o controle u varia no intervalo $[u^-, u^+]$, gerando o arco abc na figura 2.4. Neste exemplo, o conjunto K é dado por:

$$K(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}(t), U), \tag{2.22}$$

sendo que o conjunto U é dado por:

$$U(t, \mathbf{x}) = \begin{cases} u(\mathbf{x}), \text{ se } \mathbf{x} \notin S \\ [u^-, u^+], \text{ se } \mathbf{x} \in S \end{cases}$$

A solução $\mathbf{x}$ obtida pelo método do controle equivalente, é tal que a extremidade do vetor $\mathbf{f}(\mathbf{x}, u^{eq})$ coincide com o ponto b, localizado na intersecção do plano tangente à superfície de

descontinuidade S com o conjunto K. Esta solução satisfaz a equação (2.19), pois o vetor gradiente $\nabla\varphi(\mathbf{x})$ é ortogonal ao plano tangente à superfície S, denotado por $P(\mathbf{x})$ na figura 2.4.

2.3.3 O problema do chattering

As soluções numéricas de sistemas de equações diferenciais com o segundo membro descontínuo apresentam um fenômeno conhecido como *chattering*. O chattering consiste de oscilações de alta freqüência em uma vizinhança da superfície de descontinuidade [41], que compromete a precisão das soluções numéricas obtidas. Várias técnicas já foram propostas com o objetivo de atenuar ou eliminar o chattering [42, 43].

O problema do chattering também pode ser resolvido através do projeto de controladores de modo deslizante para sistemas discretos. Furuta [44] propôs um controlador de modo deslizante para sistemas lineares discretos que é estável, livre de chattering e insensível à escolha do intervalo de amostragem. Outra proposta foi apresentada por Furuta e Pan [45], em que o conceito de setores deslizantes é usado para projetar controladores para sistemas discretos, livres de chattering e quadraticamente estáveis. Recentemente foram introduzidos controladores para sistemas discretos, em que as taxas de amostragem do sinal de controle e da saída do sistema são diferentes [46]. Estes controladores são dependentes apenas da saída do sistema e não dependem dos estados.

No entanto, os projetos de controladores discretos tratados na literatura, especificamente os citados no parágrafo acima, dependem de produtos e inversão de matrizes, que demandam muito tempo de computação quando os sistemas considerados possuem dimensões elevadas. Por esta razão, optamos pela técnica da *camada limite*, que consiste em aproximar os termos descontínuos por uma função contínua, em uma pequena vizinhança da superfície de descontinuidade [47]; esta vizinhança é chamada camada limite. Esta técnica mostrou-se adequada aos problemas considerados neste trabalho, pois o custo computacional é baixo e as soluções obtidas são suficientemente precisas.

Consideremos como exemplo a equação diferencial escalar abaixo, também considerada por Utkin et al. [43]:

$$\dot{x} = -\mu \text{sign}(x), \tag{2.23}$$

em que μ é uma constante positiva e sign é a função sinal, definida como segue

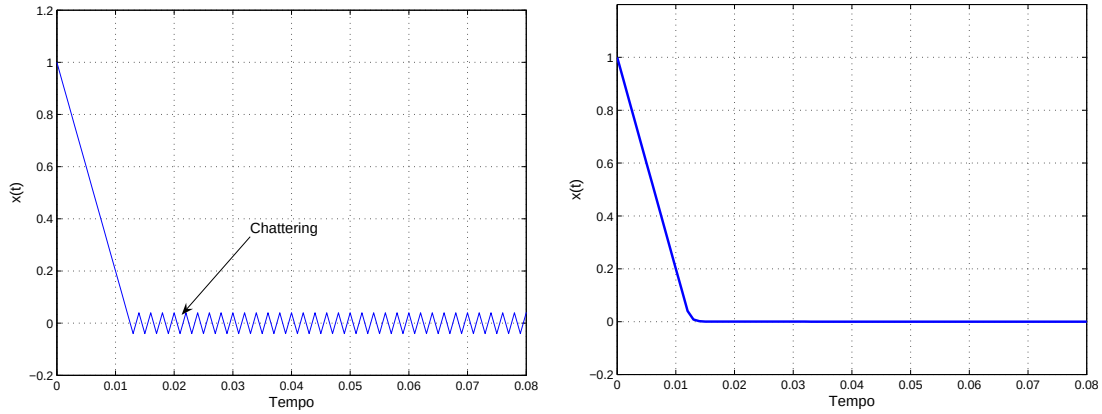
$$\text{sign}(x) = \begin{cases} -1, \text{ se } x < 0 \\ 1, \text{ se } x > 0 \end{cases}$$

O segundo membro da equação (2.23) é descontínuo em $x \equiv 0$. Aplicando a técnica da camada limite em uma vizinhança de raio ε do ponto de descontinuidade do segundo membro de (2.23), a função sign é substituída por

$$\text{ssgn}(x) = \begin{cases} x/\varepsilon, \text{ se } |x| \leq \varepsilon, \\ \text{sign}(x), \text{ caso contrário} \end{cases} \tag{2.24}$$

Note que fora do intervalo $[-\varepsilon, \varepsilon]$, que define a camada limite, a função ssgn é a própria função sign, enquanto que dentro da camada limite, é uma reta de inclinação $1/\varepsilon$, que é uma função contínua. Quando esta técnica é aplicada à equação (2.23), o chattering é eliminado.

O resultado da simulação da equação (2.23) sem a utilização da camada limite é mostrado na figura 2.5-(a). A trajetória resultante apresenta chattering em uma vizinhança do ponto de descontinuidade, comprometendo a precisão da solução. O resultado da simulação utilizando a camada limite é mostrado na figura 2.5(b), onde uma trajetória completamente livre de chattering é mostrada. As duas simulações foram feitas usando $\mu = 80$, $\varepsilon = 0.1$ e a condição inicial escolhida é $x_0 = 1$.



(a) (b)

Figura 2.5: Trajetória do sistema $\dot{x} = -\mu \text{sign}(x)$, com $\mu = 80$ e $x_0 = 1$: (a) Trajetória apresenta chattering em uma vizinhança da origem; (b) Trajetória obtida após a suavização da função sgn através da técnica da camada limite em uma vizinhança de raio igual a $\varepsilon = 0.1$ do ponto $x = 0$

Síntese do capítulo

Neste capítulo, foram introduzidos os principais conceitos matemáticos utilizados neste trabalho.

Os conceitos e resultados necessários para a caracterização dos mínimos de funções de diversas variáveis foram introduzidos, dando destaque à minimização de funções convexas diferenciáveis e não-diferenciáveis. O problema da minimização de funções convexas não diferenciáveis, foi discutido e os conceitos matemáticos necessários para a caracterização dos mínimos destas funções foram introduzidos.

Foi dada uma breve introdução à teoria de sistemas dinâmicos suaves e não-suaves. O método direto de Lyapunov e o princípio da invariância de LaSalle foram apresentados. As soluções de sistemas de equações diferenciais com o segundo membro descontínuo foram formalmente definidas. Finalmente o problema do chattering, comum em implementações numéricas de sistemas dinâmicos não-suaves foi discutido.

No próximo capítulo, a resolução de problemas de otimização através de sistemas gradientes será discutida.

Capítulo 3

Sistemas gradientes aplicados à resolução de problemas de otimização

Neste capítulo, sistemas gradientes não-suaves, obtidos a partir de um método de penalização exata, são utilizados para resolver problemas de otimização. A convergência é analisada através da forma Persidskii dos sistemas gradientes, utilizando funções de Lyapunov do tipo Lure-Persidskii não-suaves.

3.1 Descrição do problema de otimização e o método da função de penalidade

O problema de minimizar uma função escalar f, de classe C^1, em um conjunto $\Omega \subset \mathbb{R}^n$, é formulado como segue:

$$\begin{aligned} &\text{minimizar } f(\mathbf{x}) \\ &\text{sujeito a } \mathbf{x} \in \Omega, \end{aligned} \tag{3.1}$$

em que $\mathbf{x} \in \mathbb{R}^n$ é a variável de decisão, $f : \mathbb{R}^n \to \mathbb{R}$ é a função objetivo ou função de custo, e Ω é o conjunto ou região viável do problema.

O problema (3.1) é dito um *problema de otimização com restrições*. Caso o conjunto Ω seja todo o $\mathbb{R}^n$, então o problema (3.1) é dito um *problema de otimização sem restrições*. O conjunto solução do problema (3.1) é definido como segue.

Definição 3.1 (Solução de um problema de otimização). O conjunto solução do problema de otimização (3.1), quando existe, é o conjunto de todos os vetores $\mathbf{x}^* \in \Omega$ que minimizam

a função objetivo f, isto é,

$$\mathbf{x}^* = \text{argmin}(f), \quad \mathbf{x}^* \in \Omega.$$

Seja o conjunto viável do problema (3.1) dado pela seguinte intersecção

$$\Omega = \Omega_1 \cap \Omega_2 \cap \Omega_3, \tag{3.2}$$

sendo que $\Omega_1 = \{\mathbf{x} : \mathbf{r}(\mathbf{x}) = \mathbf{0}\}$, $\Omega_2 := \{\mathbf{x} : \mathbf{s}(\mathbf{x}) \geq \mathbf{0}\}$, $\Omega_3 = \{\mathbf{x} : \mathbf{v}(\mathbf{x}) \leq \mathbf{0}\}$, e as funções vetoriais $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^{m_1}$, $\mathbf{s} : \mathbb{R}^n \to \mathbb{R}^{m_2}$ e $\mathbf{v} : \mathbb{R}^n \to \mathbb{R}^{m_3}$, são de classe C^1.

A função lagrangeana para o problema (3.1), com o conjunto viável definido em (3.2), é

$$L(\mathbf{x}, \boldsymbol{\delta}, \boldsymbol{\theta}, \boldsymbol{\eta}) = f(\mathbf{x}) + \sum_{i=1}^{m_1} \delta_i r_i(\mathbf{x}) + \sum_{i=1}^{m_2} \theta_i s_i(\mathbf{x}) + \sum_{i=1}^{m_3} \eta_i v_i(\mathbf{x}),$$

sendo que δ_i, θ_i e η_i são os componentes dos vetores $\boldsymbol{\delta}$, $\boldsymbol{\theta}$ e $\boldsymbol{\eta}$, respectivamente, e são conhecidos como multiplicadores de Lagrange.

Definição 3.2 (Ponto regular [6]). Seja $\mathbf{x}^*$ um ponto que satisfaz as restrições em (3.2), e sejam os conjuntos de índices $I_s := \{j : s_j(\mathbf{x}^*) = 0\}$ e $I_v := \{k : v_k(\mathbf{x}^*) = 0\}$. Então $\mathbf{x}^*$ é dito um *ponto regular das restrições* (3.2) se os vetores $\nabla r_i(\mathbf{x}^*)$, $\nabla s_j(\mathbf{x}^*)$ e $\nabla v_k(\mathbf{x}^*)$, $1 \leq i \leq m_1$, $j \in I_s$ e $k \in I_v$, são linearmente independentes.

Se um vetor $\mathbf{x}^*$ é uma solução do problema (3.1) e é um ponto regular das restrições (3.2), então existem multiplicadores de Lagrange $\boldsymbol{\delta}^*$, $\boldsymbol{\theta}^*$ e $\boldsymbol{\eta}^*$, tais que as condições de Karush-Kuhn-Tucker (KKT) [6], dadas em (3.3), são satisfeitas.

$$\begin{cases} \nabla f(\mathbf{x}^*) + \sum_{i=1}^{m_1} \delta_i \nabla r_i(\mathbf{x}^*) + \sum_{i=1}^{m_2} \theta_i \nabla s_i(\mathbf{x}^*) + \sum_{i=1}^{m_3} \eta_i \nabla v_i(\mathbf{x}^*) = \mathbf{0}; \\ \delta_i^* \text{ não tem restrição de sinal}; \\ \theta_i^* \leq \mathbf{0}, \text{ sendo que } \theta_i^* = 0, \text{ se } s_i(\mathbf{x}^*) > 0, \; i = 1, \ldots, m_2; \\ \eta_i^* \geq \mathbf{0}, \text{ sendo que } \eta_i^* = 0, \text{ se } v_i(\mathbf{x}^*) < 0, \; i = 1, \ldots, m_3; \\ \theta_i^* s_i(\mathbf{x}^*) = 0, \; i = 1, \ldots, m_2; \\ \eta_i^* v_i(\mathbf{x}^*) = 0, \; i = 1, \ldots, m_3. \end{cases} \tag{3.3}$$

Assumimos que o problema (3.1) admite solução segundo a definição 3.1. O método utilizado para determinar as soluções dos problemas de otimização propostos, consiste em

convertê-los em um problema equivalente sem restrições, utilizando um método de penalidade exata, e mapear o novo problema em um sistema gradiente.

Método da função de penalidade

Através do método da função de penalidade, problemas de otimização com restrições são convertidos em problemas de otimização sem restrições, equivalentes ao problema original. Através deste método é possível associar problemas de otimização a sistemas do tipo gradiente, que são adequados à implementação utilizando computadores paralelos.

Considere o problema geral de otimização com restrições (3.1). O método da função de penalidade consiste em adicionar um termo à função objetivo f, que incorpora uma penalidade por violar uma restrição. Assim, o problema (3.1) toma a forma

$$\begin{aligned} &\text{minimizar } E(\mathbf{x}, \gamma) = f(\mathbf{x}) + \gamma P(\mathbf{x}) \\ &\mathbf{x} \in \mathbb{R}^n, \end{aligned} \tag{3.4}$$

sendo que E é a função objetivo do problema de otimização sem restrições, γ é um escalar positivo, conhecido como parâmetro de penalidade, e P é a função de penalidade, que satisfaz as seguintes condições:

(i) $P(\mathbf{x})$ é contínua em $\mathbb{R}^n$

(ii) $P(\mathbf{x}) \geq 0$ para todo $\mathbf{x} \in \mathbb{R}^n$

(iii) $P(\mathbf{x}) = 0$ se e somente se $\mathbf{x} \in \Omega$.

Observe que a função P é não-nula apenas fora da região viável do problema. Por esta razão, este método é também conhecido como *penalização externa*, e a função $P(\mathbf{x})$ é dita uma *função de penalidade externa*.

Durante o processo de minimização da função E, para cada violação ao conjunto de restrições, a função P assume um valor positivo, que é somado à função objetivo. Na região viável, P é nula, pois nenhuma restrição é violada.

O método de penalidades para resolver o problema de otimização (3.1), consiste em gerar uma seqüência crescente de parâmetros de penalidade, denotada por $\{\gamma_k\}$, com valores tendendo a infinito e, para cada k, a função $E(\gamma_k, \mathbf{x})$ é minimizada. Este processo gera

uma seqüência infinita de soluções $\{\mathbf{x_k}\}$ que converge para a solução do problema (3.1). A convergência do método da função de penalidade é assegurada pelo teorema 3.1.

Teorema 3.1 (Luenberger [6]). *Seja $\{\mathbf{x}_k\}$ uma seqüência infinita gerada pelo método da função de penalidade. Então todo ponto limite desta seqüência é uma solução do problema* (3.1). ■

Como a seqüência $\{\gamma_k\}$ é crescente, então para valores elevados de γ_k, os mínimos de E ocorrem quando P também é mínimo. Deste modo, à medida que γ_k cresce, a solução do problema (3.4) aproxima-se da solução do problema (3.1), e esta deverá ser obtida quando γ_k tender a infinito. Esta é uma desvantagem no método da função de penalidade, pois para que o método obtenha uma solução exata, no sentido de que a solução dos problemas (3.1) e (3.4) sejam iguais, é necessário um valor infinito do parâmetro de penalidade.

O método de *penalização interna* ou *método de barreiras* [6], utiliza funções de penalidade que estabelecem uma barreira na fronteira do conjunto viável Ω, impedindo que sejam obtidos pontos não viáveis ou de fronteira. A função de barreira $B(\mathbf{x})$ satisfaz às condições (i) e (ii) que a função de penalização externa P satisfaz, e $B(\mathbf{x}) \to \infty$, quando $\mathbf{x}$ aproxima-se da fronteira do conjunto viável Ω. Esta é a condição que caracteriza $B(\mathbf{x})$ como uma função de barreira. Entretanto, este método exige uma condição inicial no interior do conjunto viável Ω, que freqüentemente é difícil ser obtida, especialmente quando o problema considerado possui dimensões elevadas, dificultando o uso prático do método de barreiras.

Devido a estas dificuldades, as funções de penalidade usadas neste trabalho são exatas, o que garante que para um valor finito suficientemente grande do parâmetro γ, a solução do problema (3.4) é exatamente a mesma do problema original. Entretanto, Bertsekas [48] provou que, exceto para problemas com uma estrutura específica, caracterizados em [48], as funções de penalidade exatas são não-diferenciáveis na fronteira do conjunto viável Ω. Consideremos a seguir um exemplo de utilização do método de penalização exata e não-exata.

Exemplo: Seja o seguinte problema de programação quadrática sujeito a restrições lineares:

$$\begin{aligned} &\text{minimizar } \mathbf{x}^T\mathbf{Q}\mathbf{x} \\ &\text{sujeito a } \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{3.5}$$

Seja $\gamma > 0$ o parâmetro de penalidade. Consideremos duas formas de penalização apli-

cadas ao problema (3.5):

a) Penalização não-exata:

$$E_{ne}(\mathbf{x}, \gamma) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \gamma\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2;$$

b) Penalização exata:

$$E_e(\mathbf{x}, \gamma) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \gamma\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1.$$

Observe que a função de penalidade $P_{ne}(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ no ítem a) é contínua e diferenciável em todo o seu domínio, é não negativa e anula-se apenas se a restrição do problema (3.5) for satisfeita. Neste caso, prova-se que é necessário um valor infinito do parâmetro de penalidade γ para que o conjunto de minimizadores de E_{ne} coincida com o conjunto solução do problema (3.5).

Por outro lado, a função de penalidade $P_e(\mathbf{x}) = \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_1$ do ítem b) também é contínua em todo o seu domínio, é não negativa, anula-se apenas se a restrição do problema (3.5) for satisfeita, e P_e é uma função de penalidade exata [6]. Como P_e é exata, então existe um valor finito do parâmetro de penalidade $\gamma > 0$, tal que o conjunto de minimizadores da função E_e coincide com o conjunto solução do problema (3.5). No entanto, observe que P_e é não-diferenciável no conjunto viável $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ do problema (3.5).

Uma abordagem alternativa para resolver o problema (3.1), obtendo uma solução exata, é a minimização da função lagrangeana associada a este problema [49], que é uma função diferenciável. Sob o ponto de vista prático, a desvantagem desta abordagem é o elevado custo computacional, especialmente quando problemas de otimização de grande porte são considerados. A dimensão do problema de minimização da função lagrangeana é significativamente mais elevada, pois além das variáveis originais, os multiplicadores de Lagrange também precisam ser calculados. O aumento da dimensão do problema exige maiores recursos computacionais, como maior quantidade de memória para o armazenamento das variáveis, e maior poder de processamento para a computação dos multiplicadores de Lagrange.

3.2 Sistemas gradientes

Algoritmos tipo gradiente são os sistemas mais usados como referência de síntese de algoritmos para resolver problemas de otimização sem restrições na forma

$$\text{minimizar } E(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \tag{3.6}$$

sendo que $E : \mathbb{R}^n \to \mathbb{R}$ é uma função de classe C^1.

O problema (3.6) pode ser associado a um sistema dinâmico da forma (2.9), com $\mathbf{g}(\mathbf{x}) = -\mathbf{M}\nabla E(\mathbf{x})$. Neste caso, este sistema dinâmico é conhecido como *sistema gradiente*, e é dado por:

$$\dot{\mathbf{x}}(t) = -\mathbf{M}\nabla E(\mathbf{x}(t)), \tag{3.7}$$

no qual $\mathbf{M}$ é uma matriz diagonal positiva, também conhecida como matriz de aprendizado do sistema. A matriz $\mathbf{M}$ é introduzida como um fator de escala de tempo de integração. Se a função E for não-crescente ao longo das trajetórias de (3.7) e limitada inferiormente é chamada *função de energia* do sistema.

Se os pontos de equilíbrio do sistema dinâmico (3.7) existirem, estes são atingidos quando $\nabla E(\mathbf{x}) = \mathbf{0}$. Pela definição 2.2, o ponto $\mathbf{x}^*$ no qual o gradiente de E é nulo, é um ponto crítico desta função. Logo, os pontos de equilíbrio de (3.7) correspondem a pontos críticos de máximo, mínimo ou de inflexão da função E.

Uma propriedade fundamental do vetor gradiente de uma função E, é que a direção deste vetor é a que proporciona o máximo incremento à função E. Esta propriedade é mostrada através da derivada direcional de E na direção de um vetor unitário $\mathbf{u} \in \mathbb{R}^n$ no ponto $\mathbf{x}_0$, dada por

$$D_{\mathbf{u}}E(\mathbf{x}_0) = \|\nabla E(\mathbf{x}_0)\|_2 \cos\theta,$$

em que θ é o ângulo entre o vetor unitário $\mathbf{u}$ e o gradiente de E no ponto $\mathbf{x}_0$.

A direção de maior incremento é quando o ângulo θ é zero, isto é, quando os vetores $\mathbf{u}$ e $\nabla E(\mathbf{x}_0)$ estão na mesma direção. Por outro lado, a derivada direcional na direção de $\mathbf{u}$ é mínima quando θ é igual a π radianos. Neste caso, os vetores $\mathbf{u}$ e $\nabla E(\mathbf{x}_0)$ têm sentidos opostos, fazendo com que a direção do gradiente seja a que proporciona o máximo decréscimo à função E.

Observe, em (3.7), que os vetores $\dot{\mathbf{x}}$ e $\nabla E(\mathbf{x})$ têm sentidos contrários. Logo, o movimento das trajetórias ocorre no sentido contrário àquele que proporciona a maior taxa de incremento em E, eventualmente atingindo um ponto de equilíbrio, caso exista, que é um ponto critico de E. Se E for convexa em todo $\mathbb{R}^n$, então o ponto de equilíbrio atingido é um ponto de mínimo global.

Por outro lado, caso a função E seja obtida por um método de penalização exata, então esta função é não-diferenciável e, conseqüentemente, o sistema gradiente (3.7) possui o segundo membro descontínuo. A minimização de funções convexas não-diferenciáveis utilizando sistemas gradientes é discutida na próxima seção.

3.3 Sistemas gradientes com segundo membro descontínuo

Consideremos o problema de otimização (3.4), em que a função objetivo E é convexa em todo $\mathbb{R}^n$, e não-diferenciável em um conjunto Δ de medida nula, no sentido de Lebesgue. Conseqüentemente, o segundo membro do sistema gradiente (3.7) é descontínuo no conjunto Δ, e suas soluções não existem no sentido usual.

Neste caso, as soluções do sistema gradiente (3.1) existem no sentido de Filippov: são funções vetoriais $\mathbf{x}(t)$ absolutamente contínuas, tais que, para quase todo $t \in [t_0, t_f]$, a inclusão diferencial (3.8) é satisfeita.

$$\dot{\mathbf{x}}(t) \in -\partial E(\mathbf{x}(t)), \tag{3.8}$$

sendo que o conjunto $\partial E(\mathbf{x}))$ é o subdiferencial de E em $\mathbf{x}(t)$, definido na página 15.

Nos pontos em que E é diferenciável, as derivadas parciais de E existem no sentido usual, e o subdiferencial de E possui apenas um elemento, que é o próprio gradiente desta função nestes pontos. Neste caso, a solução $\mathbf{x}(t)$ da inclusão diferencial (3.8) satisfaz a equação (3.7), no sentido usual.

Devido à convexidade de E em todo $\mathbb{R}^n$, pelo teorema 2.10, um vetor $\mathbf{x}^*$ é um minimizador global de E se e somente se

$$\mathbf{0} \in \partial E(\mathbf{x}^*). \tag{3.9}$$

O ponto $\mathbf{x}^*$ é um ponto de equilíbrio da inclusão diferencial (3.8), que é uma solução do problema de otimização sem restrições (3.6).

A convergência dos sistemas gradientes propostos neste trabalho é analisada utilizando a forma Persidskii destes sistemas. A próxima seção é dedicada à análise de convergência de uma classe de sistemas Persidskii com segundo membro descontínuo.

3.4 Sistemas Persidskii

Sistemas Persidskii e funções do tipo diagonal foram introduzidos em [19], onde a estabilidade absoluta destes sistemas é provada utilizando uma função de Lyapunov do tipo diagonal. Estes sistemas também foram analisados em [50], e em [51] uma classe de sistemas Persidskii com segundo membro descontínuo foi analisada. Bhaya e Kaszkurewicz [52] utilizaram sistemas Persidskii para provar que o ponto de equilíbrio de uma rede de Hopfield [9] é globalmente assintoticamente estável. A seguir, apresentamos definições de funções tipo diagonal e sistemas Persidskii.

Definição 3.3 (Função tipo diagonal). Uma função $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n, \mathbf{x} \mapsto \mathbf{f}(\mathbf{x})$ é dita do tipo diagonal se cada f_i é uma função apenas do argumento x_i, isto é,

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1) \\ \vdots \\ f_n(x_n) \end{pmatrix},$$

em que $\mathbf{x} \in \mathbb{R}^n$ é o vetor coluna dado por $\mathbf{x} = (x_1, \ldots, x_n)^T$.

Definição 3.4 (Sistema Persidskii [19]). Um sistema de equações diferenciais ordinárias é dito do tipo Persidskii se for da forma

$$\dot{x_i}(t) = \sum_{j=1}^{n} b_{ij} f_j(x_j(t)) \qquad (i = 1, \ldots, n), \tag{3.10}$$

sendo que, para cada j, a função $f_j : \mathbb{R} \to \mathbb{R}$ é contínua e pertence à classe de não-linearidades de setor infinito

$$S = \{f_j : \xi f_j(\xi) > 0;\ f_j(0) = 0;\ \int_0^{x_j} f_j(\tau) d\tau \to \infty \text{ quando } |x_j| \to \infty\}. \tag{3.11}$$

Sistema Persidskii generalizado

O sistema Persidskii generalizado é definido a partir do sistema Persidskii (3.10), em que a função $\mathbf{f}$ é do tipo diagonal, consistindo de uma parte linear e uma parte não linear com descontinuidades, isto é:

$$\dot{\mathbf{x}}(t) = -\mathbf{B}\mathbf{f}(\mathbf{x}(t)), \tag{3.12}$$

sendo que $\mathbf{x} = (\bar{\mathbf{x}}^T, \hat{\mathbf{x}}^T)^T$, $\bar{\mathbf{x}} \in \mathbb{R}^p$, $\hat{\mathbf{x}} \in \mathbb{R}^q$, $\mathbf{f}(\mathbf{x}) = (\bar{\mathbf{x}}^T, \mathbf{g}^T(\hat{\mathbf{x}}))^T$, $\mathbf{B} \in \mathbb{R}^{n\times n}$, $n = p + q$. A função $\mathbf{g} : \mathbb{R}^q \to \mathbb{R}^q$ satisfaz as seguintes condições:

a) $\mathbf{g}(\hat{\mathbf{x}})$ é uma função contínua por partes do tipo diagonal:

$$\mathbf{g}(\hat{\mathbf{x}}) = (g_1(\hat{x}_1), \ldots, g_q(\hat{x}_q))^T, \tag{3.13}$$

b) $\hat{x}_i g_i(\hat{x}_i) \geq 0$, $i = 1, \ldots, q$;

c) $\mathbf{g}$ é descontínua em um conjunto de medida nula no sentido de Lebesgue, dado por

$$\Delta := \cup_{i=1}^{k} \Delta_i$$

d) $\int_0^{\hat{x}_i} g_i(\tau)d\tau \to \infty$ quando $|\hat{x}_i| \to \infty$, $i = 1, \ldots, q$.

Observação 1. Como $\mathbf{g}$ é contínua por partes, então $\mathbf{g}$ é limitada em qualquer intervalo, e como o conjunto em que a função $\mathbf{g}$ é descontínua é de medida nula, então cada componente g_i de $\mathbf{g}$ é integrável [53, Teorema 3-8].

Como o sistema (3.12) tem o segundo membro descontínuo, as soluções são definidas no sentido de Filippov. De acordo com a teoria de Filippov, as soluções do sistema Persidskii generalizado (3.12) são funções $\mathbf{x}(t)$ absolutamente contínuas que, para quase todo t em um intervalo, satisfazem a seguinte inclusão diferencial:

$$\dot{\mathbf{x}}(t) \in G(\mathbf{x}(t)), \tag{3.14}$$

sendo que que o conjunto G é o menor conjunto convexo fechado que contém todos os valores limites de $-\mathbf{B}\mathbf{f}(\mathbf{x}')$, quando $\mathbf{x}' \to \mathbf{x}$, em que $\mathbf{x}' \notin \Delta$. Com esta definição, denotamos

cada elemento do conjunto $G(\mathbf{x}(t))$ por

$$\dot{\mathbf{x}}(t) = -\mathbf{B}\boldsymbol{\xi}. \tag{3.15}$$

Caso as trajetórias estejam confinadas em alguma superfície de descontinuidade Δ_i, para quase todo $t \in [t_1, t_2]$, um movimento deslizante ocorre nesta superfície. Quando as trajetórias não estão confinadas à nenhuma superfície de descontinuidade Δ_i, a equação (3.12) é satisfeita no sentido usual, e o conjunto $G(\mathbf{x}(t))$ contém apenas um elemento.

Considere a seguinte função de Lyapunov do tipo Lure-Persidskii associada ao sistema Persidskii generalizado (3.12):

$$V(\mathbf{z}) = \frac{1}{2}(\bar{\mathbf{x}} - \bar{\mathbf{x}}^*)^T \mathbf{D}_{11}(\bar{\mathbf{x}} - \bar{\mathbf{x}}^*) + \sum_{i=1}^{q} d_{ii}^{(2)} \int_0^{\hat{x}_i} g_i(\tau) d\tau, \tag{3.16}$$

sendo que $\mathbf{z} = [(\bar{\mathbf{x}} - \bar{\mathbf{x}}^*)^T, \hat{\mathbf{x}}^T]^T$ e $\mathbf{x}^* = [\bar{\mathbf{x}}^{*T}, \mathbf{0}^T]^T \in \mathcal{N}(\mathbf{B})$, em que $\mathcal{N}(\mathbf{B})$ é o espaço nulo de $\mathbf{B}$, $d_{ii}^{(2)}$ são elementos diagonais de uma matriz diagonal positiva $\mathbf{D}_{22}$, e as matrizes $\mathbf{D}_{11}$ e $\mathbf{D}_{22}$ são os elementos bloco-diagonais de uma matriz diagonal por blocos $\mathbf{D}$.

Note que $V(\mathbf{z}) = 0$ se e somente se $\mathbf{z} = \mathbf{0}$, implicando em $\mathbf{x} = \mathbf{x}^*$. Por outro lado, a condição d) imposta à função $\mathbf{g}$, garante que V é radialmente ilimitada. Como, para cada i, g_i é integrável (ver observação 1), então a segunda parcela de V em (3.16) é absolutamente contínua [54, pag. 105]. Além disso, o primeiro termo no segundo membro de V é uma função continuamente diferenciável na variável $\bar{\mathbf{x}}$, e $\mathbf{z}(t)$ é absolutamente contínua. Logo, podemos concluir que $V(\mathbf{z}(t))$ admite derivadas em quase todo t.

Teorema 3.2. *Considere o sistema Persidskii generalizado* (3.12)*. Se existe uma matriz simétrica semi-definida positiva* $\mathbf{S}$ *e uma matriz bloco-diagonal definida positiva* $\mathbf{D}$ *tal que:*

$$\mathbf{D} = \left(\begin{array}{c|c} \mathbf{D}_{11} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{D}_{22} \end{array}\right),$$

em que $\mathbf{D}_{11}$ *é simétrica definida positiva,* $\mathbf{D}_{22}$ *é uma matriz diagonal positiva, tal que* $\mathbf{B} = \mathbf{SD}$*, então as trajetórias de* (3.12) *convergem globalmente para o conjunto invariante* $\mathcal{A} := \{\mathbf{x} : \boldsymbol{\xi} \in \mathcal{N}(\mathbf{B}), \mathbf{D}\boldsymbol{\xi} \in \partial V(\mathbf{z})\}$*, em que* $\mathcal{N}(\mathbf{B})$ *denota o espaço nulo da matriz* $\mathbf{B}$*.*

Prova: Considere a função de Lyapunov não-suave (3.16), associada ao sistema Persidskii generalizado (3.12). A derivada no tempo de (3.16) ao longo das trajetórias de (3.12) é dada por:

$$\dot{V}(\mathbf{z}) = (\mathbf{D}\boldsymbol{\xi})^T\dot{\mathbf{x}} = -\boldsymbol{\xi}^T\mathbf{DSD}\boldsymbol{\xi} + \mathbf{x}^{*T}\mathbf{DSD}\boldsymbol{\xi}, \quad \mathbf{D}\boldsymbol{\xi} \in \partial V(\mathbf{z}).$$

Como, por hipótese, $\mathbf{B} = \mathbf{SD}$, e $\mathbf{x}^* \in \mathcal{N}(\mathbf{B})$, então a segunda parcela do segundo membro da igualdade anterior é nula. Daí, a derivada de V toma a forma:

$$\dot{V} = -\boldsymbol{\xi}^T\mathbf{DSD}\boldsymbol{\xi}, \quad \mathbf{D}\boldsymbol{\xi} \in \partial V(\mathbf{z}). \tag{3.17}$$

Analisamos a derivada de V acima sob duas hipóteses:

i) $\mathbf{x}(t) \notin \Delta_i$, isto é, as trajetórias do sistema (3.12) não estão confinadas em nenhuma superfície de descontinuidade. Neste caso, as soluções de (3.12) existem no sentido usual, e o conjunto $\partial V(\mathbf{z})$ contém apenas o gradiente de V em $\mathbf{z}$, dado por $\nabla V(\mathbf{z}) = \mathbf{Df}(\mathbf{x})$. Neste caso, a derivada no tempo de V ao longo das trajetórias do sistema (3.12) é dada por

$$\dot{V} = -\mathbf{f}^T(\mathbf{x})\mathbf{DSDf}(\mathbf{x}).$$

Como $\mathbf{S}$ é semi-definida positiva, é imediato que $\dot{V} \leq 0$;

ii) $\mathbf{x}(t) \in \Delta_i$, para algum i, e $t \in [t_0, t_f]$, isto é, as trajetórias do sistema (3.12) estão confinadas em alguma superfície de descontinuidade durante um intervalo de tempo. Logo, $\dot{\mathbf{x}}(t)$ não existe no sentido usual, e o vetor $\mathbf{x}(t)$ satisfaz $\dot{\mathbf{x}}(t) = -\mathbf{SD}\boldsymbol{\xi} \in G(\mathbf{x}(t))$, para quase todo $t \in [t_0, t_f]$. Como $\mathbf{S}$ é semi-definida positiva, por (3.17), é imediato que $\dot{V} \leq 0$.

Por (3.15), e dado que $\mathbf{B} = \mathbf{SD}$, note que $\dot{V} \equiv 0$ se e somente se, o vetor $\mathbf{x}$ satisfaz

$$\mathbf{B}\boldsymbol{\xi} = \mathbf{0} \in G(\mathbf{x}),$$

isto é, $\mathbf{x} \in \mathcal{A}$. Logo, $\mathcal{A}$ é um conjunto invariante.

Utilizando os ítens i) e ii) e o teorema de LaSalle para sistemas não-suaves [35], conclui-se que as trajetórias do sistema Persidskii generalizado (3.12), convergem para o conjunto

invariante $\mathcal{A}$, para quaisquer condições iniciais. ■

Este resultado é usado no capítulo 5 para provar a convergência de sistemas gradientes para a resolução de problemas de programação quadrática com restrições lineares, aplicados ao treinamento de SVMs para separação linear e não linear de duas classes.

Corolário 3.2.1. *Se as matrizes* $\mathbf{S}$ *e* $\mathbf{D}$ *são simétricas definidas-positivas e* $f_i(x_i) \in [a_i, b_i]$ *para todo* $x_i \in \mathbb{R}^n$*, então as trajetórias do sistema* (3.12) *convergem em tempo finito para o conjunto invariante* $\mathcal{A}$ *e permanecem neste conjunto.*

Prova: Considere a função de Lyapunov candidata (3.16). A derivada no tempo de (3.16) ao longo das trajetórias do sistema (3.12) é dada por:

$$\dot{V} = (\mathbf{D}\boldsymbol{\xi})^T\dot{\mathbf{x}} = -\boldsymbol{\xi}^T\mathbf{DSD}\boldsymbol{\xi}, \quad \mathbf{D}\boldsymbol{\xi} \in \partial V(\mathbf{z}).$$

Como $\mathbf{S}$ é simétrica definida positiva, usando o quociente de Rayleigh, é imediato que:

$$\dot{V} \leq -\lambda_{min}(\mathbf{DSD})\boldsymbol{\xi}^T\boldsymbol{\xi} \leq -\lambda_{min}(\mathbf{DSD})\mathbf{a}^T\mathbf{a}, \tag{3.18}$$

sendo que $\mathbf{a} := (a_1, \ldots, a_n)^T$ e $\lambda_{min}(\mathbf{DSD})$ é o menor autovalor da matriz $\mathbf{DSD}$. Dado que as matrizes $\mathbf{D}$ e $\mathbf{S}$ são simétricas definidas positivas, então $\lambda_{min}(\mathbf{DSD}) > 0$, isto é, $\dot{V} \leq -\varepsilon$, em que $\varepsilon = \lambda_{min}(\mathbf{DSD})\mathbf{a}^T\mathbf{a}$.

Utilizando os mesmos argumentos da prova do teorema 3.2, concluímos que $\mathcal{A}$ é um conjunto invariante.

Conseqüentemente, por (3.18), pelas observações acima, e pelo teorema de LaSalle para sistemas não-suaves, as trajetórias do sistema Persidskii generalizado (3.12) convergem para o conjunto solução $\mathcal{A}$. Precisamos mostrar que o tempo máximo T para que $V \equiv 0$ é finito. Por (3.18), temos que, para quase todo t em um intervalo:

$$V(\mathbf{z}(t)) \leq V(\mathbf{z}(t_0)) - \varepsilon t,$$

Seja $T \geq t_0$ o instante para o qual $V(\mathbf{z}(T)) = 0$. Então, pela desigualdade anterior, tem-se que

$$T \leq \frac{V(\mathbf{z}(t_0))}{\varepsilon},$$

como $\varepsilon \neq 0$, dado um instante inicial t_0, o segundo membro desta desigualdade assume um valor finito. Logo, a convergência para o conjunto $\mathcal{A}$ é em tempo finito. ■

As análises de convergência dos sistemas gradientes estudados neste trabalho são feitas através da forma Persidskii destes sistemas, juntamente e funções de Lyapunov do tipo diagonal ou utilizando diretamente o teorema 3.2.

3.5 Análise de convergência de sistemas gradientes através da forma Persidskii

As análises de convergência dos os sistemas gradientes propostos neste trabalho são feitas utilizando a forma Persidskii (3.12) do sistema gradiente correspondente, e funções de Lyapunov do tipo diagonal ou Lure-Persidskii. Além da simplicidade das provas dos resultados, esta estratégia de análise usualmente resulta em condições de convergência tratáveis, independentes do ajuste de parâmetros.

A forma Persidskii associada aos sistemas gradientes propostos neste trabalho em geral possuem dimensão maior que a do sistema original, por isso é preciso mostrar que a solução do sistema original é a mesma solução do sistema Persiskii associado. Sem perda de generalidade, analisamos o problema geral de otimização (3.1) com restrições lineares de igualdade. Neste caso, o conjunto viável Ω é dado por

$$\Omega := \{\mathbf{x} : \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}\}, \tag{3.19}$$

em que $\mathbf{A}$ é de dimensão $m \times n$ e tem posto completo por linhas, e $\mathbf{b}$ é um vetor coluna de dimensão m. Observe que a generalidade não é perdida, pois restrições de desigualdade podem ser escritas na forma acima através da introdução de variáveis de folga.

Assumimos que o vetor gradiente ∇f da função objetivo do problema (3.1) é do tipo diagonal, isto é,

$$\nabla f(\mathbf{x}) = (\nabla_{x_1} f(x_1), \ldots, \nabla_{x_n} f(x_n))^T,$$

e pertence à classe de não-linearidades de setor infinito S, definida em (3.11).

As funções objetivo cujo gradiente é do tipo diagonal, incluem as funções lineares e quadráticas, que são consideradas neste trabalho. Utilizando o método de penalização exata,

o problema de otimização sem restrições associado ao problema com restrições (3.1), com o conjunto viável dado em (3.19), é

$$\text{minimizar}_{\mathbf{x}}\, E(\mathbf{x}) = f(\mathbf{x}) + \rho\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1, \tag{3.20}$$

sendo que ρ é o parâmetro de penalidade. Note que a função de penalidade $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1$ é não-diferenciável.

Seja o vetor de resíduos definido por $\mathbf{r} := \mathbf{A}\mathbf{x} - \mathbf{b}$. Um subgradiente da função objetivo não-diferenciável E do problema (3.20) é dado por

$$\mathbf{e} = \nabla f(\mathbf{x}) + \rho\mathbf{A}^T\text{sgn}(\mathbf{r}), \quad \mathbf{e} \in \partial E(\mathbf{x}) \tag{3.21}$$

sendo que cada componente do vetor sgn é definido em (3.22) e seu gráfico é mostrado na figura 3.1.

$$\text{sgn}(a) \begin{cases} := -1 \text{ se } a < 0 \\ \in [-1, 1] \text{ se } a = 0 \\ := 1 \text{ se } a > 0 \end{cases} \tag{3.22}$$

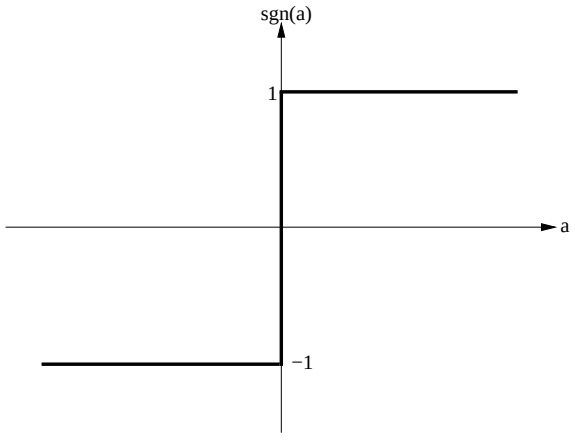


Figura 3.1: ráfico de sgn, que atua na restrição de igualdade do problema de otimização

Note que sgn é um mapeamento de $\mathbb{R}^m$ no conjunto definido em (3.22). Com esta definição, sgn é um subgradiente da função $\|\mathbf{r}\|_1$. Observe que, de acordo com (3.21), o

vetor de resíduos $\mathbf{r}$ é separável e a não-linearidade sgn é do tipo diagonal, isto é

$$\text{sgn}(\mathbf{r}) = (\text{sgn}_1^T(r_1), \ldots, \text{sgn}_m^T(r_m))^T.$$

O sistema gradiente não-suave $\dot{\mathbf{x}} = -\mathbf{Me}$, associado ao problema de otimização (3.21), também possui as variáveis separáveis e é dado por

$$\dot{\mathbf{x}} = -\mathbf{M}\nabla f(\mathbf{x}) - \rho\mathbf{M}\mathbf{A}^T\text{sgn}(\mathbf{Ax} - \mathbf{b}). \tag{3.23}$$

Dizemos que $\mathbf{x}^*$ é um ponto de equilíbrio do sistema gradiente (3.23), quando $\mathbf{x}^*$ for um ponto de equilíbrio da inclusão diferencial $\dot{\mathbf{x}} \in -\partial E(\mathbf{x})$, isto é, $\mathbf{0} \in \partial E(\mathbf{x}^*)$.

Para construir a forma Persidskii do sistema gradiente (3.23), inicialmente pré-multiplica-se o sistema (3.23) pela matriz $\mathbf{A}$. Como $\dot{\mathbf{r}} = \mathbf{A}\dot{\mathbf{x}}$ é a derivada no tempo do vetor de resíduos, obtemos o seguinte sistema

$$\dot{\mathbf{r}} = -\mathbf{AM}\nabla f(\mathbf{x}) - \rho\mathbf{AMA}^T\text{sgn}(\mathbf{Ax} - \mathbf{b}). \tag{3.24}$$

Sem perda de generalidade, consideramos nesta análise $\mathbf{M} = \mathbf{I}$. Escrevendo os sistemas (3.23) e (3.24) em notação vetorial, obtemos o sistema (3.25) na forma Persidskii, associado ao sistema gradiente (3.23).

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{r}} \end{bmatrix} = - \left[\begin{array}{c|c} \mathbf{I} & \mathbf{A}^T \\ \hline \mathbf{A} & \mathbf{AA}^T \end{array}\right] \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \rho\mathbf{I} \end{array}\right] \begin{bmatrix} \nabla f(\mathbf{x}) \\ \text{sgn}(\mathbf{r}) \end{bmatrix}. \tag{3.25}$$

O sistema Persidskii (3.25) possui dimensão $m + n$, ao passo que o sistema original (3.23) possui dimensão n. É necessário analisar as implicações da pré-multiplicação do sistema (3.23) pela matriz $\mathbf{A}$, e do aumento da dimensão do sistema. Precisamos mostrar que, se as trajetórias do sistema Persidskii convergem para um conjunto invariante Λ, então as trajetórias do sistema gradiente (3.23) convergem para um ponto de equilíbrio, e vice-versa.

i) Vamos mostrar que quando as trajetórias do sistema Persidskii convergem para um conjunto invariante Λ, então as trajetórias do sistema gradiente convergem para um ponto de

equilíbrio $\mathbf{x}^*$. Seja a seguinte função de Lyapunov candidata:

$$V((\mathbf{x},\mathbf{r})) = f(\mathbf{x}) - f(\mathbf{x}^*) + \rho\sum_{i=1}^{m}\int_0^{r_i} \text{sgn}(\tau)d\tau = f(\mathbf{x}) - f(\mathbf{x}^*) + \rho\|\mathbf{r}\|_1. \tag{3.26}$$

Note que, como a função $\|\mathbf{r}\|_1$ é Lipschitz contínua em $\mathbf{r}$, então é absolutamente contínua nesta variável, e como $\mathbf{r}(t)$ também é absolutamente contínua, então a função $\|\mathbf{r}(t)\|_1$ admite derivadas em quase todo t [54]. Dado que f é de classe C^1, e $\mathbf{x}(t)$ é absolutamente contínua então a função V admite derivada em quase todo t.

A derivada no tempo de V ao longo das trajetórias de (3.25) é dada por

$$\dot{V}(\mathbf{x},\mathbf{r}) = \mathbf{v}^T \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{r}} \end{bmatrix}, \quad \mathbf{v} \in \partial V(\mathbf{x},\mathbf{r})$$

sendo que $\mathbf{v} = [\nabla^T f(\mathbf{x}), \rho\text{sgn}^T(\mathbf{r})]^T$. Utilizando (3.25), a derivada de V é dada por

$$\dot{V}(\mathbf{x},\mathbf{r}) = -\begin{bmatrix}\nabla^T f(\mathbf{x}) & \text{sgn}^T(\mathbf{r})\end{bmatrix} \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \rho\mathbf{I} \end{array}\right] \left[\begin{array}{c|c} \mathbf{I} & \mathbf{A}^T \\ \hline \mathbf{A} & \mathbf{A}\mathbf{A}^T \end{array}\right] \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \rho\mathbf{I} \end{array}\right] \begin{bmatrix}\nabla f(\mathbf{x}) \\ \text{sgn}(\mathbf{r})\end{bmatrix}. \tag{3.27}$$

Seja $\mathbf{B}$ a matriz de blocos no segundo membro de (3.27):

$$\mathbf{B} = \mathbf{S}\mathbf{D},$$

sendo que as matrizes $\mathbf{S}$ e $\mathbf{D}$ são

$$\mathbf{S} = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{A}^T \\ \hline \mathbf{A} & \mathbf{A}\mathbf{A}^T \end{array}\right] \quad \text{e} \quad \mathbf{D} = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \\ \hline \mathbf{0} & \rho\mathbf{I} \end{array}\right]$$

Como $\mathbf{D}$ é uma diagonal positiva e a matriz $\mathbf{S}$ é simétrica semi-definida positiva, então a matriz $\mathbf{B}$ satisfaz as condições do teorema 3.2. O segundo membro de (3.27) é igual a zero no espaço nulo de $\mathbf{B}$, denotado por $\mathcal{N}(\mathbf{B})$. Logo, pelo teorema de LaSalle para sistemas não-suaves [35], as trajetórias do sistema Persidskii convergem para o seguinte conjunto invariante:

$$\Lambda = \{\mathbf{x},\mathbf{r} : \mathbf{y} \in \mathcal{N}(\mathbf{B})\}, \tag{3.28}$$

sendo que $\mathbf{y} = (\nabla^T f(\mathbf{x}), \text{sgn}^T(\mathbf{r}))^T$. Os vetores $\mathbf{x}$ e $\mathbf{r}$ que pertencem ao conjunto Λ são tais

que

$$\dot{\mathbf{x}} = \mathbf{0} \in \partial E(\mathbf{x}) \quad \text{e} \quad \dot{\mathbf{r}} = \mathbf{0}.$$

Isto significa que o conjunto invariante Λ, para o qual as trajetórias do sistema Persidskii convergem, é a intersecção do conjunto dos pontos de equilíbrio do sistema gradiente (3.23) e do conjunto $\{\mathbf{r} : \dot{\mathbf{r}} = \mathbf{0}\}$. Portanto, a convergência das trajetórias do sistema Persidskii para o conjunto invariante Λ, implica na convergência das trajetórias do sistema gradiente para um ponto de equilíbrio.

ii) Vamos mostrar que se as trajetórias do sistema gradiente convergem para um conjunto invariante $\bar{\Lambda}$, então as trajetórias do sistema Persidskii convergem para o conjunto invariante Λ. Observe que a função $V(\mathbf{x}, \mathbf{r})$ em (3.26) é a função de energia computacional E em (3.20), em que $\mathbf{r}$ é considerada como uma variável independente de $\mathbf{x}$. Considere agora a função E, dependente apenas da variável $\mathbf{x}$, como uma função de Lyapunov candidata associada ao sistema gradiente (3.23), cuja derivada no tempo é:

$$\dot{E}(\mathbf{x}) = \mathbf{e}^T \dot{\mathbf{x}} = -\|\mathbf{e}\|^2, \quad \mathbf{e} \in \partial E(\mathbf{x})$$

Observe que $\dot{E}(\mathbf{x}) \equiv 0$ se e somente se $\mathbf{e} \equiv \mathbf{0}$, implicando que $\dot{\mathbf{x}} \equiv \mathbf{0} \in \partial E(\mathbf{x})$. Portanto, o conjunto invariante procurado é o próprio conjunto dos pontos de equilíbrio do sistema gradiente (3.23):

$$\bar{\Lambda} = \{\mathbf{x} : \mathbf{0} \in -\partial E(\mathbf{x})\},$$

segue que, pelo teorema de LaSalle para sistemas não-suaves, as trajetórias do sistema gradiente (3.23) convergem para o conjunto invariante $\bar{\Lambda}$.

Por outro lado, como a derivada do vetor de resíduos é $\dot{\mathbf{r}} = \mathbf{A}\dot{\mathbf{x}}$, se $\mathbf{x} \in \bar{\Lambda}$, então $\dot{\mathbf{r}} = \mathbf{0}$. Logo, se $\mathbf{x}^* \in \bar{\Lambda}$ é um ponto de equilíbrio do sistema gradiente (3.23), então o vetor $[\mathbf{x}^*, \mathbf{r}^*]^T \in \Lambda$, $\mathbf{r}^* = \mathbf{A}\mathbf{x}^* - \mathbf{b}$. Portanto, a convergência das trajetórias do sistema gradiente (3.23) para o conjunto invariante $\bar{\Lambda}$, implica na convergência das trajetórias do sistema Persidskii para o conjunto invariante Λ.

Portanto, pelos ítens i) e ii) concluímos que o sistema gradiente (3.23) e o sistema Persidskii (3.25) são equivalentes, no sentido que a convergência das trajetórias do sistema Persidskii para um conjunto invariante, que contém um ponto de equilíbrio do sistema gradiente, implicam na convergência das trajetórias do sistema gradiente para o mesmo ponto de

equilíbrio, e vice-versa. Este resultado valida a utilização da forma Persidskii dos sistemas gradientes nas análises de convergência.

3.6 Programação linear

No caso em que a função f em (3.1) é linear e o conjunto Ω é descrito por funções lineares, esta classe de problemas de otimização recebe o nome de *problema de programação linear* (PPL).

Diversos métodos para resolução de problemas de programação linear estão disponíveis, o mais popular é o método do SIMPLEX [55]. No entanto, este método precisa de um ponto inicial pertencente ao conjunto viável. Os métodos de pontos interiores para programação linear ganharam popularidade. Entretanto, estes métodos apresentam problemas numéricos quando os resultados produzidos ao longo das iterações aproximam-se da fronteira do conjunto viável [56].

A seguir, apresentamos resultados de convergência para três variantes de problemas de programação linear. A análise completa e as provas dos resultados estão disponíveis em [20].

As análises de convergência dos sistemas gradientes para resolução de problemas de programação linear apresentadas em [20], são baseadas na análise de Chong et al. [15]. A análise de convergência em [15] resulta em limites inferiores que os parâmetros de penalidade têm que satisfazer para que a convergência em tempo finito para a solução do problema de programação linear em questão seja garantida. Entretanto, estas condições são dependentes das trajetórias do sistema. Por outro lado, os limites inferiores obtidos em [21] são constantes e dependem apenas dos parâmetros do problema de programação linear. A mesma estratégia de análise é utilizada no capítulo 4, em que um circuito k-winners-take-all (KWTA) é formulado a partir de um problema de programação linear.

3.6.1 Forma canônica I

A forma canônica do problema de programação linear é dada por:

$$\begin{aligned} &\text{minimizar } \mathbf{c}^T\mathbf{x} \\ &\text{sujeito a } \mathbf{A}\mathbf{x} \geq \mathbf{b}, \end{aligned} \tag{3.29}$$

onde $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, com posto$(\mathbf{A}) = m$.

Este problema é resolvido utilizando o método da função de penalidade, obtendo o seguinte problema de otimização sem restrições:

$$\text{minimizar } E(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \kappa \sum_{i=1}^{m} \min(0, \mathbf{a}_i\mathbf{x} - \mathbf{b}_i), \tag{3.30}$$

em que $\mathbf{a}_i$ é a i-ésima linha da matriz $\mathbf{A}$ e $\kappa > 0$ é o parâmetro de penalidade.

Note que a função objetivo E em (3.30) é composta pela função objetivo do problema original (3.29), mais um termo de penalidade, associado à restrição do problema. Note também que o termo de penalidade é não-diferenciável na fronteira do conjunto viável do PPL (3.29).

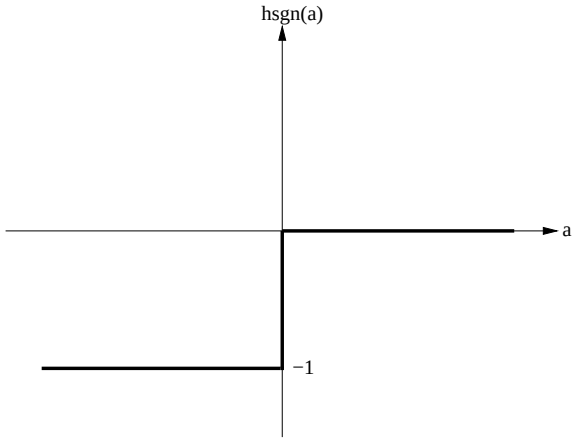


Figura 3.2: Gráfico de hsgn, que está associada às restrições de desigualdade do problema de otimização

Como a função objetivo E do problema (3.30) é não-diferenciável no conjunto $\Delta := \{\mathbf{x} : \mathbf{Ax} - \mathbf{b} = \mathbf{0}\}$, então as derivadas de E não existem no sentido usual. Um subgradiente desta função é dado por

$$\mathbf{e} = \mathbf{c} + \kappa \mathbf{M}\mathbf{A}^T \text{hsgn}(\mathbf{r}), \quad \mathbf{e} \in \partial E(\mathbf{x})$$

sendo que $\mathbf{r} := \mathbf{Ax} - \mathbf{b}$ é o vetor de resíduos, e cada componente do vetor hsgn é definido em (3.31) e seu gráfico é mostrado na figura 3.2. O vetor hsgn, é um subgradiente da função de penalidade $\sum_{i=1}^{p} \min(0, r_i)$.

$$\text{hsgn}(a)\begin{cases} := -1 \text{ se } a < 0 \\ \in [-1, 0] \text{ se } a = 0 \\ := 0 \text{ se } a > 0 \end{cases} \tag{3.31}$$

O problema de otimização (3.30), é resolvido através do sistema gradiente $\dot{\mathbf{x}} = -\mathbf{Me}$, dado por:

$$\dot{\mathbf{x}} = -\mathbf{Mc} - \kappa \mathbf{M}\mathbf{A}^T \text{hsgn}(\mathbf{r}), \tag{3.32}$$

sendo que $\mathbf{M}$ é uma diagonal positiva, conhecida como matriz de aprendizado.

A forma Persidskii deste sistema é obtida através da pré-multiplicação do mesmo pela matriz $\mathbf{A}$, obtendo:

$$\dot{\mathbf{r}} = -\mathbf{AMc} - \kappa \mathbf{AMA}^T \text{hsgn}(\mathbf{r}), \tag{3.33}$$

observe que esta é uma forma Persidskii deslocada pelo vetor constante $\mathbf{AMc}$.

A função de Lyapunov do tipo diagonal associada à equação anterior é

$$V(\mathbf{r}) = \kappa \sum_{i=1}^{m} \int_0^{r_i} \text{hsgn}(\tau) d\tau, \tag{3.34}$$

Denotando o conjunto viável do PPL (3.29) por $\Omega := \{\mathbf{r}(\mathbf{x}) : \mathbf{r}(\mathbf{x}) \geq \mathbf{0}\}$, a função V dada em (3.34), possui as seguintes características:

(i) $V(\mathbf{r}) > 0$, se $\mathbf{r} \notin \Omega$;

(ii) $V(\mathbf{r}) \equiv 0$, se $\mathbf{r} \in \Omega$.

A função V é não-nula apenas fora do conjunto viável do PPL. Utilizando a forma Persidskii (3.33) do sistema (3.32) e a função V, dada em (3.34), é possível obter condições suficientes que κ deve satisfazer para garantir a convergência para o conjunto Ω. Este resultado é formalmente estabelecido no teorema 3.3.

Teorema 3.3. *Considere o sistema gradiente* (3.32) *e a função de Lyapunov candidata* (3.34). *Se a matriz* $\mathbf{A}$ *tem posto por linhas completo e* κ *satisfaz:*

$$\kappa > \frac{\max(\mathbf{AMc})}{\lambda_{\min}(\mathbf{AMA}^T)}, \tag{3.35}$$

sendo que $\max(\mathbf{AMc})$ *é o maior componente do vetor* $\mathbf{AMc}$*. Então as trajetórias de* (3.32) *convergem para o conjunto solução de* (3.29) *em tempo finito e permanecem neste conjunto indefinidamente.*

Observação 2. Observe que a função V em (3.32) não é radialmente ilimitada. No entanto este fato não afeta o resultado estabelecido pelo teorema 3.3, pois o conjunto viável do problema é $\{\mathbf{x} : \mathbf{Ax} \geq \mathbf{b}\}$, e o segundo termo no segundo membro do sistema gradiente (3.32) é nulo quando $\mathbf{x}$ atinge este conjunto.

3.6.2 Forma canônica II

O problema de programação linear na forma canônica II é dado por:

$$\begin{aligned} &\text{minimizar } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{sujeito a } \mathbf{Ax} - \mathbf{b} \geq \mathbf{0} \\ &\mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.36}$$

sendo que $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m < n$, $posto(A) = m$, $\mathbf{c} \in \mathbb{R}^{n\times 1}, \mathbf{x} \in \mathbb{R}^n$ e $\mathbf{b} \in \mathbb{R}^m$.

Este problema é resolvido do mesmo modo que o problema na forma canônica I, apresentado na seção anterior. O problema de otimização sem restrições associado é dado por:

$$E(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \gamma \sum_{j=1}^{n} \min(0, x_j) - \rho \sum_{i=1}^{m} \min(0, r_i), \tag{3.37}$$

sendo que $\gamma > 0$ e $\rho > 0$ são os parâmetros de penalidade. Neste caso, a função objetivo do problema apresenta um termo de penalidade adicional, que está associado à restrição de sinal sobre a variável de projeto $\mathbf{x}$.

O sistema gradiente associado ao problema sem restrições (3.37) é

$$\dot{\mathbf{x}} = -\mathbf{c} - \gamma \text{hsgn}(\mathbf{x}) - \rho\mathbf{A}^T\text{hsgn}(\mathbf{r}), \tag{3.38}$$

em que $\gamma, \rho > 0$ são os parâmetros de penalidade.

A forma Persidskii deste sistema é obtida aumentando-se o mesmo com a derivada no tempo do vetor de resíduos, obtida através da pré-multiplicação da equação anterior pela

matriz $\mathbf{A}$, resultando em:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{r}} \end{pmatrix} = - \begin{pmatrix} \mathbf{c} \\ \mathbf{Ac} \end{pmatrix} - \begin{pmatrix} \mathbf{I}_n & \mathbf{A}^T \\ \mathbf{A} & \mathbf{A}\mathbf{A}^T \end{pmatrix} \begin{pmatrix} \gamma \text{ hsgn}(\mathbf{x}) \\ \rho \text{ hsgn}(\mathbf{r}) \end{pmatrix} \tag{3.39}$$

A função de Lyapunov do tipo diagonal associada ao sistema Persidskii anterior é

$$V(\mathbf{x}, \mathbf{r}) = \gamma \sum_{j=1}^{n} \int_0^{x_j} \text{hsgn}(\tau) d\tau + \rho \sum_{i=1}^{m} \int_0^{r_i} \text{hsgn}(\tau) d\tau, \tag{3.40}$$

esta função possui as mesmas características que a função V dada em (3.34), o que permite obter condições suficientes que os parâmetros ρ e γ devem satisfazer para garantir a convergência para o conjunto viável do PPL (3.36) e, conseqüentemente, para a solução deste problema. Este resultado é formalizado pelo teorema 3.4.

Teorema 3.4. *Considere o problema de programação linear* (3.36)*. Se a matriz* $\mathbf{A}$ *tem posto por linhas completo e os ganhos* γ *e* ρ *satisfazem as seguintes desigualdades*

$$\begin{aligned} \gamma &> \|\mathbf{c}\|_2 \\ \rho &> \frac{1}{\sigma_{min}(\mathbf{A})}(\mathbf{c} + 2\alpha\sqrt{n}), \end{aligned}$$

então as trajetórias de (3.38) *convergem para para o conjunto solução do problema* (3.29) *e permanecem neste conjunto indefinidamente.*

3.6.3 Forma padrão

A forma padrão do problema de programação linear é dada por:

$$\begin{aligned} &\text{minimizar } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{sujeito a } \mathbf{Ax} - \mathbf{b} = \mathbf{0} \\ &\mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.41}$$

onde $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $m < n$ e $posto(\mathbf{A}) = m$.

Utilizando o método da função de penalidade, obtemos o seguinte problema de otimiza-

ção sem restrições:

$$\text{minimizar } E(\mathbf{x}) = \mathbf{c^T x} + \rho\|\mathbf{r}\|_1 + \gamma \sum_{j=1}^{n} \min(0, x_j) \tag{3.42}$$

O sistema gradiente que resolve o problema anterior é dado por:

$$\dot{\mathbf{x}} = -\mathbf{c} - \gamma \text{hsgn}(\mathbf{x}) - \rho \mathbf{A}^T \text{sgn}(\mathbf{r}). \tag{3.43}$$

Por um procedimento análogo ao utilizado para o PPL na forma canônica II, a forma Persidskii do sistema anterior é

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{r}} \end{pmatrix} = - \begin{pmatrix} \mathbf{c} \\ \mathbf{Ac} \end{pmatrix} - \begin{pmatrix} \mathbf{I}_n & \mathbf{A}^T \\ \mathbf{A} & \mathbf{A}\mathbf{A}^T \end{pmatrix} \begin{pmatrix} \gamma \text{ hsgn}(\mathbf{x}) \\ \rho \text{ sgn}(\mathbf{r}) \end{pmatrix} \tag{3.44}$$

A função de Lyapunov tipo diagonal associada ao sistema Persidskii (3.44) é

$$V(\mathbf{x}, \mathbf{r}) = \gamma \sum_{j=1}^{n} \int_0^{x_j} \text{hsgn}(\tau) d\tau + \rho \sum_{i=1}^{m} \int_0^{r_i} \text{sgn}(\tau) d\tau. \tag{3.45}$$

Prova-se, utilizando o sistema Persidskii (3.45), que o teorema 3.4 também é válido para o sistema gradiente (3.43) [20].

3.7 Programação quadrática

Problemas de programação quadrática com restrições lineares podem ser resolvidos através de sistemas gradientes. Considere o seguinte problema geral de otimização quadrática com restrições lineares:

$$\begin{aligned} &\text{minimizar } \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} \\ &\text{sujeito a } \mathbf{Ax} = \mathbf{b} \\ &\mathbf{Hx} \geq \mathbf{c} \end{aligned} \tag{3.46}$$

sendo que $\mathbf{Q} \in \mathbb{R}^{n\times n}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{H} \in \mathbb{R}^{p\times n}$, $\mathbf{c} \in \mathbb{R}^p$ e $\mathbf{x} \in \mathbb{R}^n$ é a variável de decisão do problema.

Utilizando o método da função de penalidade, o seguinte problema de otimização sem

restrições é associado ao problema (3.46):

$$\text{minimizar } E(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} - \gamma\sum_{i=1}^{p}\min(0, \mathbf{h}_i^T\mathbf{x} - c_i) + \rho\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1, \tag{3.47}$$

em que $\mathbf{h}_i$ denota a i-ésima linha da matriz $\mathbf{H}$ e os escalares γ e ρ os *parâmetros de penalidade*.

Note que a função E em (3.47) é composta pela função objetivo do problema (3.46) mais dois termos de penalidade, cada um associado a um conjunto de restrições diferente. Note também que os termos de penalidade são não-diferenciáveis na fronteira do conjunto viável do problema (3.46).

Um subgradiente da função objetivo E do problema (3.47) é dado por:

$$\mathbf{e} = \mathbf{Q}\mathbf{x} + \rho\mathbf{A}^T\,\text{sgn}(\mathbf{r}^{(1)}) + \gamma\mathbf{H}^T\text{hsgn}(\mathbf{r}^{(2)}), \quad \mathbf{e} \in \partial E(\mathbf{x}),$$

sendo que $\mathbf{r}^{(1)} := \mathbf{A}\mathbf{x} - \mathbf{b}$, $\mathbf{r}^{(2)} := \mathbf{H}\mathbf{x} - \mathbf{c}$.

O sistema gradiente $\dot{\mathbf{x}} = -\mathbf{M}\mathbf{e}$, associado ao problema (3.47) é dado por:

$$\dot{\mathbf{x}} = -\mathbf{M}[\mathbf{Q}\mathbf{x} + \rho\mathbf{A}^T\,\text{sgn}(\mathbf{r}^{(1)}) + \gamma\mathbf{H}^T\text{hsgn}(\mathbf{r}^{(2)})]. \tag{3.48}$$

A convergência do sistema (3.48) é provada escrevendo o sistema gradiente (3.48) na forma Persidskii generalizada (3.12), e utilizando o teorema 3.2. Sem perda de generalidade, assumimos que $\mathbf{M} = \mathbf{I}$.

Para obter a forma Persidskii do sistema gradiente (3.48), observe que as derivadas no tempo de $\mathbf{r}^{(1)}$ e $\mathbf{r}^{(2)}$ são dadas por $\dot{\mathbf{r}}^{(1)} = \mathbf{A}\dot{\mathbf{x}}$ e $\dot{\mathbf{r}}^{(2)} = \mathbf{H}\dot{\mathbf{x}}$, isto é:

$$\dot{\mathbf{r}}^{(1)} = -\mathbf{A}\mathbf{Q}\mathbf{x} - \rho\mathbf{A}\mathbf{A}^T\,\text{sgn}(\mathbf{r}^{(1)}) - \gamma\mathbf{A}\mathbf{H}^T\,\text{hsgn}(\mathbf{r}^{(2)}) \tag{3.49}$$

$$\dot{\mathbf{r}}^{(2)} = -\mathbf{H}\mathbf{Q}\mathbf{x} - \rho\mathbf{H}\mathbf{A}^T\,\text{sgn}(\mathbf{r}^{(1)}) - \gamma\mathbf{H}\mathbf{H}^T\,\text{hsgn}(\mathbf{r}^{(2)}) \tag{3.50}$$

Escrevendo o sistema aumentado (3.48)-(3.50) em notação matricial, a forma Persidskii

do sistema gradiente (3.48) é dada por (3.12), em que a matriz $\mathbf{B}$ pode ser fatorada na forma:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I} & \mathbf{A}^T & \mathbf{H}^T \\ \mathbf{A} & \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{H}^T \\ \mathbf{H} & \mathbf{H}\mathbf{A}^T & \mathbf{H}\mathbf{H}^T \end{pmatrix} \left(\begin{array}{c|cc} \mathbf{Q} & \mathbf{0}_{n\times n} & \mathbf{0}_{n\times p} \\ \hline \mathbf{0}_{m\times n} & \rho\mathbf{I}_{n\times n} & \mathbf{0}_{n\times p} \\ \mathbf{0}_{n\times n} & \mathbf{0}_{m\times n} & \gamma\mathbf{I}_{p\times p} \end{array} \right)$$

Este sistema satisfaz as condições do teorema 3.2, pois na fatoração acima podemos identificar:

$$\mathbf{S} = \begin{pmatrix} \mathbf{I} & \mathbf{A}^T & \mathbf{H}^T \\ \mathbf{A} & \mathbf{A}\mathbf{A}^T & \mathbf{A}\mathbf{H}^T \\ \mathbf{H} & \mathbf{H}\mathbf{A}^T & \mathbf{H}\mathbf{H}^T \end{pmatrix};$$

$$\mathbf{D}_{11} = \mathbf{Q};$$

$$\mathbf{D}_{22} = \begin{pmatrix} \rho\mathbf{I}_{n\times n} & \mathbf{0}_{n\times p} \\ \mathbf{0}_{m\times n} & \gamma\mathbf{I}_{p\times p} \end{pmatrix}.$$

A matriz $\mathbf{S}$ é simétrica semi-definida positiva, a matriz $\mathbf{Q}$ é, por hipótese, definida positiva e, como ρ e γ são escalares positivos, a matriz $\mathbf{D}_{22}$ é uma diagonal positiva. Portanto, as condições do teorema 3.2 estão todas satisfeitas, garantindo a convergência das trajetórias do sistema (3.48) para o conjunto solução do problema (3.46), para quaisquer parâmetros γ e ρ positivos. Este resultado é enunciado no teorema 3.5.

Teorema 3.5. *As trajetórias do sistema gradiente* (3.48) *convergem globalmente para o conjunto solução do problema de programação quadrática* (3.46)*, para quaisquer parâmetros positivos ρ e γ.*

O sistema gradiente (3.48) pode ser interpretado como um sistema de controle. Neste contexto, o termo $\mathbf{u}_1 := \rho\mathbf{A}^T\text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})$ é interpretado como um controle com ganho ρ, composto por um relé multidimensional, que é responsável por forçar as trajetórias para a intersecção de hiperplanos $\Delta := \{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$, ao mesmo tempo que o controle $\mathbf{u}_2 = \gamma\text{hsgn}(\mathbf{x})$ é responsável por forçar as trajetórias para o conjunto $\mathbf{x} \geq \mathbf{0}$.

3.8 Resolução de sistemas de equações lineares

A abordagem proposta é baseada em associar o sistema de equações lineares a um problema de otimização sem restrições, cuja solução corresponde à mesma solução do sistema de equações lineares. Ao problema de otimização é associado um sistema tipo gradiente com segundo membro descontínuo. O uso de sistemas gradientes descontínuos para este tipo de problema justifica-se, pois neste caso é possível provar a convergência em tempo finito para a solução do sistema de equações lineares.

Este procedimento foi utilizado por Cichocki e Unbehauen [12], que formularam o problema de resolver sistemas lineares do tipo $\mathbf{Ax} = \mathbf{b}$ através de um problema de otimização sem restrições, utilizando a norma L_p do resíduo $\mathbf{Ax} - \mathbf{b}$, entretanto sem apresentar análise de convergência.

Neste trabalho o sistema de equações lineares é resolvido através da minimização da norma L_1 do vetor de resíduos. A solução em norma L_1 mínima apresenta diversas vantagens, como a menor sensibilidade a ruídos e *outliers* [12, 57].

3.8.1 Formulação do problema

Considere o seguinte sistema de equações lineares:

$$\mathbf{Ax} = \mathbf{b}, \tag{3.51}$$

em que $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ e assume-se que $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \leq n$ possui posto igual a m, que garante a existência de pelo menos uma solução.

O problema da resolução do sistema de equações lineares (3.51) utilizando a norma L_1, pode ser formulado através do seguinte problema de otimização sem restrições

$$\text{minimizar } E(\mathbf{x}) := \|\mathbf{r}(\mathbf{x})\|_1, \ \mathbf{x} \in \mathbb{R}^n, \tag{3.52}$$

sendo que $\mathbf{r}(\mathbf{x}) := \mathbf{Ax} - \mathbf{b}$ é o vetor de resíduos, $\|\cdot\|_1$ é a norma L_1 do argumento.

Note que o mínimo global de E é zero e o conjunto solução do problema (3.52) é dado por $\Delta := \{\mathbf{x} : \mathbf{r}(\mathbf{x}) = \mathbf{0}\}$, que é o conjunto solução do sistema (3.51). O conjunto Δ é

determinado através da seguinte intersecção:

$$\Delta = \bigcap_{i=1}^{m} \Delta_i;\ \Delta_i := \{\mathbf{x} : r_i(\mathbf{x}) = 0\}, \tag{3.53}$$

em que os escalares $r_i : \mathbb{R}^n \to \mathbb{R}$ são componentes do vetor $\mathbf{r}$.

O problema de minimização (3.52) é resolvido através de um sistema gradiente. Porém, como a função objetivo do problema (3.52) é não-diferenciável em Δ, o gradiente de E é descontínuo neste conjunto e, as soluções do sistema gradiente que minimizam E são consideradas no sentido de Filippov. Neste contexto, os conjuntos Δ_i são chamados de *superfícies de descontinuidade*.

Um subgradiente da função objetivo não-diferenciável E do problema (3.52), é dado por:

$$\mathbf{e} = \mathbf{A}^T \text{sgn}(\mathbf{r}), \quad \mathbf{e} \in \partial E(\mathbf{x}),$$

sendo que $\mathbf{r} := \mathbf{A}\mathbf{x} - \mathbf{b}$, e o vetor sgn é definido em (3.22).

O sistema gradiente não-suave $\dot{\mathbf{x}} = -\mathbf{M}\mathbf{e}$, associado ao problema (3.52) é dado por:

$$\dot{\mathbf{x}} = -\mathbf{M}\mathbf{A}^T \text{sgn}(\mathbf{r}), \tag{3.54}$$

sendo que $\mathbf{M} = \text{diag}(\mu_i)_{i=1}^n$ é uma matriz diagonal positiva que, no contexto de redes neurais, é denominada matriz de aprendizado.

Na figura 3.3, é apresentado o diagrama funcional do circuito formulado por (3.54), que pode ser interpretado como uma rede neural [12]. Observe que sgn atua como uma função de ativação da rede que resolve o sistema (3.51). Por outro lado, a representação do sistema (3.54) sob a perspectiva de um sistema de controle é mostrado na figura 3.4, onde o controlador corresponde a um relé multidimensional com matriz de ganho $\mathbf{M}$. Neste contexto, o vetor $\mathbf{A}^T \text{sgn}(\mathbf{r})$ desempenha o papel do controlador responsável por conduzir as trajetórias para o conjunto Δ.

3.8.2 Análise de convergência

A análise de convergência é feita representando o sistema (3.54) na forma Persidskii e utilizando uma função de Lyapunov candidata do tipo diagonal. A representação de (3.54)

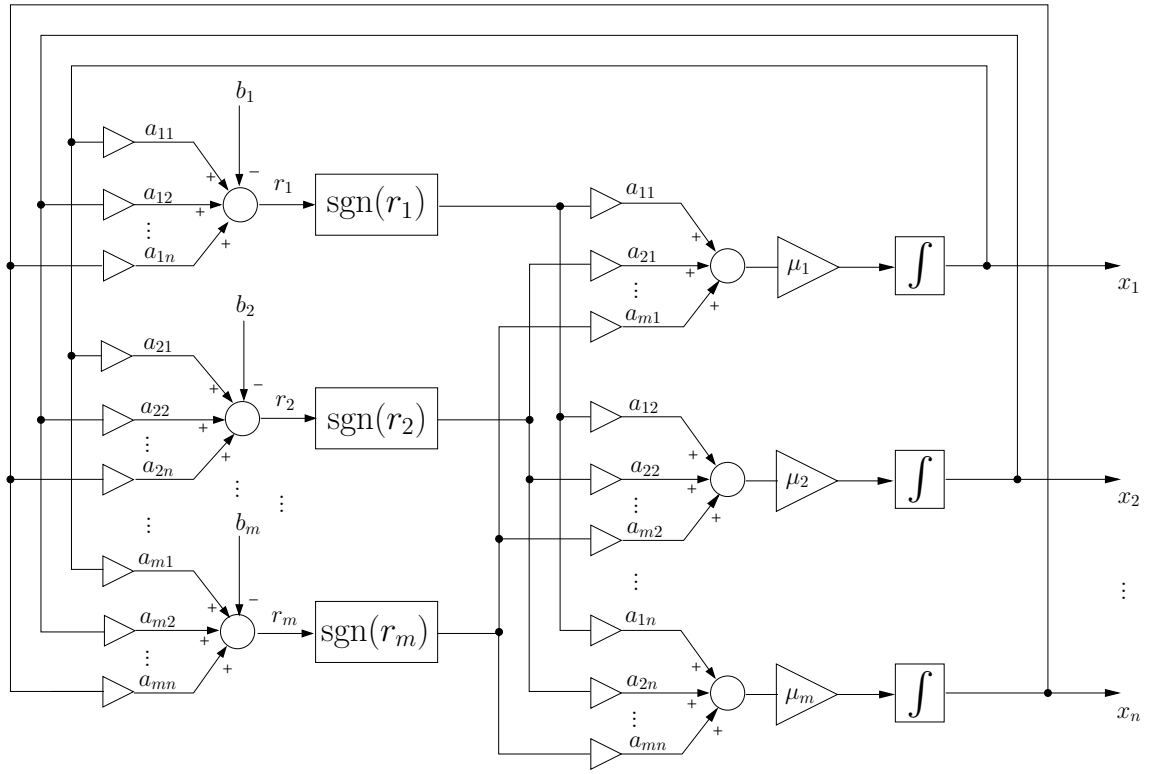


Figura 3.3: Diagrama funcional do circuito descrito pelo sistema gradiente (3.54), visto sob a perspectiva de uma rede neural

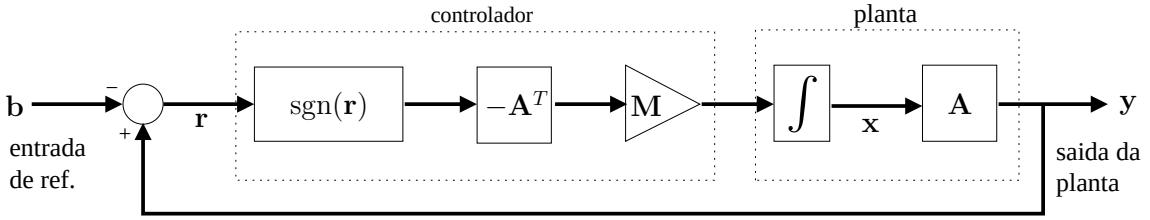


Figura 3.4: Representação do sistema gradiente (3.54) sob a perspectiva de um sistema de controle, em que $\mathbf{y} = \mathbf{A}\mathbf{x}$ é a saída da planta

na forma Persidskii é obtida através da pré-multiplicação de (3.54) pela matriz $\mathbf{A}$. Observe que $\dot{\mathbf{r}} = \mathbf{A}\dot{\mathbf{x}}$, portanto de (3.54) decorre que:

$$\dot{\mathbf{r}} = -\mathbf{A}\mathbf{M}\mathbf{A}^T \mathrm{sgn}(\mathbf{r}). \tag{3.55}$$

Esta classe de sistemas Persidskii é analisada em [51]. Para que o sistema (3.55) possa ser usado na análise de convergência, é necessário provar o resultado a seguir.

Proposição 1. *O sistema* (3.55) *é equivalente ao sistema original* (3.54) *no sentido que* $\dot{\mathbf{r}} \equiv \mathbf{0}$ *se e somente se* $\dot{\mathbf{x}} \equiv \mathbf{0}$.

Prova: Se $\dot{\mathbf{x}} \equiv \mathbf{0}$, é imediato que $\dot{\mathbf{r}} \equiv \mathbf{0}$. Por outro lado, se $\dot{\mathbf{r}} \equiv \mathbf{0}$ então o vetor

$\sqrt{\mathbf{M}}\nabla E = \sqrt{\mathbf{M}}\mathbf{A}^T\text{sgn}(\mathbf{r})$ pertence ao espaço nulo $\mathcal{N}(\mathbf{A}\sqrt{\mathbf{M}})$ da matriz $\mathbf{A}\sqrt{\mathbf{M}}$, entretanto $\sqrt{\mathbf{M}}\nabla E$ é um vetor que pertence ao espaço linha $\mathcal{R}(\sqrt{\mathbf{M}}\mathbf{A}^T)$ de $\mathbf{A}\sqrt{\mathbf{M}}$. Como $\mathcal{N}(\mathbf{A}\sqrt{\mathbf{M}}) \perp \mathcal{R}(\sqrt{\mathbf{M}}\mathbf{A}^T)$, a única solução possível para $\dot{\mathbf{r}} \equiv \mathbf{0}$ é $\sqrt{\mathbf{M}}\,\mathbf{e} \equiv \mathbf{0}$ e, conseqüentemente $\dot{\mathbf{x}} \equiv \mathbf{0}$. ■

Como o sistema (3.55) tem o segundo membro descontínuo, a função de Lyapunov candidata do tipo diagonal escolhida é não-suave [51]:

$$V(\mathbf{r}) = \sum_{i=1}^{m} \int_0^{r_i} \text{sgn}(\tau)\, d\tau = \sum_{i=1}^{m} |r_i|. \tag{3.56}$$

A função V em (3.56) possui as seguintes propriedades:

i) $V(\mathbf{r}) > 0$ para $\mathbf{r} \neq \mathbf{0}$;

ii) $V(\mathbf{r}) = 0$ se e somente se $\mathbf{r} = \mathbf{0}$;

iii) $V(\mathbf{r})$ é radialmente ilimitada;

(iv) Como $V(\mathbf{r})$ é Lipschitz contínua em relação à variável $\mathbf{r}$, e $\mathbf{r}(t)$ é absolutamente contínua, então $V(\mathbf{r}(t))$ é absolutamente contínua em relação à variável t, admitindo derivadas em quase todo t [54].

Seja $\boldsymbol{\xi} \in \partial V(\mathbf{r})$. A derivada no tempo de V ao longo das trajetórias de (3.55) é dada por

$$\dot{V}(\mathbf{r}) = \boldsymbol{\xi}^T\dot{\mathbf{r}} = -\text{sgn}^T(\mathbf{r})\mathbf{A}\mathbf{M}\mathbf{A}^T\text{sgn}(\mathbf{r}). \tag{3.57}$$

Note que como $\mathbf{A}$ tem posto por linhas completo e $\mathbf{M}$ é definida positiva, então $\mathbf{A}\mathbf{M}\mathbf{A}^T$ também é definida positiva. Conseqüentemente, $\dot{V} \equiv 0$ se e somente se $\text{sgn}(\mathbf{r}) \equiv \mathbf{0}$, o que implica em $\dot{\mathbf{x}} = \mathbf{0} \in \partial E(\mathbf{x})$. Logo, $\mathbf{x}$ é um minimizador de E e, como E é convexa, $\mathbf{x}$ é um minimizador global. Portanto $\mathbf{x} \in \Delta$, sendo que Δ é definido em (3.53).

Teorema 3.6. *As trajetórias do sistema* (3.54) *convergem, com quaisquer condições iniciais, para o conjunto solução do sistema de equações lineares* (3.51) *em tempo finito e permanecem neste conjunto. O tempo de convergência t_f satisfaz $t_f \leq (V(\mathbf{r}_0)/\lambda_{min}(\mathbf{A}\mathbf{M}\mathbf{A}^T))$, em que $\mathbf{r}_0 := \mathbf{r}(\mathbf{x}_0)$.*

Prova: Considere o sistema (3.55), a derivada no tempo (3.57) de (3.56) e a partição do conjunto Δ definida em (3.53). Por simplicidade seja $\mathbf{M} = \mathbf{I}$.

Considerando as soluções de (3.55) no sentido de Filippov, o objetivo é mostrar existe um escalar $\varepsilon > 0$, tal que $\dot{V} \leq -\varepsilon$ e, finalmente mostrar que Δ é um conjunto invariante. Duas situações devem ser consideradas:

i) $\mathbf{x} \notin \Delta_i$, para todo i. Neste caso as trajetórias não estão confinadas em nenhuma superfície de descontinuidade e as soluções de (3.55) existem no sentido usual. Neste caso $\partial E(\mathbf{x})$ contém apenas o gradiente de E em $\mathbf{x}$. Como $\mathbf{A}$ tem posto por linhas completo, então a matriz $\mathbf{A}\mathbf{A}^T$ é definida positiva, e usando o princípio de Rayleigh e o fato que $\|\text{sgn}(\mathbf{r})\|_2 = m^2 \geq 1$, para $r_i \neq 0$, podemos escrever:

$$\dot{V}(\mathbf{r}) \leq -\lambda_{min}(\mathbf{A}\mathbf{A}^T)m^2 \leq -\lambda_{min}(\mathbf{A}\mathbf{A}^T), \tag{3.58}$$

em que $\lambda_{min}(\mathbf{A}\mathbf{A}^T) > 0$ é o menor autovalor de $\mathbf{A}\mathbf{A}^T$.

ii) $\mathbf{x} \in \Delta_i$, para algum i e quase todo t em um intervalo $\mathcal{I}$. Neste caso as trajetórias estão confinadas em alguma superfície de descontinuidade Δ_i, ocorrendo um movimento deslizante nestas superfícies. Portanto, os vetores $\mathbf{e}$ que descrevem este movimento são subgradientes de E em $\mathbf{x}$, i.e.,

$$\dot{\mathbf{r}} = -\mathbf{A}\mathbf{e}, \ \mathbf{e} \in \partial E(\mathbf{x}),$$

em que $\mathbf{e} = \mathbf{A}^T\text{sgn}(\mathbf{r})$, e $\text{sgn}(\mathbf{r}) \in \partial V(\mathbf{r})$ é definido em (3.22).

Como existe pelo menos um índice i tal que $\mathbf{x} \notin \Delta_i$, então $\text{sgn}_i(r_i) \in \{-1, 1\}$, implicando em $\|\text{sgn}(\mathbf{r})\|_2^2 \geq 1$. Logo usando (3.57) e o princípio de Rayleigh, chegamos a $\dot{V}(\mathbf{r}) \leq -\lambda_{min}(\mathbf{A}\mathbf{A}^T)$.

Portanto, dos ítens i) e ii), concluímos que as trajetórias de (3.54) convergem para o conjunto Δ em tempo finito. Falta mostrar que as trajetórias permanecem neste conjunto, isto é, que nestas condições Δ é um conjunto invariante. Se $\mathbf{x}(t) \in \Delta$, então $V(t) = 0$ e $\dot{V}(t) = 0$. Se para algum $t = T$ a trajetória $\mathbf{x}(T)$ sai de algum conjunto Δ_i, então $\dot{V}(T) < 0$ e $V(T) > 0$, o que é uma contradição, pois V é não crescente ao longo das trajetórias de (3.55). Logo, as trajetórias de (3.55) atingem Δ e permanecem neste conjunto. Pela proposição 1, este resultado aplica-se também ao sistema gradiente original (3.54).

Resta obter o limite inferior para o tempo de convergência t_f. De (3.58), podemos es-

crever $V(t) \leq V(t_0) - \lambda_{min}(\mathbf{AMA}^T)\ t$, logo, o tempo t_f para $\mathbf{r}$ atingir zero não excede $V_0/\lambda_{min}(\mathbf{AMA}^T)$, concluindo a prova. ■

Na próxima seção são apresentados os métodos de integração numérica empregados para resolver os sistemas gradientes deste trabalho.

3.9 Métodos de integração

Os sistemas gradientes apresentados neste trabalho são resolvidos numericamente através de dois métodos de integração, consagrados na literatura, pertencentes à classe dos métodos de Runge-Kutta: o método de Euler, que é um método de primeira ordem, e um método de Runge-Kutta de segunda ordem. Para maiores detalhes sobre estes, e outros, métodos de integração, ver, por exemplo, [58].

Os métodos de integração com controle automático de passo são utilizados nos capítulos 5 e 6, em que os termos descontínuos dos sistemas gradientes são aproximados por funções contínuas através da técnica da camada limite, descrita no capítulo 2. Nos exemplos deste capítulo e do capítulo 4, o método de Euler com passo constante é utilizado.

3.9.1 Método de Euler

O método de Euler, é o mais simples da classe dos métodos de Runge-Kutta. Este método apresenta baixo custo computacional, pois necessita de apenas uma avaliação do segundo membro do sistema gradiente (3.7) em cada iteração.

A discretização do sistema gradiente (3.7) para resolução através do método de Euler é dada por:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mu_k \nabla E(\mathbf{x}_k), \tag{3.59}$$

sendo que μ_k é o passo de integração.

O método de Euler, implementado com passo $\mu_k = \mu$ constante, é descrito no algoritmo 3.1. No entanto, o bom desempenho do método de Euler, depende do uso de um método eficiente de ajuste automático do passo μ_k. Para esta finalidade, no contexto de sistemas gradientes, diversos métodos estão disponíveis na literatura [59–64]. Utilizamos a fórmula de Barzilai-Borwein [59], que proporciona um controle eficiente do passo integração μ_k de sistemas gradientes discretizados na forma (3.59).

Algoritmo 3.1 Método de Euler com passo constante

$\mathbf{x}_0 =$ condição inicial
$\mu =$ passo de integração
`Para` $k = 0, 1, 2, \ldots$
    $\mathbf{x}_{k+1} = \mathbf{x}_k - \mu \nabla E(\mathbf{x}_k)$ (atualiza solução)
    $\mathbf{x}_k = \mathbf{x}_{k+1}$ (armazena solução anterior)
`Fim`

Algoritmo 3.2 Método de Euler com controle de passo pela fórmula de Barzilai-Borwein

$\mathbf{x}_0 =$ condição inicial apenas para o cálculo do passo
$\mathbf{x}_1 =$ condição inicial
$\nabla E_0 = \nabla E(\mathbf{x}_0)$ (gradiente de E em $\mathbf{x}_0$)
$\nabla E_1 = \nabla E(\mathbf{x}_1)$ (gradiente de E em $\mathbf{x}_1$)
`Para` $k = 1, 2, \ldots$
    $\mu_k = (\mathbf{x}_k - \mathbf{x}_{k-1})^T (\nabla E_k - \nabla E_{k-1}) / \|\nabla E_k - \nabla E_{k-1}\|_2^2$ (calcula novo passo)
    $\mathbf{x}_{k+1} = \mathbf{x}_k - \mu_k \nabla E_k$ (atualiza solução)
    $\mathbf{x}_{k-1} = \mathbf{x}_k$ (armazena penúltima iteração)
    $\nabla E_{k-1} = \nabla E_k$
    $\mathbf{x}_k = \mathbf{x}_{k+1}$ (armazena última iteração)
    $\nabla E_k = \nabla E(\mathbf{x}_k)$ (computa novo gradiente)
`Fim`

O passo de integração μ_k, determinado iterativamente pela fórmula de Barzilai-Borwein, é dado por:

$$\mu_k = \frac{(\mathbf{x}_k - \mathbf{x}_{k-1})^T (\nabla E(\mathbf{x}_k) - \nabla E(\mathbf{x}_{k-1}))}{\|\nabla E(\mathbf{x}_k) - \nabla E(\mathbf{x}_{k-1})\|_2^2}. \tag{3.60}$$

O procedimento de resolução de um sistema gradiente utilizando o método de Euler com controle de passo pela fórmula de Barzilai-Borwein (Euler-BB), é descrito no algoritmo 3.2. O cálculo do passo μ_k depende apenas das diferenças entre gradientes $\nabla E(\mathbf{x}_k)$ e $\nabla E(\mathbf{x}_{k-1})$ e entre os vetores $\mathbf{x}_k$ e $\mathbf{x}_{k-1}$, computados nas duas últimas iterações do método de Euler. Adicionalmente, Barzilai e Borwein sugerem que o método de Euler em (3.59), com controle de passo através da fórmula 3.60, é menos sensível ao mal-condicionamento do problema.

Por outro lado, a memória necessária para o armazenamento da solução $\mathbf{x}_{k-1}$ e do vetor gradiente ∇E_{k-1}, computados na penúltima iteração, para o cálculo do passo μ_k em (3.59), representa um custo adicional do algoritmo 3.2 que, em geral, é pequeno.

Algoritmo 3.3 Método de Runge-Kutta de Segunda Ordem

$t_0 =$ instante inicial
$\mathbf{x}_0 =$ condição inicial
$\mu_0 =$ passo inicial
`Para` $k = 0, 1, 2, \ldots$
$\quad \mathbf{s}_0 = -\mu_k \nabla E(\mathbf{x}_k)$ (primeiro estágio)
$\quad \bar{\mathbf{x}}_{k+1} = \mathbf{x}_k + \mathbf{s}_0$ (solução de primeira ordem)
$\quad \mathbf{s}_1 = -\mu_k \nabla E(\mathbf{x}_k + \frac{1}{5}\mathbf{s}_0)$ (segundo estágio)
$\quad t_{k+1} = t_k + \mu_k$
$\quad \mathbf{x}_{k+1} = \mathbf{x}_k - \frac{3}{2}\mathbf{s}_0 + \frac{5}{2}\mathbf{s}_1$ (solução de segunda ordem)
$\quad$ Atualiza passo de integração μ_k
`Fim`

3.9.2 Método de Runge-Kutta de segunda ordem

O método de Runge-Kutta de segunda ordem, é um método de dois estágios, em que a aproximação obtida no primeiro estágio, é utilizada para computar a aproximação no segundo estágio. Utilizamos este método com os coeficientes determinados por Cash e Karp [65], dados em (3.61), sob a forma de um *tableau*. O método de Runge-Kutta de segunda ordem, aplicado ao sistema gradiente (3.7), utilizando os coeficientes de Cash e Karp, é descrito no algoritmo 3.3.

$$\begin{array}{c|cc|l} 0 & 0 & & \\ \frac{1}{5} & \frac{1}{5} & 0 & \\ \hline & -\frac{3}{2} & \frac{5}{2} & \text{Solução de } 2^{\underline{a}} \text{ ordem} \\ & 1 & 0 & \text{Solução de } 1^{\underline{a}} \text{ ordem} \end{array} \tag{3.61}$$

Para que o método de Runge-Kutta de segunda ordem apresente um bom desempenho, é fundamental que seja utilizado um algoritmo eficiente de controle automático do passo de integração. O controle do passo integração μ_k é feito automaticamente através do algoritmo baseado em um controlador PID descrito em [66]. O passo μ_k é ajustado através da seguinte fórmula:

$$\mu_{k+1} = \left(\frac{e_{k-1}}{e_k}\right)^P \left(\frac{1}{e_k}\right)^I \left(\frac{e_{k-1}^2}{e_k e_{k-2}}\right)^D \mu_k, \tag{3.62}$$

sendo que $e_k = \|\mathbf{x}_k - \bar{\mathbf{x}}_k\|/(\|\mathbf{x}_k\| \times tol)$, tol é uma tolerância para o integrador determinada a priori, e P, I e D são os parâmetros do controlador, também determinados a priori.

3.10 Exemplos numéricos

Com o objetivo de ilustrar a utilização dos sistemas gradientes apresentados neste capítulo, seguem exemplos de simulação numérica de cada um dos casos discutidos.

Devido à presença de termos descontínuos no segundo membro dos sistemas gradientes acima, as soluções dos mesmos são afetadas por *chattering*. A amplitude do chattering, que pode ser controlada através dos ganhos das não linearidades, faz com que a precisão das soluções obtidas seja comprometida.

Nos exemplos abaixo, as soluções numéricas dos sistemas gradientes foram obtidas através do método de Euler, descrito no algoritmo 3.1, em que o passo de integração μ é constante, igual a 10^{-6}, e garante soluções numéricas suficientemente precisas dos sistemas gradientes utilizados.

Os ganhos das não-linearidades foram escolhidos de modo a não aumentar a amplitude do chattering e, ao mesmo tempo, satisfazendo as condições de convergência estabelecidas pelos teoremas correspondentes. A exceção é o exemplo de programação quadrática, em que os ganhos das não-linearidades foram escolhidos propositalmente grandes, para evidenciar o fenômeno de chattering e como este afeta as soluções numéricas obtidas.

3.10.1 Programação linear

Consideremos um exemplo de um problema de programação linear na forma canônica II, também considerado em [12]. Os dados do problema são:

$$\mathbf{A} = \begin{pmatrix} 3 & -2 & 4 \\ -1 & -2 & -1 \\ -2 & -1 & 0 \end{pmatrix}$$

$$\mathbf{b} = (8, -9, -6)^T$$

$$\mathbf{c} = (3, 1, 2)^T$$

De acordo com o teorema 3.4, as condições suficientes que os parâmetros de penalidade devem satisfazer para garantir a convergência das trajetórias para a solução do problema

(3.29) são:

$$\gamma > 6.4807$$

$$\rho > 7.1650$$

A solução deste problema é $\mathbf{x}^* = (0, 0, 2)$, obtida através do pacote de otimização do MATLAB. A solução obtida pelo sistema gradiente (3.38), usando os valores para os parâmetros de penalidade $\gamma = 6.5$ e $\rho = 7.17$, é mostrada na figura 3.5. Por esta figura, nota-se que as trajetórias convergem para a solução correta do problema de programação linear em tempo finito.

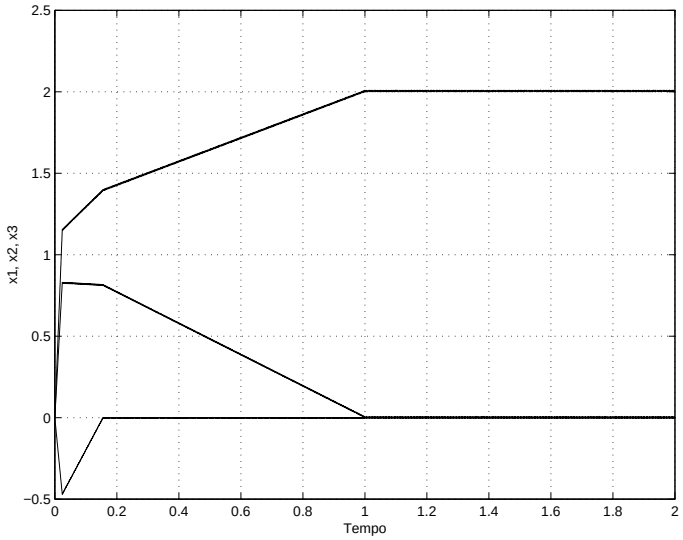


Figura 3.5: Programação linear – convergência em tempo finito para a solução do problema de programação linear na forma canônica II, com os parâmetros de penalidade $\gamma = 6.5$ e $\rho = 7.17$

3.10.2 Programação quadrática

Consideremos o problema (3.48) com os seguintes dados:

$$\mathbf{Q} := \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$$

$$\mathbf{A} := \begin{pmatrix} 5 & -9 \end{pmatrix}; b := 3$$

$$\mathbf{H} := -\mathbf{I}; \mathbf{c} := \mathbf{0}$$

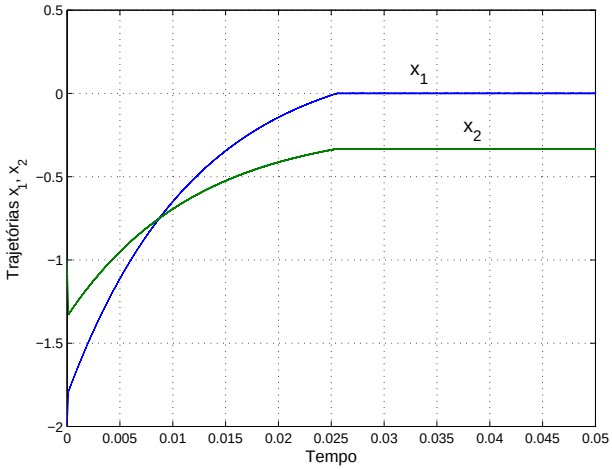


Figura 3.6: Programação quadrática– resultado da simulação numérica do sistema (3.48), com condições iniciais $x_1 = -2$ e $x_2 = -1$, escolhidas arbitrariamente, mostrando convergência das trajetórias para a solução do problema de programação quadrática (3.63)

Com os dados acima, o problema (3.46) toma a forma:

$$\begin{aligned} \text{minimizar} \quad & \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} \\ \text{sujeito a} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \leq \mathbf{0} \end{aligned}$$

A solução do problema anterior é $x_1 = 0$ e $x_2 = -0.33$, obtida através do comando `quadprog`, do pacote de otimização do MATLAB 6.5.

A simulação foi executada com os parâmetros de penalidade $\rho = 5$ e $\gamma = 5$. As trajetórias do sistema (3.48) são apresentadas na figura 3.6. As curvas de nível da função objetivo quadrática e a reta $\Pi = \{x_1, x_2 : 5x_1 - 9x_2 = 3\}$, que faz parte do conjunto de restrições do problema de programação quadrática considerado, são mostradas na figura 3.7. A trajetória parte da condição inicial $\mathbf{x}_0 = (-2, -1)$ e atinge a reta Π, onde fica confinada. Um movimento deslizante é gerado sobre esta reta, indicado pela linha mais grossa na figura, até atingir a solução ótima $\mathbf{x}^* = (0, -0.33)$. Os valores assumidos pela função objetivo ao longo da trajetória obtida pelo sistema gradiente (3.48) são mostrados na figura 3.8.

O sistema gradiente (3.48) pode ser interpretado como um sistema de controle. O termo $\mathbf{u}_1 := \rho\mathbf{A}^T\text{sgn}(\mathbf{A}\mathbf{x} - \mathbf{b})$ pode ser visto como um controle com ganho ρ, composto por um relé multidimensional, que é responsável por forçar as trajetórias para a reta $\Pi := \{x_1, x_2 : 5x_1 - 9x_2 = 3\}$, ao mesmo tempo que o controle $\mathbf{u}_2 = \gamma\text{hsgn}(-\mathbf{x})$ é responsável por forçar

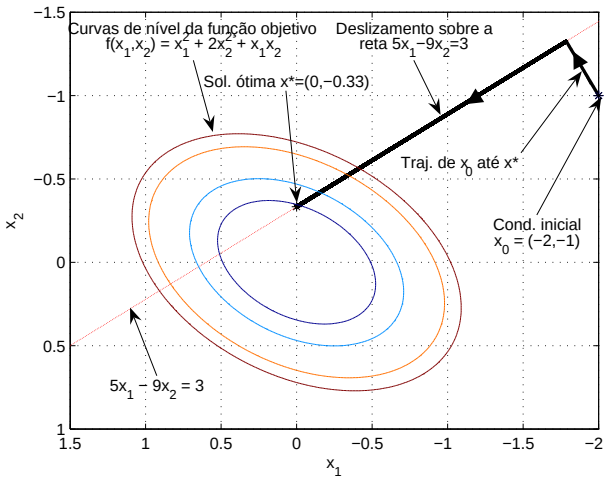


Figura 3.7: Programação quadrática – curvas de nível da função objetivo do problema de programação quadrática, mostrando a trajetória do sistema (3.48) partindo da condição inicial $\mathbf{x}_0 = (-2, -1)$ até a solução ótima $\mathbf{x}^* = (0, -0.33)$

as trajetórias para o conjunto $\mathbf{x} \leq \mathbf{0}$. Este fenômeno pode ser observado na figura 3.8, que mostra as trajetórias x_1 e x_2, partindo da condição inicial $(-2, -1)$, em que o controle u_2 não atua, pois a restrição de sinal sobre o vetor $\mathbf{x}$ está satisfeita. O controle $\mathbf{u}_1$ conduz as trajetórias em direção à reta Π. O caminho sobre esta reta é mostrado na figura 3.8.

Ao atingir a reta Π, um movimento deslizante é iniciado sobre esta reta. O controle $\mathbf{u}_1$ atua mantendo a trajetória em Π, enquanto que o controle $\mathbf{u}_2$ não atua, pois a restrição $\mathbf{x} \leq \mathbf{0}$ está satisfeita.

Quando o ponto ótimo $\mathbf{x}^* = (0, -0.33)$ é atingido, o controle $\mathbf{u}_2$ atua sobre a coordenada x_1, sempre que a trajetória ultrapassa o ponto $x_1^* = 0$, forçando a satisfação da restrição $x_1 \leq 0$. Observe que o controle $\mathbf{u}_2$ não atua em $x_2^* = -0.33$. Entretanto, como o movimento ocorre na direção de maior descida, isto é, na direção de $-\nabla E$, o gradiente da função E impede que as trajetória x_2 ultrapasse o ponto $x_2^* = -0.33$, pois neste caso o valor da função E passa a crescer.

Com o objetivo de ilustrar os efeitos do chattering na solução do problema de programação quadrática, a simulação foi repetida com o parâmetro de penalidade aumentado para $\rho = 200$. Com este parâmetro de penalidade, o chattering é claramente perceptível, como pode ser observado nas trajetórias no plano $x_1 \times x_2$ mostradas na figura 3.9-(a) e nas trajetórias em função do tempo, mostradas na figura 3.9-(b). Observe que o chattering só ocorre quando a trajetória atinge a reta Π, onde o segundo membro do sistema gradiente (3.48) é descontínuo.

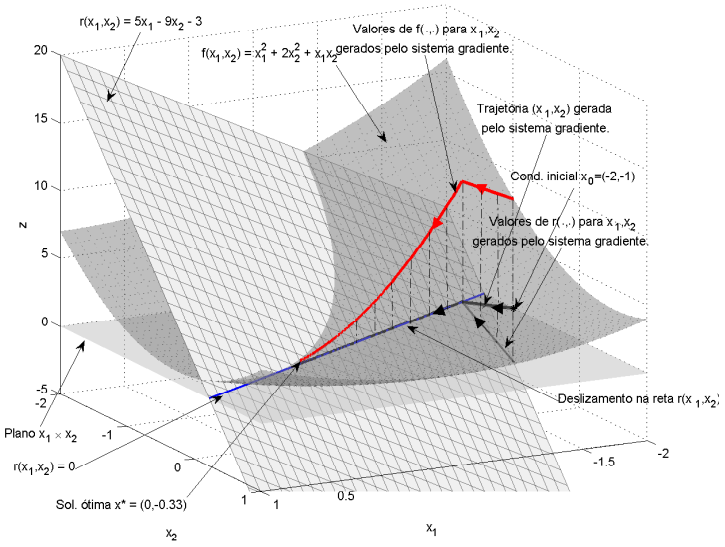
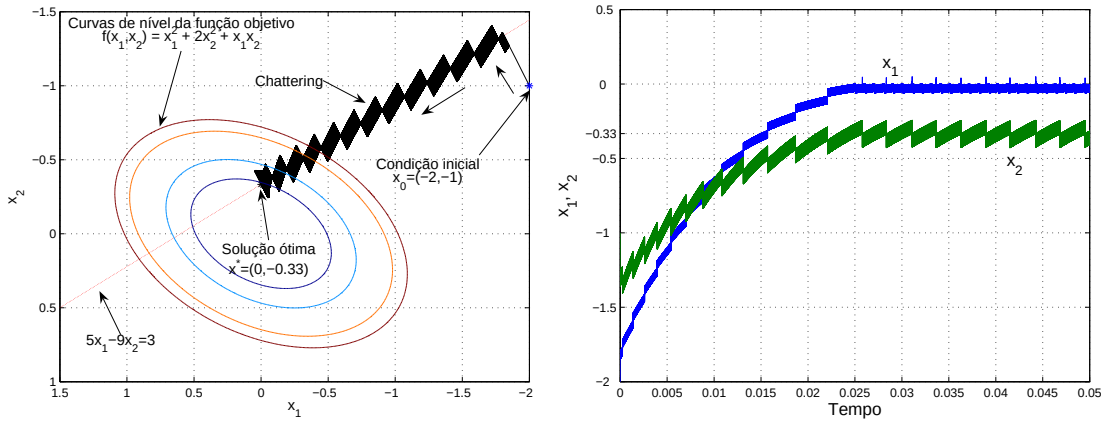


Figura 3.8: Programação quadrática – a linha grossa sobre a superfície da função quadrática indica os valores assumidos pela função objetivo ao longo da trajetória obtida através do sistema gradiente (3.48). No plano $x_1 \times x_2$ é mostrada a trajetória partindo da condição inicial $x_0 = (-2, -1)$, atingindo a reta $5x_1 - 9x_2 = 3$ e realizando um movimento deslizante sobre ela até atingir a solução ótima $\mathbf{x}^*$

A amplitude do chattering, neste exemplo, é grande o suficiente para comprometer a precisão da solução obtida. Na figura 3.9-(a) nota-se que a restrição de igualdade do problema não é satisfeita e as soluções permanecem em uma vizinhança da reta Π de raio consideravelmente grande. Examinando as trajetórias em função do tempo na figura 3.9-(b), nota-se que, devido ao chattering, a solução do problema é difícil de ser identificada com precisão, a trajetória x_1 varia ao longo do tempo no intervalo $[-0.35, 0.35]$, enquanto que a trajetória x_2 varia no intervalo $[-0.39, -0.27]$.

Uma aplicação em que o chattering impede a determinação correta da solução é o treinamento de ν-SVMs, discutido no capítulo 5. A fase de treinamento do ν-SVM é modelada por um problema de programação quadrática com restrições de igualdade, desigualdade e em caixa. A restrição em caixa é dada por $\alpha_i \in [0, 1/m]$, para $i = 1, \ldots, m$, em que α_i são as variáveis de decisão, e m é a dimensão da matriz $\mathbf{Q}$. Quando m é grande, o comprimento deste intervalo é pequeno. Neste caso, devido ao chattering, esta restrição não é satisfeita e a solução correta não é obtida.



(a) (b)

Figura 3.9: Programação quadrática– (a) curvas de nível da função objetivo do problema de programação quadrática, enfatizando o chattering que inicia quando a trajetória, partindo do ponto inicial $(-2, -1)$, atinge a reta Π e continua até atingir o ponto ótimo $(0, -0.33)$; (b) Programação quadrática– trajetórias em função do tempo, partindo do ponto inicial $(-2, -1)$ e evidenciando o chattering

3.10.3 Sistemas de equações lineares

Consideremos o sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$ com os seguintes dados:

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & -1 \\ 2 & 1 & 4 \end{pmatrix}; \quad \mathbf{b} = (1 - 3)^T$$

Utilizamos o sistema (3.54) para resolver este sistema de equações lineares. Observe que a matriz $\mathbf{A}$ tem posto completo por linhas, logo, o sistema $\mathbf{Ax} = \mathbf{b}$ tem pelo menos uma solução.

Através de observações das simulações com diferentes condições iniciais, notamos que as trajetórias do sistema convergem para a solução do sistema de equações lineares mais próxima da condição inicial escolhida, o que é justificado pelo fato de que (3.51) tem infinitas soluções e portanto a função E em (3.52) têm infinitos mínimos locais.

A matriz de ganho $\mathbf{M}$ é utilizada para aumentar ou reduzir a velocidade de convergência das trajetórias para um conjunto Δ. Realizamos simulações para três escolhas de $\mathbf{M}$. Nos experimentos realizados, a matriz de ganho $\mathbf{M} = \mathbf{I}$ proporcionou um tempo de convergência menor que aquele obtido com $\mathbf{M} = 0.3\mathbf{I}$, porém a convergência mais rápida ocorreu para $\mathbf{M} = 10\mathbf{I}$. O resultado da simulação com $\mathbf{M} = 10\,\mathbf{I}$ e condições iniciais, escolhidas arbi-

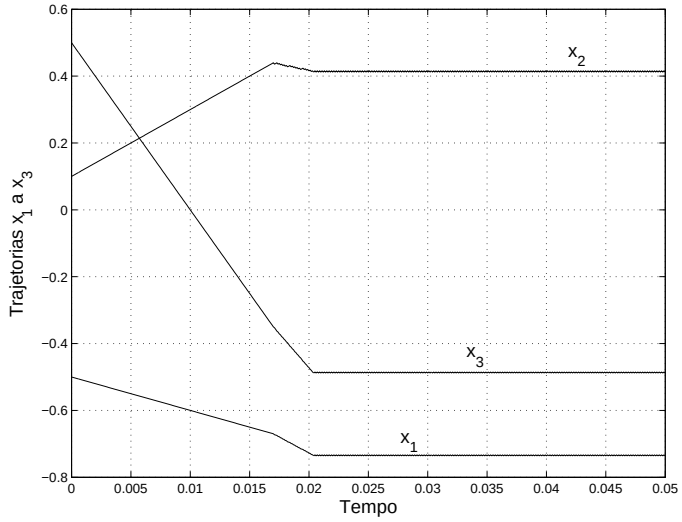


Figura 3.10: Sistemas de equações lineares – trajetórias x_1 a x_3, mostrando a convergência em tempo finito para o conjunto solução do sistema de equações lineares. Resultados usando $\mathbf{M} = 10\mathbf{I}$

trariamente, $\mathbf{x}_0 = (-0.5\ 0.1\ 0.5)^T$, é exibido na figura 3.10, mostrando a convergência em tempo finito para uma solução do sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$.

O aumento dos valores diagonais da matriz de ganho $\mathbf{M}$ tem como efeito, além de aumentar a velocidade de convergência, aumentar também a amplitude do fenômeno de *chattering*, que ocorre devido a presença da função sgn, que é descontínua no conjunto solução do sistema (3.51). Devido a este problema, recomenda-se cautela na escolha da matriz de ganho $\mathbf{M}$, que deve ser escolhida de modo a aumentar a velocidade de convergência, porém mantendo a amplitude do chattering em níveis reduzidos.

O sistema gradiente (3.54) também é adequado para a resolução de sistemas equações lineares de dimensão mais elevada. No capítulo 5, o problema separação de duas classes é resolvido através de LS-SVM, que é formulado através de um sistema de equações lineares quadrado, e no capítulo 6, este sistema é utilizado para a resolução de sistemas de equações lineares decorrentes do problema de restauração de imagens.

Síntese do capítulo

Neste capítulo, apresentamos os sistemas gradientes associados aos problemas de otimização linear, quadrática e destinados à resolução de sistemas de equações lineares. Análises e resultados de convergência para os sistemas obtidos foram apresentados.

Sistemas Persidskii e funções do tipo diagonal foram definidos, e um teorema geral de convergência global para uma classe de sistemas Persidskii com segundo membro descontínuo foi provado, utilizando uma função de Lyapunov do tipo Lure-Persidskii. Este teorema foi usado para provar a convergência global de sistemas gradientes que resolvem problemas de otimização quadrática.

Exemplos ilustrativos do uso dos sistemas propostos foram apresentados. Estes exemplos ilustraram a aplicação dos teoremas de convergência para o ajuste correto dos parâmetros, bem como a ocorrência do fenômeno de *chattering* e os seus efeitos nas soluções obtidas.

No próximo capítulo, apresentamos a aplicação do sistema gradiente para a resolução do problema de discriminar os k maiores elementos de um vetor. Este problema é conhecido na literatura como *k*-winners-take-all (KWTA).

Capítulo 4

Resolução do problema "*k*-winners-take-all" utilizando um sistema gradiente

O problema de determinar os k maiores componentes de um dado vetor $\mathbf{c} \in \mathbb{R}^n$ é conhecido como *k*-winners-take-all (KWTA). Este problema surge em diversos campos, como reconhecimento de padrões, memórias associativas e redes de aprendizado competitivo [67–69]. Diversas propostas de modelos de redes neurais para resolver este problema estão disponíveis na literatura [70–73]. Estas redes são conhecidas como *redes KWTA*.

Neste capítulo, propomos uma rede KWTA obtida através de um problema de programação linear (PPL) com variáveis limitadas, utilizando um método de penalidade exata. Funções de penalidade exatas já foram utilizadas por Cichocki e Unbehauen [12] e Chong et al. [15] para resolver problemas de programação linear. A função objetivo do problema sem restrições obtido é minimizada por um sistema gradiente, que pode ser realizado como uma rede KWTA.

Esta formulação é parcialmente inspirada na formulação apresentada em Urahama e Nagao [70], baseada no seguinte problema de programação inteira:

$$\begin{aligned} &\text{maximizar } z = \mathbf{c}^T\mathbf{x} \\ &\text{sujeito a } \mathbf{1}^T\mathbf{x} = k \\ &\mathbf{x} \in \{0,1\}^n, \end{aligned} \tag{4.1}$$

que é mapeado em um problema de programação não-linear, e é resolvido minimizando-se uma função lagrangeana associada.

O circuito proposto por Urahama e Nagao [70] não possui realimentação e tem como vantagens a alta velocidade na obtenção da solução do problema e a facilidade de implementação. Segundo os autores, a convergência é praticamente instantânea e o circuito não apresenta os longos transitórios que apresentam os circuitos baseados em sistemas dinâmicos. Entretanto, os próprios autores afirmam que o circuito proposto apresenta menor resolução que os baseados em sistemas dinâmicos, e um exemplo ilustrando este aspecto é dado [70].

Neste capítulo, apresentamos uma modificação da proposta de Urahama e Nagao [70]. Ao invés de utilizar o problema de programação inteira, relaxamos o problema (4.1) para um PPL com as variáveis confinadas no intervalo $[0, 1]$.

Em comparação com a abordagem de Urahama e Nagao [70], o PPL tem vantagens. O gradiente da função de energia construída a partir do método de penalidade é facilmente obtido, levando à derivação de condições de convergência simples. Adicionalmente, o circuito obtido é simples e pode ser implementado fisicamente utilizando resistores, capacitores, amplificadores e switches. Como sugerido por Cichocki e Unbehauen [12], o transitório da rede pode ser ajustado através do uso de um fator de escala de tempo no modelo, o que torna a rede proposta adequada à operação em tempo real. Os exemplos apresentados na seção 4.5 indicam que a rede proposta apresenta melhor resolução que o circuito proposto por Urahama e Nagao [70].

A facilidade de paralelização é uma vantagem adicional do circuito proposto. Devido à estrutura mais simples do que as propostas mencionadas acima, a paralelização também é fácil, e a comunicação necessária entre os processos paralelos é também simples, envolvendo apenas grandezas escalares, reduzindo o *overhead* de comunicação. Esta vantagem é particularmente significativa no que diz respeito à implementação em arquiteturas de memória distribuída, como *clusters* de PCs, nas quais a comunicação entre processos é feita por meio de redes.

A estratégia de análise apresentada neste capítulo já foi utilizada por Ferreira et al. [21] para analisar sistemas gradientes que resolvem diversas variantes de problemas de programação linear. Em [24], os resultados discutidos neste capítulo são apresentados de forma resumida. Nestes trabalhos, funções de Lyapunov do tipo diagonal e sistemas Persidskii foram usados para obter condições de convergência para os sistemas gradientes apresenta-

dos.

A estrutura lógica da análise de convergência do circuito proposto segue os passos da análise em [15] e da teoria de convergência em tempo finito no mesmo trabalho. De fato, Chong et al. [15] apresentaram uma análise rigorosa de redes neurais para a resolução de problemas de programação linear na forma padrão utilizando sistemas gradientes com o segundo membro descontínuo. São obtidas condições suficientes que os parâmetros de penalidade devem satisfazer para que a convergência em tempo finito para o conjunto solução do PPL seja garantida. Entretanto, estas condições são difíceis de calcular ou estimar, pois dependem das trajetórias do sistema e, de acordo com Zhang [23], uma sub-rede adicional é necessária para a determinação *online* dos parâmetros de penalidade.

Em contraste, a análise apresentada neste capítulo resulta em limites inferiores constantes, dependentes apenas dos parâmetros do PPL, que os parâmetros de penalidade devem satisfazer para garantir convergência em tempo finito para a solução do PPL, o que é uma vantagem da proposta baseada em programação linear, pois o circuito é mais simples e os parâmetros de penalidade são calculados *offline* e apenas uma vez.

Em Utkin [7] também são utilizadas funções de penalidade exatas e modos deslizantes para obter sistemas gradientes que resolvem problemas de otimização, cuja análise é feita utilizando o método do controle equivalente, que também é usado neste capítulo na análise de convergência para obter a equação do sistema em modo deslizante. Neste contexto, a contribuição deste capítulo é a análise através de uma função de Lyapunov de uma classe similar de sistemas dinâmicos, levando a condições simples que os parâmetros de penalidade devem satisfazer para garantir convergência global para a solução do problema proposto.

Para melhorar a legibilidade deste capítulo, as provas das proposições, lemas e teoremas são agrupados na seção 4.6.

4.1 Modelagem matemática do problema

Considere o seguinte PPL com variáveis limitadas:

$$\begin{aligned} &\text{maximizar } z = \mathbf{c}^T\mathbf{x} \\ &\text{sujeito a } \mathbf{1}^T\mathbf{x} = k \\ &\mathbf{x} \in [0,1]^n, \end{aligned} \tag{4.2}$$

sendo que $\mathbf{c} = [c_1, \ldots, c_n]^T$, $\mathbf{1} = [1, \ldots, 1]^T \in \mathbb{R}^{n\times 1}$, $k \leq n \in \mathbb{N}$ é um inteiro positivo e $\mathbf{x} \in \mathbb{R}^{n\times 1}$.

O conjunto viável de (4.2) é fechado e limitado, o que garante que a solução do problema (4.2) existe e é limitada. Se os componentes do vetor $\mathbf{c}$ são distintos, a solução do problema (4.2) apresenta duas propriedades, formalmente enunciadas na proposição 2, que possibilitam a síntese de uma rede KWTA:

1. A solução $\mathbf{x}^*$ do problema (4.2) é única e tem k componentes iguais a 1 e $n - k$ componentes iguais a 0.

2. Os k componentes não nulos da solução $\mathbf{x}^*$ correspondem exatamente aos k maiores componentes do vetor $\mathbf{c}$ da função objetivo.

Proposição 2. *Considere o PPL* (4.2)*, e os componentes do vetor* $\mathbf{c}$ *distintos. Então, a solução do problema* (4.2) *é única e apresenta* k *componentes iguais a 1, que correspondentemente, multiplicam os* k *maiores componentes do vetor* $\mathbf{c}$ *na função objetivo* z*, enquanto que os* $n - k$ *componentes restantes são iguais a zero.*

Prova: Ver seção 4.6. ■

Comparando com a formulação utilizando programação inteira, considerada em Urahama e Nagao [70], que é obtida através da substituição das restrições sobre o vetor $\mathbf{x}$, no problema (4.2), por $x_i \in \{0, 1\}$, $i = 1, \ldots, n$. A proposição 2 garante que o problema de programação inteira (4.1) e o PPL (4.2) têm a mesma solução $\mathbf{x}^*$.

Com base na proposição anterior, podemos construir um circuito que resolve o problema (4.2), ou, em outras palavras, projetar um circuito que seleciona os k maiores componentes de um vetor $\mathbf{c}$. Este circuito é apresentado na próxima seção e é referido como um *circuito KWTA*.

4.2 Formulação matemática do circuito KWTA

O circuito que resolve o problema (4.2) é formulado matematicamente através de um sistema gradiente não-suave. Utilizamos o *método da função de penalidade* com duas funções de penalidade exatas, cada uma associada ao conjunto de restrições correspondente do problema (4.2).

Após converter o problema (4.2) em um problema de minimização, utilizando o método de penalização exata, obtemos o seguinte problema de otimização sem restrições associado ao problema original:

$$\text{minimizar } E(\mathbf{x},\gamma,\rho) = -\,\mathbf{c}^T\mathbf{x} - \gamma\left(\sum_{j=1}^{n}\min(0,x_j) - \sum_{i=1}^{n}x_j^+\right) + \rho\,|\mathbf{1}^T\mathbf{x} - k| \qquad (4.3)$$

sendo que, para cada j

$$x_j^+ = \begin{cases} x_j & \text{se } x_j > 1 \\ 0 & \text{se } x_j \leq 1. \end{cases}$$

Observe que E é formada pela função objetivo do problema original mais dois termos de penalidade, cada um correspondendo a um conjunto de restrições diferente. O termo $\gamma(\sum_{j=1}^{n}\min(0,x_j) + \sum_{i=1}^{n}x_j^+)$ está associado à restrição em caixa da variável $\mathbf{x}$, enquanto que o termo $\rho|\mathbf{1}^T\mathbf{x} - k|$ esta associado à restrição de igualdade. Como estas funções de penalidade são exatas, existem valores finitos de γ e ρ, para os quais a solução do problema (4.3) corresponde exatamente à solução do problema original (4.2).

A função objetivo E, definida em (4.3), é convexa, o que garante que qualquer mínimo local desta função é global. Entretanto, devido ao uso de funções de penalidade exatas, esta função é não-diferenciável. Seja $\mathbf{e}$ um subgradiente de E, dado por:

$$\mathbf{e} = -\mathbf{c} + \gamma[\text{hsgn}(\mathbf{x}) + \text{ uhsgn}(\mathbf{x})] + \rho\mathbf{1}\text{sgn}(\mathbf{1}^T\mathbf{x} - k), \quad \mathbf{e} \in \partial E(\mathbf{x})$$

sendo que o vetor uhsgn é um subgradiente da função de penalidade $\sum_{i=1}^{n}x_j^+$, e cada componente deste vetor é definido em (4.4).

$$\text{uhsgn}(x_j) \begin{cases} = 0 & \text{se } x_j < 1 \\ = 1 & \text{se } x_j > 1 \\ \in [0,1] & \text{se } x_j = 0 \end{cases} \qquad (4.4)$$

Consideremos o sistema gradiente $\dot{\mathbf{x}} = -\mathbf{M}\mathbf{e}$, que minimiza E, dado por

$$\dot{\mathbf{x}} = \mathbf{M}\mathbf{c} - \gamma\mathbf{M}[\text{hsgn}(\mathbf{x}) + \text{ uhsgn}(\mathbf{x})] - \rho\mathbf{M}\mathbf{1}\text{sgn}(\mathbf{1}^T\mathbf{x} - k). \qquad (4.5)$$

As soluções do sistema (4.5), no sentido de Filippov, são funções vetoriais $\mathbf{x}(t)$ absolutamente contínuas que, para quase todo $t \in [t_1, t_2]$, satisfazem a inclusão diferencial:

$$\dot{\mathbf{x}}(t) \in -\partial E(\mathbf{x}(t)).$$

Por compacidade de notação, definimos as seguintes funções vetoriais:

$$r = \mathbf{1}^T\mathbf{x} - k \tag{4.6}$$

$$\mathbf{h}(\mathbf{x}) = \text{hsgn}(\mathbf{x}) + \text{ uhsgn}(\mathbf{x}), \tag{4.7}$$

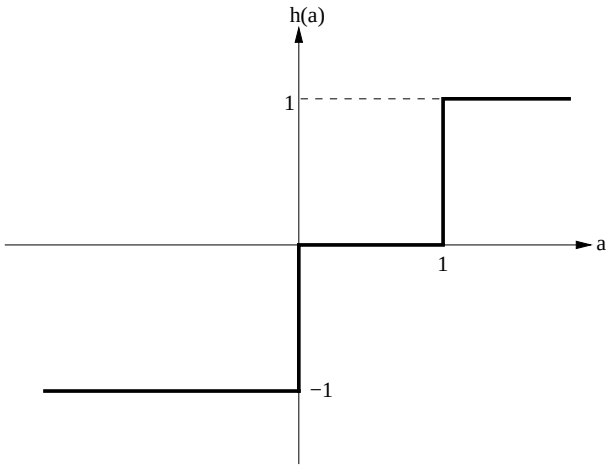


Figura 4.1: A não-linearidade h, associada à restrição de caixa sobre a variável $\mathbf{x}$ do problema (4.2).

Usando (4.6) e (4.7), a equação (4.5) pode ser escrita como:

$$\dot{\mathbf{x}} = \mathbf{c} - \gamma\mathbf{h}(\mathbf{x}) - \rho\mathbf{1}\text{sgn}(r). \tag{4.8}$$

Note que os argumentos da não-linearidade h são separáveis, permitindo que o sistema (4.8) seja representado em uma forma Persidskii.

Como as funções $\mathbf{h}(\mathbf{x})$ e $\text{sgn}(r)$ são limitadas, pode-se mostrar que $\|\mathbf{e}\| \leq \|\mathbf{c}\| + (\gamma + \rho)\sqrt{n}$, satisfazendo as condições dos Teoremas 5 e 6 em [37], que garantem a existência de soluções do sistema gradiente (4.8).

A descrição do movimento quando as variáveis de estado de (4.8) estão confinadas à superfície de descontinuidade $\Pi := \{\mathbf{x} : \mathbf{1}^T\mathbf{x} - k = 0\}$, pode ser determinado através do método do controle equivalente. Assumindo que $\mathbf{x}(t) \in \Pi$, para quase todo $t \in [t_0, t_f]$, e

seja $u = \rho\text{sgn}(r)$, tomando a derivada no tempo de r e igualando a zero obtemos:

$$\dot{r} = \mathbf{1}^T\dot{\mathbf{x}} = \mathbf{0}.$$

Substituindo (4.8) na última equação, resulta em:

$$\mathbf{1}^T\mathbf{c} - \gamma\mathbf{1}^T\mathbf{h}(\mathbf{x}) - \mathbf{1}^T\mathbf{1}u = \mathbf{0}.$$

Resolvendo a equação acima para u e notando que $\mathbf{1}^T\mathbf{1} = n$, obtemos o controle equivalente:

$$u_{eq} = \frac{1}{n}(\mathbf{1}^T\mathbf{c} - \gamma\mathbf{1}^T\mathbf{h}(\mathbf{x})).$$

Substituindo u_{eq} em (4.8), obtemos a seguinte equação do movimento deslizante:

$$\dot{\mathbf{x}} = \mathbf{P}[\mathbf{c} - \gamma\mathbf{h}(\mathbf{x})], \tag{4.9}$$

em que $\mathbf{P} := \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ é a matriz de projeção no espaço nulo de $\mathbf{1}^T$. Esta é a equação que descreve o movimento sobre hiperplano Π, que é a superfície de descontinuidade do segundo membro de (4.8). Note que a matriz de projeção $\mathbf{P}$ é dada por $p_{ij} = 1 - 1/n$ se $i = j$ e $p_{i,j} = -1/n$ caso contrário.

O sistema gradiente (4.8) descreve uma rede que, para γ e ρ suficientemente grandes, é capaz de determinar os k maiores componentes do vetor $\mathbf{c}$, que são chamados de *vencedores*. Os limites inferiores que γ e ρ devem satisfazer para garantir que o circuito proposto possui a propriedade KWTA, são obtidas na próxima seção. O diagrama funcional desta rede é mostrado na figura 4.2.

O vetor $\mathbf{c}$ é a entrada da rede e os parâmetros de penalidade γ e ρ são interpretados como ganhos. As saídas da rede x_j determinam os vencedores e os perdedores, que são determinados quando as saídas x_j, após um transitório, convergem para a solução do PPL (4.2), neste estágio as k saídas x_j que convergem para a unidade, correspondem aos k vencedores e as restantes correspondem aos perdedores.

Dois termos distintos podem ser identificados na figura 4.2, cada um associado ao con-

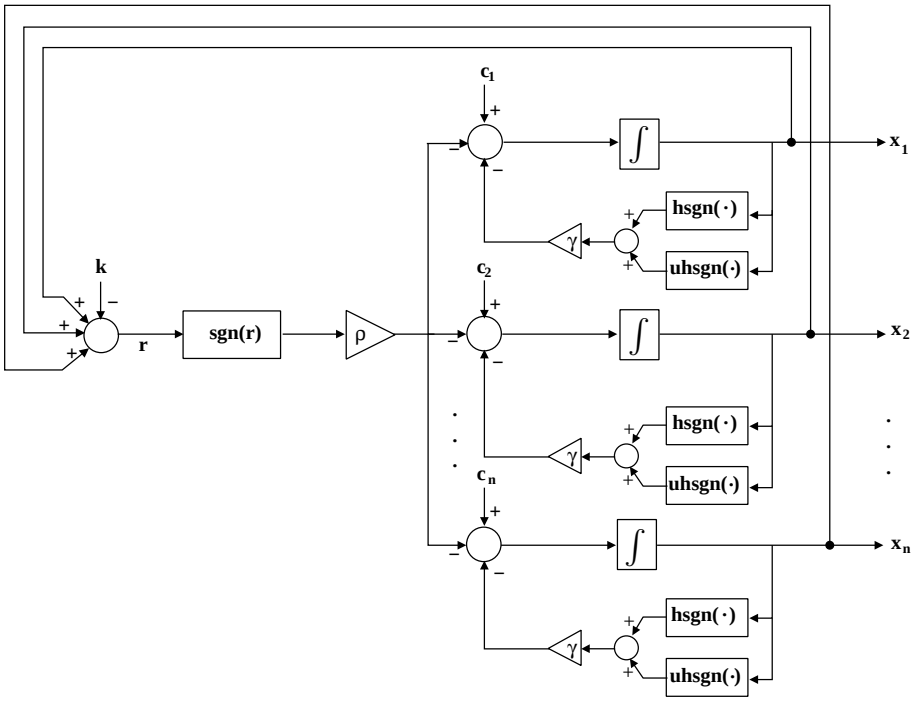


Figura 4.2: Diagrama funcional da rede KWTA descrita pelo sistema gradiente (4.8), neste contexto os parâmetros de penalidade γ e ρ são interpretados como os ganhos da rede.

junto de restrições correspondente:

$$\mathbf{u}_1 = \rho \mathbf{1}^T \text{sgn}(\mathbf{1}^T\mathbf{x} - k)$$
$$\mathbf{u}_2 = \gamma \mathbf{h}(\mathbf{x}).$$

Para γ e ρ suficientemente grandes, o termo $\mathbf{u}_1$ é responsável por manter as trajetórias no conjunto $\{\mathbf{x} : \mathbf{1}^T\mathbf{x} - k = 0\}$. Ao mesmo tempo, o termo $\mathbf{u}_2$ é responsável por manter as trajetórias no conjunto $\{\mathbf{x} : x_j \in [0, 1]\}$. Estes dois controles forçam as trajetórias para o conjunto viável do PPL (4.2).

Caso as saídas dos controladores $\mathbf{u}_1$ e $\mathbf{u}_2$ sejam iguais a zero, o sistema gradiente (4.8) é reduzido a $\dot{\mathbf{x}} = \mathbf{c}$. Como $\mathbf{c}$ é o gradiente da função objetivo $\mathbf{c}^T\mathbf{x}$ do PPL (4.2), o movimento ocorre na direção que proporciona maior acréscimo à função objetivo, e continua até que uma das restrições seja violada. Quando alguma restrição é violada, o controlador correspondente é ativado, gerando um movimento no sentido oposto, que força a trajetória de volta ao conjunto viável.

A figura 4.3 ilustra com um exemplo simples o funcionamento do circuito formulado por (4.8), onde é mostrada a trajetória partindo de um ponto fora do conjunto viável, e atingindo a solução ótima do PPL. Neste exemplo tomamos $\mathbf{c} = (2, 1)$ e $k = 1$. A trajetória entra no

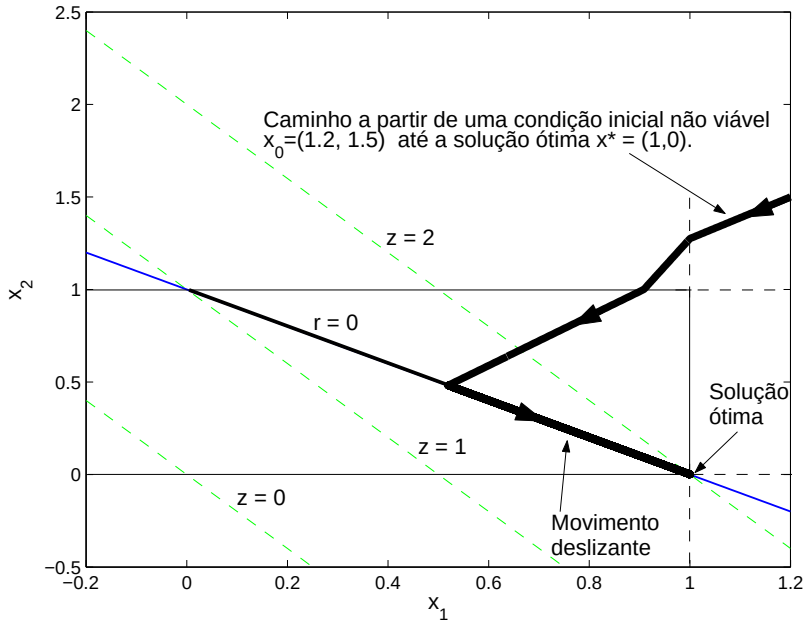


Figura 4.3: Trajetória partindo do ponto inicial $\mathbf{x}_0 = (1.2, 1.5)$ em direção à solução do PPL. A trajetória entra no hipercubo $[0,1]^2$ e atinge o hiperplano $\Pi := \{\mathbf{x} : \mathbf{1}^T\mathbf{x} = 1\}$. Um movimento deslizante é iniciado e a solução ótima do PPL é atingida. Neste exemplo $\mathbf{c} = (1,2)^T$ e $k = 1$.

conjunto viável, satisfazendo a restrição $\mathbf{x} \in [0,1]^n$, e prossegue até que atinja o hiperplano Π, quando um movimento deslizante é iniciado na direção que maximiza a função objetivo do PPL. Este movimento prossegue até que as restrições $x_1 \leq 1$ e $x_2 \geq 0$ são satisfeitas e permanecem ativas. Esta é a solução ótima do PPL (4.2); note que as restrições $x_1 \leq 1$ e $x_2 \geq 0$ evitam que a função objetivo $\mathbf{c}^T\mathbf{x}$ cresça indefinidamente.

A escolha da condição inicial pode ajudar a aumentar a velocidade de convergência. Em geral um ponto inicial pode ser encontrado escolhendo, por exemplo, $x_1 = \cdots = x_k = 1$ e $x_{k+1} = \cdots = x_n = 0$, que corresponde a um dos vértices do hipercubo $\Gamma := \{\mathbf{x} : \mathbf{x} \in [0,1]^n\}$ intersectado pelo hiperplano Π. Entretanto, as vantagens de iniciar em um ponto viável não são claras, pois este pode estar mais longe da solução ótima que um determinado ponto não-viável. Considere o exemplo mostrado na figura 4.3, onde o ponto viável $\mathbf{x} = (0,1)$ está mais longe do ponto ótimo $\mathbf{x}^* = (1,0)$ que o ponto não-viável $(0, 1.1)$.

4.3 Análise de convergência

Considerando o sistema (4.8), o conjunto Π, que é um hiperplano, divide $\mathbb{R}^n$ em dois subconjuntos, usualmente conhecidos como estruturas.

Nesta análise, mostramos uma propriedade importante do sistema gradiente (4.8), que é a convergência em tempo finito para o conjunto solução $\mathcal{X}$ do PPL (4.2), que significa que existe $\bar{t}$ tal que $\min_{\mathcal{X}} \|\mathbf{x}(t) - \mathbf{x}^*\| \to 0$, $\mathbf{x}^* \in \mathcal{X}$, quando $t \to \bar{t}$.

A análise de convergência do sistema descrito pela equação (4.8) é dividida em dois passos. Primeiro, obtemos condições suficientes que garantem a convergência para o conjunto viável do problema (4.2), que é dado pela seguinte intersecção:

$$\Omega := \Pi \cap \Gamma, \tag{4.10}$$

em que $\Pi := \{\mathbf{x} : \mathbf{1}^T\mathbf{x} - k = 0\}$ e $\Gamma := \{\mathbf{x} : x_j \in [0, 1], \text{ para cada } j\}$.

Uma vez obtida a convergência para o conjunto viável, é necessário provar que as trajetórias convergem para a solução do problema (4.2). Desta análise resultam condições suficientes que os parâmetros de penalidade γ e ρ devem satisfazer para garantir convergência em tempo finito das trajetórias da equação (4.8) para a solução $\mathcal{X}$ do PPL (4.2).

Os resultados de convergência são obtidos usando uma função de Lyapunov não suave V. Para provar convergência em tempo finito, é preciso mostrar que existe uma constante $\varepsilon > 0$, tal que $\dot{V} \leq -\varepsilon$ [35, 36].

Condições suficientes que garantem que as trajetórias do sistema (4.8) convergem para o conjunto viável do problema (4.2) em tempo finito são dadas pelo lema 4.1.

Lema 4.1. *Considere o sistema gradiente* (4.8). *Se os parâmetros de penalidade γ e ρ satisfazem as seguintes desigualdades:*

$$\gamma > \|\mathbf{c}\| \tag{4.11}$$

$$\rho > \frac{1}{\sqrt{n}}\|\mathbf{c}\| + 2\,\gamma, \tag{4.12}$$

então, para qualquer condição inicial, as trajetórias atingem o conjunto viável Ω, definido em (4.10)*, em tempo finito e permanecem neste conjunto.*

Prova: Ver seção 4.6. ■

O lema 4.1 estabelece condições de convergência simples, dadas pelas desigualdades (4.11) e (4.12). Note que os limites inferiores que γ e ρ devem satisfazer são fáceis de calcular ou estimar, e dependem apenas da norma euclideana do vetor $\mathbf{c}$. Os limites dados pelas desigualdades (4.11) e (4.12) são pequenos, e quanto maior a dimensão n do vetor $\mathbf{c}$, menor é o limite inferior dado por (4.12), o que é uma vantagem importante quando se considera a implementação prática do circuito proposto.

Para o PPL na forma padrão, as condições de convergência obtidas por Chong et al. [15] dependem das variáveis de estado da rede, enquanto que os limites dados em (4.11) e (4.12) são independentes das trajetórias. Devido à dependência das trajetórias das condições de convergência obtidas em [15], Zhang [23, pag. 323] afirma que a rede proposta por Chong et al. [15] "precisa de um computador externo, ou alternativamente, uma subrede adicional para calcular os parâmetros de penalidade em tempo real", o que provoca o aumento da complexidade da rede.

Por outro lado, como os limites dados por (4.11) e (4.12) dependem apenas da norma euclideana do vetor $\mathbf{c}$ e de sua dimensão, o ajuste dos parâmetros γ e ρ do circuito formulado pelo sistema gradiente (4.8), não depende dos estados do sistema. Como resultado, não são necessários sistemas externos para calcular os parâmetros γ e ρ, já que eles podem ser determinados *offline*, o que garante uma simplificação significativa da implementação prática da rede descrita por (4.8).

O lema 4.1 garante que, para parâmetros γ e ρ satisfazendo as desigualdades (4.11) e (4.12), as trajetórias do sistema atingem o conjunto viável do PPL (4.2), mas ainda é necessário provar dois resultados adicionais – i) se γ e ρ satisfazem o lema 4.1, então o PPL (4.2), e o problema sem restrições associado (4.3), possuem a mesma solução $\mathbf{x}^*$; ii) se γ e ρ satisfazem o lema 4.1, então as trajetórias do sistema convergem para a solução do PPL (4.2). Este procedimento é necessário pois o lema 4.1 não garante a convergência para a solução do PPL (4.2), mas apenas para o conjunto viável Ω.

Lema 4.2. *Se os parâmetros de penalidade γ e ρ satisfazem o lema 4.1, então o PPL* (4.2) *e o problema sem restrições* (4.3) *possuem a mesma solução.*

Prova: Ver seção 4.6. ■

A estratégia usada para provar o lema 4.2 foi utilizada por Chong et al. [15] para provar um resultado similar para PPLs na forma padrão. Seguindo os passos em [15], resta mostrar

que as trajetórias do sistema gradiente (4.8) convergem para a solução do PPL, desde que as condições do lema 4.1 sejam verificadas. Este resultado é enunciado a seguir e provado na seção 4.6.

Lema 4.3. *Se os parâmetros de penalidade γ e ρ satisfazem o lema 4.1, então as trajetórias do sistema gradiente* (4.8) *convergem para a solução do PPL* (4.2) *em tempo finito e permanecem neste conjunto.*

Prova: Ver seção 4.6. ■

Os três lemas anteriores completam a análise de convergência. O lema 4.2 garante que o minimizador da função de energia computacional E, dada em (4.3), é o maximizador do PPL (4.2), e o lema 4.3, garante que as trajetórias do sistema (4.8) convergem para a solução do PPL. Estes resultados, juntamente com a proposição 2 e o lema 4.1, garantem que a rede descrita pelo sistema gradiente (4.8) é uma rede "*k*-winners-take-all". Estes são os elementos necessários para enunciar o teorema 4.1, que é o principal resultado deste capítulo.

Teorema 4.1. *Considere o circuito descrito pelo sistema gradiente* (4.8)*, e suponha que os ganhos da rede γ eρ satisfazem os limites* (4.11) *e*(4.12)*, estabelecidos no lema 4.1. Então, dado um vetor $\mathbf{c} \in \mathbb{R}^{n\times 1}$, com todos os componentes distintos, e um inteiro positivo $k \leq n$, a rede descrita por* (4.8) *é uma rede KWTA.* □

O teorema 4.1 garante que a rede KWTA proposta é capaz de selecionar k vencedores de um dado vetor $\mathbf{c}$. Com base no fato que a solução do PPL (4.2) satisfaz a proposição 2, a convergência das trajetórias de (4.8) para a solução deste problema, garante ao circuito proposto a propriedade KWTA.

Os resultados obtidos nesta seção também garantem que a propriedade KWTA do sistema proposto não depende da escolha das condições iniciais, isto é, não é necessário encontrar um ponto inicial dentro do conjunto viável. Isto é ilustrado no exemplo da figura 4.3, onde a convergência para a solução correta partindo de um ponto não viável é obtida. Pelo lema 4.3, as trajetórias não saem da intersecção $\Gamma \cap \Pi$, que é o conjunto viável do PPL, e o circuito busca a solução KWTA apenas dentro do conjunto viável.

4.4 Análise de complexidade

Nesta seção, utilizamos as definições de complexidade dadas em [74], que possibilitam a comparação do circuito formulado pelo sistema gradiente (4.8), com outras redes para a resolução de PPLs em termos de complexidade. A seguir, reproduzimos as definições de complexidade dadas em [74].

Definição 4.1 (Complexidade do neurônio). A complexidade de cada neurônio é o número de multiplicações/divisões e adições/subtrações efetuadas por cada neurônio em cada iteração.

Definição 4.2 (Complexidade do modelo). É o número total de multiplicações/divisões e adições/subtrações efetuadas pela rede em cada iteração.

A notação Θ *[75]*: Dada uma função $g(n) : \mathbb{N} \to \mathbb{R}$, sendo que $\mathbb{N}$ denota o conjunto dos números naturais, representa-se por $\Theta(g(n))$ o seguinte conjunto de funções:

$$\Theta(g(n)) := \{f(n) : \exists\, c_1, c_2, n_0 \text{ constantes; } 0 \leq c_1\, g(n) \leq f(n) \leq c_2\, g(n), \forall n \geq n_0\}.$$

No conjunto Θ, para cada $n \geq n_0$, $f(n)$ é limitada por $g(n)$, dentro de um fator constante. A função $g(n)$ é um limite assintótico para $f(n)$, e a complexidade dada em termos de Θ é chamada de *complexidade assintótica* [75].

De acordo com Żak et al. [74], a distinção entre adições e subtrações e multiplicações e divisões nas definições acima é necessária, pois elas podem ter efeito considerável no custo da implementação. A definição de complexidade do modelo é útil em implementações práticas, e para implementação utilizando hardware, a complexidade está relacionada ao número total de multiplicações/divisões e adições/subtrações [74].

No circuito proposto, o i-ésimo neurônio é modelado pela i-ésima equação do sistema gradiente (4.8). Cada neurônio deste circuito efetua 2 adições e 3 multiplicações por iteração e, pela definição 4.1, a complexidade do neurônio é 5, resultando em complexidade assintótica do neurônio igual a $\Theta(1)$. Por outro lado, a complexidade do modelo é $3n + 2$, e a complexidade do modelo é igual a $\Theta(n)$.

O circuito proposto por Chong et al. [15] aplicado ao PPL (4.2) apresenta complexidade assintótica do neurônio igual a $\Theta(n)$ e a complexidade assintótica do modelo é $\Theta(n^2)$, que

são maiores que as complexidades assintóticas correspondentes do sistema (4.8). Isto ocorre porque a complexidade deste circuito é aumentada pela tarefa adicional de determinar os parâmetros de penalidade em tempo real. O circuito descrito por (4.8) apresenta menor complexidade porque os parâmetros de penalidade são calculados offline e apenas uma vez.

O circuito modelado por (4.8) apresenta também menor complexidade que os propostos por Kennedy e Chua [76] e Rodríguez-Vázquez et al. [13], ambos apresentam complexidade assintótica do neurônio igual a $\Theta(n)$ e complexidade assintótica do modelo igual a $\Theta(n^2)$ quando aplicados ao PPL (4.2). Isto mostra que os custos de implementação do circuito proposto neste trabalho, são menores que a dos propostos por Chong et al. [15], Kennedy e Chua [16] e Rodríguez-Vázquez et al. [13], o que é uma vantagem do sistema gradiente (4.8), do ponto de vista de implementação.

4.5 Exemplos numéricos

A presença de funções descontínuas no segundo membro do sistema (4.8) provocam a existência do fenômeno conhecido como *chattering*. Quando as trajetórias convergem para o conjunto viável do PPL, onde o segundo membro de (4.8) é descontínuo, as trajetórias iniciam um "chaveamento" em uma vizinhança da superfície de descontinuidade. O chattering ocorre devido a erros numéricos de integração e sua amplitude pode ser reduzida através da escolha adequada dos ganhos γ e ρ.

Exemplo 4.5.1. Considere o vetor abaixo:

$$\mathbf{c} = \begin{pmatrix} 1.23 & 1.32 & 1.25 & 1.47 \end{pmatrix},$$

do qual desejamos determinar os dois maiores componentes, ou seja, $k = 2$.

Pelo teorema 4.1 os limites que os parâmetros de penalidade ou, equivalentemente, ganhos da rede devem satisfazer para garantir a convergência global são $\gamma > 2.64$ e $\rho > 6.60$. Sejam $\gamma = 2.70$ e $\rho = 7.00$.

Os resultados da simulação são mostrados na figura 4.4, com condições iniciais escolhidas arbitrariamente na origem. Os componentes $x_2 = x_4 = 1$ correspondem aos dois maiores componentes do vetor $\mathbf{c}$, que são os vencedores, e os demais componentes, $x_1 = x_3 = 0$, correspondem aos perdedores.

Neste exemplo, com o objetivo de acelerar a convergência, usamos um fator de escala de tempo $\mathbf{M} = 10\mathbf{I}$, escolhido empiricamente. Outras técnicas para escolher o fator μ são discutidas em [12, seção 3.1.4].

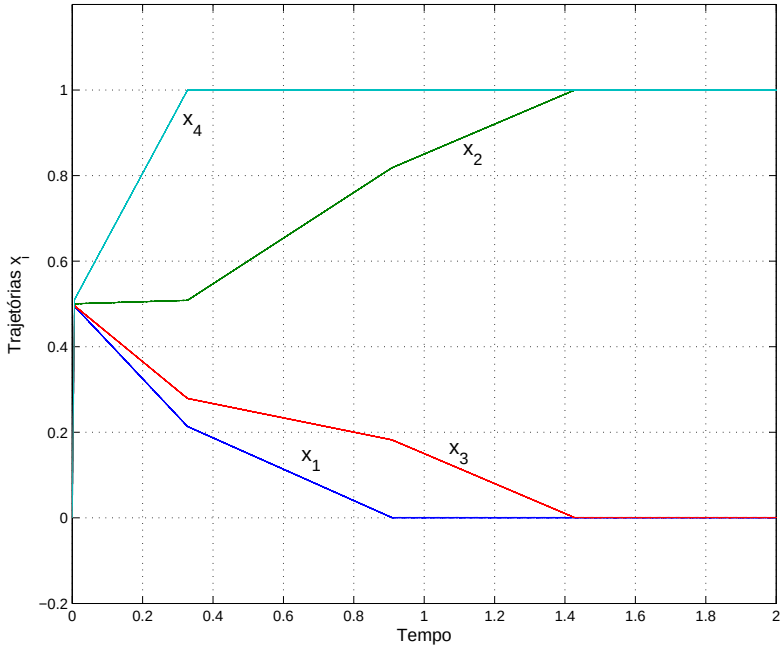


Figura 4.4: Trajetórias do exemplo (4.5.1), mostrando a correta discriminação dos dois vencedores.

Exemplo 4.5.2. (Adaptado de [70, pag. 777]) Sejam $n = 10$ e $k = 3$, sendo que $c_j = j/2$ Volts, para $j = 1, \ldots, 9$, e as saídas do circuito são avaliadas para c_{10} variando de 0 a 5 Volts. Urahama e Nagao observam que, para este exemplo, o circuito por eles proposto tem resolução limite de 0.5 Volts e se a diferença entre o menor vencedor e o maior perdedor for menor que este limite, então circuito de Urahama e Nagao não é capaz de discriminar os vencedores e os perdedores. Para produzir um exemplo em que o circuito de Urahama e Nagao não é capaz de discriminar vencedores e perdedores, seja $c_{10} = 3.3$ Volts, isto é:

$$\mathbf{c} = \begin{pmatrix} 0.5 & 1 & 1.5 & 2 & 2.5 & 3 & 3.5 & 4 & 4.5 & 3.3 \end{pmatrix}$$

Os vencedores são c_7, c_8 e c_9 e a diferença entre o menor vencedor e o maior perdedor é $c_7 - c_{10} = 0.2$, que é menor que o limite de resolução observado por Urahama e Nagao [70] para este exemplo. Pelo lema 4.1, obtemos $\gamma > 9.06$ e $\rho > 20.99$. Sejam $\gamma = 9.1$ e $\rho = 22.0$ os ganhos da rede. Do mesmo modo que no exemplo anterior, para aumentar a taxa

de convergência, usamos um fator de escala de tempo $\mu = 10$. Os resultados da simulação são mostrados na figura 4.5.

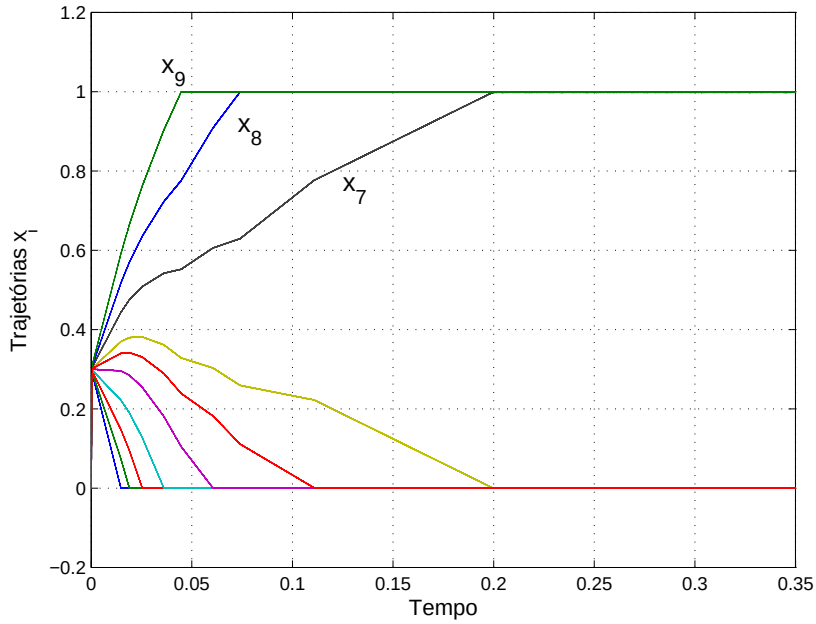


Figura 4.5: Trajetórias do exemplo (4.5.2), mostrando a correta discriminação dos três maiores elementos do vetor dado.

Note que, para este exemplo, o sistema gradiente (4.8) é capaz de discriminar os vencedores e perdedores no vetor dado, o que não ocorre com o circuito de Urahama & Nagao, indicando que o sistema gradiente (4.8) possui maior resolução.

4.6 Provas dos resultados de convergência

Prova da proposição 2: Seja $\mathcal{C}$ o conjunto dos k maiores componentes do vetor $\mathbf{c}$ e defina os conjuntos de índices $I := \{i : c_i \in \mathcal{C}\}$ e $J := \{i : c_i \notin \mathcal{C}\}$. Sem perda de generalidade, sejam $c_1, c_2, \ldots, c_k$ os k maiores componentes do vetor $\mathbf{c}$.

O conjunto viável do PPL (4.2) corresponde a um hiperplano $\Pi := \{\mathbf{x} : \sum_{p=1}^{n} x_p = k\}$ que intersecta o hipercubo definido por $\Gamma := \{\mathbf{x} : x_p \in [0, 1], p = 1, \ldots, n\}$. Os vértices do hipercubo Γ que são intersectados pelo hiperplano Π são apenas aqueles que apresentam k componentes iguais a um e os demais $n - k$ componentes iguais a zero. Adicionalmente, nenhuma aresta do hipercubo é intersectada pelo hiperplano Π. Isto ocorre pois cada ponto em uma aresta do hipercubo Γ é descrito por um componente $x_q \in]0, 1[$, para algum índice q, e os demais $n - 1$ componentes são $x_p \in \{0, 1\}$, para cada $p \neq q$. Então, para estes pontos, a

soma $\sum_{p=1}^{n} x_p$ não resulta em um valor inteiro. Assim, para cada $k \in \mathbb{N}$ o hiperplano Π não intersecta nenhuma aresta do hipercubo Γ.

Como os coeficientes c_p, $p = 1, \ldots, n$ são todos distintos, $\mathbf{c}$ não é um vetor normal ao hiperplano Π, implicando que a única solução de (4.2) é um vértice do hipercubo Γ intersectado pelo hiperplano Π. Então, a solução de (4.2) é um vetor de k uns e $n - k$ zeros. Conseqüentemente, dado que $c_i > c_j$, para todo $i \in I$ e $j \in J$, é imediato que o vetor que maximiza z em (4.2), satisfazendo todas as restrições do PPL é $x_i^* = 1$, para $i \in I$ e $x_j^* = 0$, para $j \in J$. Então, o vetor $\mathbf{x}^*$ que resolve o PPL (4.2) é tal que seus k componentes iguais a 1 correspondem aos k maiores componentes do vetor $\mathbf{c}$ e os demais componentes são iguais a zero, concluindo a prova. ■

Prova do lema 4.1: Suponha que $\mathbf{x} \notin \Pi$ e $x_i \notin [0, 1]$, para algum i. Premultiplicando (4.8) pelo vetor linha $\mathbf{1}^T$ e notando que $\dot{r} = \mathbf{1}^T \dot{\mathbf{x}}$ obtemos,

$$\dot{r} = \mathbf{1}^T \mathbf{c} - \gamma \mathbf{1}^T \mathbf{h}(\mathbf{x}) - \rho \mathbf{1}^T \mathbf{1} \text{sgn}(r). \tag{4.13}$$

Escrevendo as equações (4.8) e (4.13) em notação vetorial, obtemos o seguinte sistema de referência da forma Persidskii [50]:

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \mathbf{1}^T \mathbf{c} \end{pmatrix} - \begin{pmatrix} \mathbf{I}_n & \mathbf{1} \\ \mathbf{1}^T & \mathbf{1}^T \mathbf{1} \end{pmatrix} \begin{pmatrix} \gamma\, \mathbf{I}_n & \mathbf{0}_{n \times 1} \\ \mathbf{0}_{1 \times n} & \rho \end{pmatrix} \begin{pmatrix} \mathbf{h}(\mathbf{x}) \\ \text{sgn}(r) \end{pmatrix} \tag{4.14}$$

sendo que $\mathbf{I}_n$ denota a matriz identidade de ordem n.

Considere a seguinte função de Lyapunov candidata do tipo diagonal associada ao sistema Persidskii (4.14):

$$V(\mathbf{x}, r) = \gamma \sum_{j=1}^{n} \int_0^{x_j} h_j(\tau) d\tau + \rho \int_0^{r} \text{sgn}(\tau) d\tau. \tag{4.15}$$

Como a função $V(\mathbf{x}, r)$ é Lipschitz contínua em $(\mathbf{x}, r)$ e $\mathbf{x}(t)$ e $r(t)$ são absolutamente contínuas, então V também é absolutamente contínua em relação à variável t [54].

A derivada no tempo de (4.15) ao longo das trajetórias do sistema (4.14) é:

$$\begin{aligned}\dot{V} &= \gamma \mathbf{h}^T(\mathbf{x})\ \dot{\mathbf{x}} + \rho \text{sgn}(r)\ \dot{r} \\ &= \gamma \mathbf{h}^T(\mathbf{x})(\mathbf{c} - \gamma \mathbf{h}(\mathbf{x}) - \rho \mathbf{1} \text{sgn}(r)) + \rho \text{sgn}(r)(\mathbf{1}^T\mathbf{c} - \gamma \mathbf{1}^T\mathbf{h}(\mathbf{x}) - \rho \mathbf{1}^T\mathbf{1} \text{sgn}(r)) \\ &= \gamma \mathbf{h}^T(\mathbf{x})\mathbf{c} - \gamma^2 \|\mathbf{h}(\mathbf{x})\|^2 - \gamma\rho\ \mathbf{h}^T(\mathbf{x})\mathbf{1}\text{sgn}(r) + \rho \text{sgn}(r)\mathbf{1}^T\mathbf{c} - \gamma\rho \text{sgn}(r)\mathbf{1}^T\ \mathbf{h}(\mathbf{x}) - \\ &\rho^2 \text{sgn}(r)^2 \mathbf{1}^T\mathbf{1}.\end{aligned}$$

Usando a desigualdade de Schwarz:

$$\begin{aligned}\dot{V} \leq & \gamma \|\mathbf{h}(\mathbf{x})\|\ \|\mathbf{c}\| + \rho |\text{sgn}(r)|\ \|\mathbf{1}\|\ \|\mathbf{c}\| - \gamma^2 \|\mathbf{h}(\mathbf{x})\|^2 + \\ & \gamma\rho |\text{sgn}(r)|\ \|\mathbf{1}\|\ \|\mathbf{h}(\mathbf{x})\| + \gamma\rho |\text{sgn}(r)|\ \|\mathbf{1}\|\ \|\mathbf{h}(\mathbf{x})\| - \rho^2 \text{sgn}(r)^2 \mathbf{1}^T\mathbf{1}.\end{aligned}$$

Como, por hipótese, $r \neq 0$, então $|\text{sgn}(r)| = 1$ e notando que $\|\mathbf{1}\| = \sqrt{n}$, e $\mathbf{1}^T\mathbf{1} = n$, a última desigualdade toma a forma:

$$\begin{aligned}\dot{V} &\leq \gamma \|\mathbf{h}(\mathbf{x})\|\ \|\mathbf{c}\| + \rho\sqrt{n}\|\mathbf{c}\| - \gamma^2 \|\mathbf{h}(\mathbf{x})\|^2 + 2\gamma\rho\sqrt{n}\|\mathbf{h}(\mathbf{x})\| - \rho^2\ n \\ &= -\gamma \|\mathbf{h}(\mathbf{x})\|(\gamma \|\mathbf{h}(\mathbf{x})\| - \|\mathbf{c}\|) - \rho(\rho\ n - 2\gamma\sqrt{n}\|\mathbf{h}(\mathbf{x})\| - \sqrt{n}\|\mathbf{c}\|).\end{aligned}$$

Como $x_i \notin [0, 1]$, para algum $_i$, então $|h_i(x_i)| = 1$, daí decorre que $\|\mathbf{h}(\mathbf{x})\| \in [1, \sqrt{n}]$. Além disso, como $\gamma > 0$ e $\rho > 0$, a última desigualdade pode ser escrita como:

$$\begin{aligned}\dot{V} &\leq -\gamma \|\mathbf{h}(\mathbf{x})\|(\gamma \|\mathbf{h}(\mathbf{x})\| - \|\mathbf{c}\|) - \rho(\rho\ n - 2\gamma\sqrt{n}\|\mathbf{h}(\mathbf{x})\| - \sqrt{n}\|\mathbf{c}\|) \\ &\leq -\gamma(\gamma - \|\mathbf{c}\|) - \rho(\rho n - 2\gamma n - \sqrt{n}\|\mathbf{c}\|).\end{aligned}$$

Para que $\dot{V}$ seja definida negativa, é necessário que os termos entre parênteses na última desigualdade sejam positivos. Logo, para γ e ρ escolhidos de tal modo que as seguintes desigualdades sejam satisfeitas:

$$\begin{aligned}\gamma &> \|\mathbf{c}\|, \\ \rho &> \frac{1}{\sqrt{n}}\ \|\mathbf{c}\| + 2\gamma,\end{aligned}$$

temos que $\dot{V} < -\varepsilon$, em que o escalar $\varepsilon = \gamma(\gamma - \|\mathbf{c}\|) + \rho(\rho n - 2\gamma n - \sqrt{n}\|\mathbf{c}\|)$ é positivo e

constante.

Logo, para γ e ρ satisfazendo as duas últimas desigualdades, as trajetórias do sistema (4.8) convergem para o conjunto viável do PPL (4.2) em tempo finito.

Resta mostrar que uma vez que as trajetórias convergem para o conjunto viável do PPL, elas permanecem neste conjunto. Provamos este resultado por contradição. Seja T o instante em que $V(\mathbf{x}(T)) = 0$, e suponha que existe um instante $T_s \geq T$ tal que $V(\mathbf{x}(T_s)) > 0$. Como V é absolutamente contínua, então podemos escrever

$$\int_{T}^{T_s} \dot{V}(\mathbf{x}(t))dt = V(\mathbf{x}(T_s)) - V(\mathbf{x}(T)) = V(\mathbf{x}(T_s)) > 0,$$

conseqüentemente $\dot{V} > 0$, o que é uma contradição, pois provamos acima que $\dot{V} < 0$. Logo, não existe $T_s \geq T$ tal que $V(\mathbf{x}(T_s)) > 0$.

Portanto, as trajetórias do sistema gradiente (4.8) convergem em tempo finito para o conjunto viável do PPL (4.2) e permanecem neste conjunto indefinidamente. ■

Prova do lema 4.2: Como E é convexa para todo $\mathbf{x}$, qualquer mínimo de E é, de fato, um mínimo global. Precisamos provar apenas que os minimizadores de E pertencem ao conjunto Ω, pois para todo $\mathbf{x} \in \Omega$, $E(\mathbf{x})$ é reduzida a $-\mathbf{c}^T\mathbf{x}$, sendo que Ω é definido em (4.10).

Provamos o lema por contradição. Seja $\mathbf{x}^* \notin \Omega$ um minimizador de E e suponha que γ e ρ satisfazem as desigualdades (4.11) e (4.12) do lema 4.1, então $\mathbf{0} \in \partial E(\mathbf{x}^*)$. Conseqüentemente, existe pelo menos uma trajetória do sistema (4.8) que não converge para Ω, o que é uma contradição, pois pelo lema 4.1 todas as trajetórias de (4.8) convergem para Ω, em tempo finito e permanecem neste conjunto. Logo $\mathbf{x}^* \in \Omega$, concluindo a prova. ■

Prova do lema 4.3: Sejam γ e ρ satisfazendo o lema 4.1, então as trajetórias de (4.8) convergem para o conjunto viável Ω em tempo finito. Considere que $\mathbf{x} \in \Omega$.

Considere a seguinte função de Lyapunov candidata:

$$V(\mathbf{x}) = E(\mathbf{x}) - E(\mathbf{x}^*),$$

sendo que $\mathbf{x}^*$ é a solução do PPL (4.2).

Como, por hipótese, $\mathbf{x} \in \Pi$ e $\mathbf{x} \in \Gamma$, com Π e Γ definidos em (4.10), consideramos duas

possibilidades:

i) Se $\mathbf{x}$ é interior ao conjunto Γ, isto é $x_j > 0$ e $x_j < 1$ para todo j. Conseqüentemente, $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ e a dinâmica em modo deslizante (4.9) é reduzida a:

$$\dot{\mathbf{x}} = \mathbf{P}\mathbf{c}.$$

Para todo $\mathbf{x}$ no interior de Γ, a função de Lyapunov candidata V, definida acima, é reduzida a:

$$V(\mathbf{x}) = -(\mathbf{c}^T\mathbf{x} - \mathbf{c}^T\mathbf{x}^*).$$

A derivada no tempo de V ao longo das trajetórias da equação $\dot{\mathbf{x}} = \mathbf{P}\mathbf{c}$ é dada por:

$$\dot{V} = -\nabla^T V(\mathbf{x})\ \dot{\mathbf{x}} = -\mathbf{c}^T\mathbf{P}\mathbf{c} = -\|\mathbf{P}\mathbf{c}\|_2^2 < 0, \tag{4.16}$$

note que se $\mathbf{P}\mathbf{c} = \mathbf{0}$, então $\dot{\mathbf{x}} = \mathbf{0}$ e qualquer $\mathbf{x}$ viável, é uma solução ótima.

ii) Se $\mathbf{x}$ não está no interior de Γ, então $x_i = 1$ ou $x_i = 0$, para algum i, gerando um movimento deslizante ao longo da fronteira do conjunto Γ. A derivada no tempo da função de Lyapunov candidata toma a forma:

$$\dot{V} = -\|\mathbf{P}(\mathbf{c} - \gamma\ \mathbf{h}(\mathbf{x})\|_2^2,$$

sendo que, cada componente do vetor $\mathbf{h}(\mathbf{x})$, que é um subgradiente da função de penalidade $\sum_{i=1}^{n} x_j^+ - \sum_{j=1}^{n} \min(0, x_j)$, é dado por $h_i(x_i) \in [-1, 0]$, se $x_i = 0$, $h_i(x_i) \in [0, 1]$, se $x_i = 1$, e $h_i(x_i) = 0$, se x_i está no interior de Γ.

Se $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, então temos o mesmo caso analisado no ítem i). Seja $\mathbf{h}(\mathbf{x}) \neq \mathbf{0}$.

Note que, $\mathbf{P}(\mathbf{c} - \gamma\ \mathbf{h}) = \mathbf{0}$ se e somente se $\dot{\mathbf{x}} = \mathbf{0} \in \partial E(\mathbf{x})$, que corresponde ao conjunto solução do problema sem restrições (4.3). Logo, para $\mathbf{x}$ não pertencente ao conjunto solução de (4.3), tem-se que $\mathbf{P}(\mathbf{c} - \gamma\ \mathbf{h}) \neq \mathbf{0}$.

Como $\mathbf{h} \in [-1, 1]^n$, então o vetor $\mathbf{P}(\mathbf{c} - \gamma\mathbf{h})$ é limitado e, como $\mathbf{x}$ não pertence ao conjunto solução do problema (4.3), é não-nulo. Como a função $\|\cdot\|_2^2$ é contínua,

então, pelo teorema de Weierstrass, existe um número real ε tal que

$$\varepsilon = \min \|\mathbf{P}(\mathbf{c} - \gamma\mathbf{h})\|_2^2 \neq 0,$$

para qualquer $\mathbf{x} \in \Omega$ e que não pertence ao conjunto solução do problema sem restrições (4.3). Logo, por (4.16), tem-se que $\dot{V} \leq -\varepsilon$.

Pelos ítens i) e ii), as trajetórias de (4.8) convergem, em tempo finito, para a solução do problema sem restrições (4.3), que pelo lema 4.2 é também solução do problema de programação linear (4.2). ■

Síntese do capítulo

Neste capítulo, um sistema gradiente capaz de determinar os k maiores componentes de um dado vetor é proposto e analisado. O sistema é obtido a partir de um problema de programação linear com variáveis limitadas, e a utilização de um método de penalização exata, resulta em um sistema gradiente que converge para a solução do PPL.

Da análise de convergência resultam condições suficientes que os parâmetros de penalidade devem satisfazer para que as trajetórias do sistema convirjam para a solução KWTA em tempo finito. Exemplos numéricos foram apresentados, mostrando a eficácia do sistema proposto.

No próximo capítulo, sistemas gradientes são utilizados para o treinamento de *support vector machines*, apresentando uma análise teórica dos sistemas propostos. São apresentados oito exemplos numéricos, sendo que um deles os sistemas gradientes são resolvidos em paralelo.

Capítulo 5

Treinamento de support vector machines utilizando sistemas gradientes

Neste capítulo, sistemas gradientes não-suaves são utilizados para o treinamento de *support vector machines* (SVMs). O SVM original foi introduzido por Cortes e Vapnik [77], e desde então diversas modificações a este modelo foram propostas, como *least squares SVMs* (LS-SVM) [78] e ν-SVM [79].

O SVM é uma técnica de classificação formulada matematicamente através de um problema convexo de programação quadrática com restrições lineares [80]. Esta é uma das principais vantagens do SVM, pois existe um único mínimo global, e mostramos neste trabalho que um minimizador pode ser determinado eficientemente através de um sistema gradiente. Além disso, o SVM apresenta notável capacidade de generalização [77].

Redes neurais recorrentes foram usadas em trabalhos anteriores para o treinamento de SVMs [81, 82]. A idéia é tirar vantagem das propriedades das redes neurais, que as tornam adequadas à implementação através de hardware, utilizando tecnologia VLSI, para processamento em tempo real [12]. Como as redes neurais recorrentes são formuladas matematicamente por sistemas de equações diferenciais ordinárias (EDOs), estas são também adequadas para implementação digital usando softwares para integração de EDOs.

Uma implementação de um SVM através de circuitos foi apresentada em [81], utilizando a rede proposta por Kennedy e Chua [16] com funções de penalidade suaves. Tan et al. [82], propuseram uma rede neural para treinamento de SVMs para reconhecimento de padrões. Anguita et al. [83] desenvolveram uma arquitetura digital para treinamento de SVMs através

de um sistema dinâmico, apresentando uma implementação utilizando *field programmable array*.

Neste capítulo, abordamos o problema de discriminação linear e não-linear de duas classes utilizando sistemas gradientes não-suaves, obtidos a partir do método de penalização exata, descrito no capítulo 3. O primeiro sistema é desenvolvido para a discriminação linear de duas classes: diferentemente dos modelos propostos por Anguita et al. [81] e Tan et al. [82], que são baseados no problema dual, o sistema proposto é obtido a partir da formulação primal do SVM. O segundo sistema é formulado para a discriminação não-linear de duas classes e é baseado na formulação ν-SVM [79]; este sistema é baseado no problema dual, pois é preciso utilizar funções de kernel para a separação não-linear das classes.

A análise de convergência dos dois modelos propostos é feita utilizando a forma Persidskii dos sistemas gradientes e o resultado geral de convergência para sistemas Persidskii, estabelecido pelo teorema 3.2. A convergência é global, e independe do ajuste dos parâmetros de penalidade e dos parâmetros dos SVMs correspondentes.

Os sistemas gradientes obtidos podem ser implementados através de circuitos analógicos utilizando apenas resistores, amplificadores, capacitores e switches [12], sendo apropriados para aplicações em tempo real utilizando tecnologia VLSI. Como estes sistemas dependem da integração de sistemas de EDOs, são também adequados para implementação utilizando softwares padrão para integração de equações diferenciais. A escalabilidade dos sistemas propostos é uma vantagem, pois estes sistemas podem ser facilmente implementados em computadores paralelos, atingindo taxas de convergência mais altas que algoritmos já consagrados para treinamento de SVMs, viabilizando o uso de conjuntos de dados de dimensões elevadas.

5.1 SVM para separação linear

O problema de classificar os elementos de um determinado conjunto, denominado *espaço de entrada*, em duas classes A e B, é conhecido como *classificação binária*. A separação das classes é feita através de hipersuperfícies, e o problema de determinar a melhor superfície que separa estas classes é conhecido como *treinamento*, que pode ser resolvido eficientemente através de SVMs [77]. Se a superfície separadora ótima for um hiperplano, e a separação é feita sem erros, então as classes A e B são ditas *linearmente separáveis*.

O subconjunto do espaço de entrada utilizado para obter a superfície separadora é chamado de *conjunto de treinamento*.

Considere o seguinte conjunto de m pares de treinamento, não necessariamente linearmente separáveis:

$$(y_1, \mathbf{z}_1),\ (y_2, \mathbf{z}_2), \ldots, (y_m, \mathbf{z}_m),\ y_i \in \{-1, 1\},\ i = 1 \ldots, m, \tag{5.1}$$

sendo que os vetores $\mathbf{z}_i \in \mathbb{R}^n$ são elementos do espaço de entrada, e cada escalar y_i define a posição de cada vetor $\mathbf{z}_i$ relativo à superfície separadora.

Sejam as classes A e B, identificadas como $y_i = 1$ se $\mathbf{z}_i \in A$ e $y_i = -1$ se $\mathbf{z}_i \in B$, respectivamente. Considere o problema de encontrar o melhor hiperplano que separa as classes A e B. Se A e B não são linearmente separáveis, o SVM pode ser formulado pelo seguinte problema de programação quadrática com restrições lineares [84, pag.. 104]:

$$\begin{aligned} &\text{minimize}_{\mathbf{w},\boldsymbol{\xi},b}\ \frac{1}{2}\left(\mathbf{w}^T\mathbf{w} + c\sum_{i=1}^{m}\xi_i^2\right) \\ &\text{sujeito a} \quad y_i(\mathbf{w}^T\mathbf{z}_i + b) \geq 1 - \xi_i,\ y_i \in \{-1, 1\}, \end{aligned} \tag{5.2}$$

sendo que m é o número de elementos no conjunto de treinamento, $\mathbf{w} \in \mathbb{R}^n$ é o vetor de pesos, $\mathbf{z}_i \in \mathbb{R}^n$, $\xi_i \in \mathbb{R}$ são as variáveis de folga, b é o bias e c é um parâmetro a ser escolhido.

O vetor de pesos $\mathbf{w}$ é normal ao hiperplano que separa as classes A e B, e as variáveis de folga ξ_i são introduzidas para proporcionar tolerância a classificações erradas. Caso este modelo seja aplicado a uma base de dados separável, as variáveis de folga são anuladas na solução do problema (5.2).

Como o escalar b também é otimizado, definimos os seguinte vetores aumentados:

$$\mathbf{u} := (\mathbf{w}^T \vdots b)^T,\ \bar{\mathbf{z}}_i := (\mathbf{z}_i^T \vdots 1)^T, \tag{5.3}$$

em que $\mathbf{u} \in \mathbb{R}^{n+1}$ e $\bar{\mathbf{z}}_i \in \mathbb{R}^{n+1}$.

Utilizando os vetores aumentados definidos em acima, o problema de otimização (5.2)

toma a forma

$$\begin{aligned} &\text{minimizar}_{\mathbf{u},\boldsymbol{\xi}} \ \frac{1}{2}\left(\mathbf{u}^T\mathbf{u} + c\sum_{i=1}^{m}\xi_i^2\right) \\ &\text{sujeito a} \quad y_i(\mathbf{u}^T\bar{\mathbf{z}}_i) \geq 1-\xi_i,\ y_i \in \{-1,1\}. \end{aligned} \tag{5.4}$$

Pelo método da função de penalidade, obtemos o seguinte problema sem restrições associado ao problema (5.4):

$$\text{minimizar } E(\mathbf{u},\boldsymbol{\xi}) = \frac{1}{2}\left(\mathbf{u}^T\mathbf{u} + c\sum_{i=1}^{m}\xi_i^2\right) - \rho\sum_{i=1}^{m}\min(0,r_i), \tag{5.5}$$

no qual $r_i = y_i(\bar{\mathbf{z}}_i^T\mathbf{u}) + \xi_i - 1$. No contexto de redes neurais, a função objetivo E é conhecida como uma *função de energia computacional*.

A matriz a seguir é definida para reescrever as restrições do problema (5.2):

$$\mathbf{Z} := \begin{pmatrix} y_1\bar{\mathbf{z}}_1^T \\ \vdots \\ y_m\bar{\mathbf{z}}_m^T \end{pmatrix} \in \mathbb{R}^{m\times(n+1)}, \tag{5.6}$$

sendo que os vetores $\bar{\mathbf{z}}_i$ são definidos em (5.3).

Por compacidade de notação, definimos o seguinte vetor de resíduos:

$$\mathbf{r} := \mathbf{Z}\mathbf{u} + \boldsymbol{\xi} - \mathbf{1}, \tag{5.7}$$

sendo que $\boldsymbol{\xi} := (\xi_1,\ldots,\xi_m)$ é o vetor de variáveis de folga e $\mathbf{1}$ é o vetor-coluna de uns de dimensão m.

O sistema gradiente associado à função de energia E é dado por:

$$\dot{\mathbf{u}} = -\mathbf{M}[\mathbf{u} + \rho\mathbf{Z}^T\text{hsgn}(\mathbf{r})] \tag{5.8}$$

$$\dot{\boldsymbol{\xi}} = -\mathbf{M}[c\,\boldsymbol{\xi} + \rho\text{hsgn}(\mathbf{r})], \tag{5.9}$$

sendo que $\mathbf{M} := \text{diag}(\boldsymbol{\mu})$, é uma matriz diagonal positiva, conhecida como matriz de aprendizado e é usada para reduzir o tempo do transitório do sistema [12], e o vetor hsgn foi definido em (3.31).

O sistema gradiente (5.8)–(5.9) pode ser interpretado como um sistema de controle quando representado usando uma notação adequada:

$$\boldsymbol{\Delta} := \begin{pmatrix} \mathbf{I}_{n+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & c\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \rho\mathbf{I}_m \end{pmatrix}, \; \boldsymbol{\Gamma}^T := \begin{pmatrix} \mathbf{I}_{n+1} & \mathbf{0} & \mathbf{Z}^T \\ \mathbf{0} & \mathbf{I}_m & \mathbf{I}_m \end{pmatrix},$$

$$\boldsymbol{\kappa} = [\mathbf{0}_{n+1}^T, \mathbf{0}_m^T, \mathbf{1}_m^T]^T, \; \boldsymbol{\upsilon} := [\mathbf{u}^T, \boldsymbol{\xi}^T]^T, \; \mathbf{f}(\boldsymbol{\upsilon}, \mathbf{r}) := [\boldsymbol{\upsilon}^T, \text{hsgn}^T(\mathbf{r})]^T.$$

Deste modo, o sistema gradiente (5.8)–(5.9) pode ser escrito como:

$$\dot{\boldsymbol{\upsilon}} = -\boldsymbol{\Gamma}^T \boldsymbol{\Delta} \mathbf{f}(\boldsymbol{\Gamma}\boldsymbol{\upsilon} - \boldsymbol{\kappa}). \tag{5.10}$$

O diagrama de blocos do sistema gradiente, reescrito na forma (5.10), é mostrado na figura 5.1. Nesta representação, o sistema de controle apresenta uma realimentação de saída e o controlador é formado por uma parte não-linear e uma linear, com matriz de ganho Δ. O vetor de estados é $\boldsymbol{\upsilon}$. A parte não-linear do controlador é responsável por forçar as trajetórias para o conjunto viável do problema (5.2), enquanto que a parte linear dirige as trajetórias para a solução global do problema deste problema.

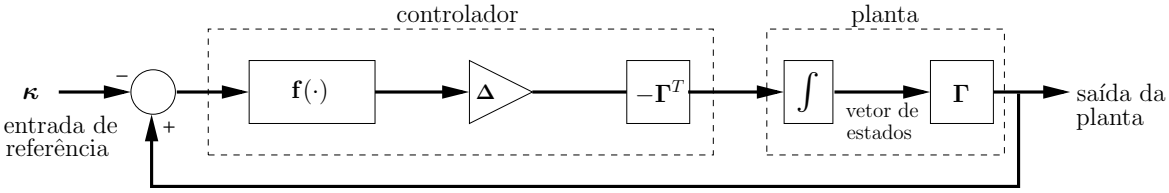


Figura 5.1: Diagrama de blocos que representa o sistema gradiente não-suave (5.8)–(5.9) como um sistema de controle

Análise de convergência

A análise de convergência é conduzida utilizando a representação do sistema (5.8)–(5.9) na forma Persidskii (3.12). Sem perda de generalidade, consideramos a matriz de aprendizado $\mathbf{M}$ em (5.8)–(5.9) como sendo a matriz identidade. A derivada no tempo do vetor de resíduos $\mathbf{r}$ é dada por:

$$\dot{\mathbf{r}} = -\mathbf{Z}\mathbf{u} - c\,\boldsymbol{\xi} - \rho\,(\mathbf{I}_m + \mathbf{Z}\mathbf{Z}^T)\text{hsgn}(\mathbf{r}), \tag{5.11}$$

sendo que $\mathbf{I}_m$ é a matriz identidade de ordem m.

A forma Persidskii (3.12) do sistema (5.8)–(5.9), consiste do sistema original (5.8)–(5.9) aumentado com a derivada no tempo do vetor de resíduos, dada em (5.11). Em notação vetorial temos:

$$\dot{\mathbf{x}} = -\mathbf{S}\mathbf{D}\mathbf{f}(\mathbf{x}), \tag{5.12}$$

no qual $\mathbf{x} = (\mathbf{u}^T, \boldsymbol{\xi}^T, \mathbf{r}^T)^T$, $\mathbf{f}(\mathbf{x}) = (\mathbf{u}^T, \boldsymbol{\xi}^T, \text{hsgn}^T(\mathbf{r}))^T$ e

$$\mathbf{S} = \left[\begin{array}{cc|c} \mathbf{I}_{n+1} & \mathbf{0} & \mathbf{Z}^T \\ \mathbf{0} & \mathbf{I}_m & \mathbf{I}_m \\ \hline \mathbf{Z} & \mathbf{I}_m & \mathbf{I}_m + \mathbf{Z}\mathbf{Z}^T \end{array}\right]; \tag{5.13}$$

$$\mathbf{D} = \left[\begin{array}{cc|c} \mathbf{I}_{n+1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & c\,\mathbf{I}_m & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \rho\mathbf{I}_m \end{array}\right]. \tag{5.14}$$

Note que a matriz $\mathbf{S}$ é claramente: (i) simétrica, (ii) singular, pois seu terceiro bloco-linha é dado pela soma do primeiro bloco-linha pré-multiplicado por $\mathbf{Z}$ com o segundo. De fato, esta matriz é semi-definida positiva, pois operações elementares nos blocos de $\mathbf{S}$ a reduzem para $\text{diag}(\mathbf{I}_{n+1}, \mathbf{I}_m, \mathbf{0})$, mostrando que todos os pivôs são não negativos.

Como o sistema (5.12) possui o segundo membro descontínuo, as soluções são consideradas no sentido de Filippov [38].

Teorema 5.1. *Dadas quaisquer constantes positivas c e ρ, e quaisquer condições iniciais, as trajetórias do sistema* (5.8)-(5.9) *convergem para a solução do problema de programação quadrática* (5.2)*.*

Prova: Como a matriz $\mathbf{S}$ é semi-definida positiva e $\mathbf{D}$ é definida positiva, então o sistema (5.12) satisfaz todas as condições do teorema 3.2. Segue que as trajetórias de (5.12) convergem para o conjunto invariante $\mathcal{A} := \{\mathbf{x} : \mathbf{f}(\mathbf{x}) \in \mathcal{N}(\mathbf{SD})\}$. Conseqüentemente $\dot{\mathbf{x}} = \mathbf{0}$ e $\dot{\mathbf{r}} = \mathbf{0}$. Logo, as trajetórias do sistema gradiente (5.8)–(5.9) convergem para o conjunto solução do problema (5.2). ■

5.2 ν-SVM para discriminação não-linear

Considere novamente o conjunto de treinamento dado em (5.1). Desejamos resolver o problema de classificação não-linear dos dados do espaço de entrada.

Diversas formulações para este problema estão disponíveis na literatura [77, 84], porém optamos pelo ν-SVM [79], que é adequado à formulação do teorema 3.2. O ν-SVM é formulado pelo seguinte problema de otimização:

$$\begin{aligned} &\text{minimizar}_{\mathbf{w},\boldsymbol{\xi},\eta}\ \frac{1}{2}\|\mathbf{w}\|^2 - \nu\eta + \frac{1}{m}\sum_{i=1}^{m}\xi_i \\ &\text{sujeito a} \quad y_i(\mathbf{w}^T\phi(\mathbf{z}_i) + c) \geq \eta - \xi_i, \\ &\qquad\qquad \eta \geq 0,\ \xi_i \geq 0, \end{aligned} \tag{5.15}$$

sendo que ϕ é uma função que confere ao classificador a capacidade de executar a discriminação não-linear dos elementos do conjunto considerado. O parâmetro adicional ν é um limite inferior para o número de vetores de suporte e um limite superior para a fração de erros de margem [85].

O problema dual de (5.15) é dado pelo seguinte problema de programação quadrática com restrições lineares [85]:

$$\text{minimizar}_{\boldsymbol{\alpha}} \quad \frac{1}{2}\,\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} \tag{5.16}$$

$$\text{subjeito a} \quad \mathbf{y}^T\boldsymbol{\alpha} = 0, \tag{5.17}$$

$$\sum_{i=1}^{m}\alpha_i \geq \nu \tag{5.18}$$

$$0 \leq \alpha_i \leq \frac{1}{m} \tag{5.19}$$

sendo que $\boldsymbol{\alpha} \in \mathbb{R}^m$ é o vetor de variáveis duais, e $\mathbf{Q}$ é a matriz cujos elementos são dados por $q_{ij} = y_i y_j \phi(\mathbf{z}_i)^T\phi(\mathbf{z}_j)$.

A função $k(\mathbf{z}_i, \mathbf{z}_j) = \phi(\mathbf{z}_i)^T\phi(\mathbf{z}_j)$, define uma classe de funções conhecidas como *funções de kernel* [84]. A matriz $\mathbf{K}$, cujos componentes são definidos pela função $k(\mathbf{z}_i, \mathbf{z}_j)$, é denominada matriz de kernel. Na representação dual, dada pelo problema (5.16)–(5.19), a função ϕ, não precisa ser conhecida explicitamente. A função de kernel utilizada tem que satisfazer às condições de Mercer, que exigem que a matriz de kernel $\mathbf{K}$ seja simétrica

definida positiva [84]. Conseqüentemente, a matriz $\mathbf{Q}$ no problema (5.16)–(5.19), que é formada a partir da função de kernel $k(\mathbf{z}_i, \mathbf{z}_j)$, também é simétrica definida positiva. As funções de kernel que satisfazem as condições de Mercer são também conhecidas como *kernels de Mercer*.

Agrupando as restrições do problema anterior em notação vetorial, obtemos

$$\text{minimizar}_{\boldsymbol{\alpha}} \quad \frac{1}{2}\,\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} \tag{5.20}$$

$$\text{subjeito a} \quad \mathbf{y}^T\boldsymbol{\alpha} = 0, \tag{5.21}$$

$$\boldsymbol{\alpha} - m^{-1}\mathbf{1}_m \leq \mathbf{0}_m \tag{5.22}$$

$$\mathbf{B}\boldsymbol{\alpha} - \nu\mathbf{h} \geq \mathbf{0}_{m+1}. \tag{5.23}$$

no qual $\mathbf{0}_m, \mathbf{1}_m \in \mathbb{R}^m$, $\mathbf{0}_{m+1} \in \mathbb{R}^{m+1}$ são vetores coluna, e a matriz $\mathbf{B}$ e o vetor $\mathbf{h}$ são definidos como

$$\mathbf{B} := \begin{pmatrix} \mathbf{1}_m^T \\ \mathbf{I}_m \end{pmatrix}; \mathbf{h} := \begin{pmatrix} 1 \\ \mathbf{0}_m \end{pmatrix}.$$

Note que a função objetivo do problema (5.20)–(5.23) é homogeneamente quadrática em $\boldsymbol{\alpha}$ e as restrições do problema de programação quadrática são lineares.

Seja $\mathbf{r} := \mathbf{B}\boldsymbol{\alpha} - \nu\mathbf{h}$, $e = \mathbf{y}^T\boldsymbol{\alpha}$ e $\mathbf{v} := \boldsymbol{\alpha} - m^{-1}\mathbf{1}$. Aplicando o método da função de penalidade exata ao problema (5.20)–(5.23) obtemos:

$$\text{minimizar } E(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{Q}\boldsymbol{\alpha} + \rho|e| - \gamma\sum_{i=1}^{n}\min(0, r_i) + \beta\sum_{i=1}^{n}\max(0, v_i) \tag{5.24}$$

O sistema gradiente associado ao problema de otimização sem restrições (5.24) é dado por:

$$\dot{\boldsymbol{\alpha}} = -\mathbf{M}[\mathbf{Q}\boldsymbol{\alpha} + \rho\,\mathbf{y}\,\text{sgn}(e) + \gamma\mathbf{B}^T\text{hsgn}(\mathbf{r}) + \beta\,\text{uhsgn}(\mathbf{v})], \tag{5.25}$$

sendo que $\mathbf{M}$ é uma matriz de aprendizado, definida em (5.8)–(5.9) e as não-linearidades hsgn, sgn e uhsgn são definidas em (3.31), (3.22) e (4.5), respectivamente.

Como as não-linearidades no sistema gradiente (5.26) são do tipo diagonal, o sistema

gradiente (5.25) escrito elemento a elemento é:

$$\dot{\alpha}_i = -\mu_i[\sum_{j=1}^{m} q_{ij}\alpha_j + \rho\, y_i\, \text{sgn}(\sum_{j=1}^{m} y_j\alpha_j) + \gamma(\text{hsgn}(\sum_{j=1}^{m} \alpha_j - \nu) + \text{hsgn}(\alpha_i)) + \beta\ \text{uhsgn}(\alpha_i - 1/m)], \tag{5.26}$$

sendo que $i = 1, \ldots, m$ e μ_i é o i-ésimo elemento da diagonal de $\mathbf{M}$.

O sistema gradiente (5.25) também pode ser interpretado como um sistema de controle. Definimos os seguintes vetores e matrizes:

$$\boldsymbol{\Gamma} := [\mathbf{I}_m \quad \mathbf{y} \quad \mathbf{B}^T \quad \mathbf{I}_m]^T, \tag{5.27}$$

$$\boldsymbol{\Delta} := \text{diag}(\mathbf{Q}, \rho\mathbf{1}, \gamma\mathbf{I}_m, \beta\mathbf{I}_m), \tag{5.28}$$

$$\boldsymbol{\kappa} := (\mathbf{0}_m^T, 0, \nu\mathbf{h}^T, m^{-1}\mathbf{1}_m^T)^T \tag{5.29}$$

Utilizando esta notação, o sistema gradiente (5.26) pode ser escrito como:

$$\dot{\boldsymbol{\alpha}} = -\boldsymbol{\Gamma}^T\boldsymbol{\Delta}\mathbf{f}(\boldsymbol{\Gamma}\boldsymbol{\alpha} - \boldsymbol{\kappa})$$

em que $\mathbf{f} := (\boldsymbol{\alpha}^T, \text{sgn}, \text{hsgn}^T,\ \text{uhsgn}^T)^T$ é uma função tipo diagonal [50].

Note que a equação anterior está na forma de (5.10) e seu diagrama de blocos também pode ser representado pelo diagrama da figura 5.1, sendo que $\boldsymbol{\Gamma}$, $\boldsymbol{\Delta}$ e $\boldsymbol{\kappa}$ são dados em (5.27), (5.28) e (5.29), respectivamente e o vetor de estados é $\boldsymbol{\alpha}$. Nesta representação, o sistema de controle apresenta uma realimentação de saída e o controlador é composto por uma parte não-linear e uma linear, com matriz de ganho $\boldsymbol{\Delta}$.

Análise de convergência

A análise de convergência é feita de modo análogo à análise apresentada na seção 5.1. Usamos a representação do sistema gradiente (5.25) na forma Persidskii (3.12) e, sem perda de generalidade, consideramos a matriz de aprendizado $\mathbf{M}$ como sendo a matriz identidade. Finalmente, usamos o teorema 3.2 para provar o resultado.

Sejam a função $\mathbf{f}$ e o vetor $\boldsymbol{\theta}$ definidos como segue:

$$\mathbf{f}(\boldsymbol{\theta}) := (\boldsymbol{\alpha}^T, \text{sgn}(e), \text{hsgn}^T(\mathbf{r}),\ \text{uhsgn}^T(\mathbf{v}))^T \tag{5.30}$$

$$\boldsymbol{\theta} := (\boldsymbol{\alpha}, e, \mathbf{r}, \mathbf{v}). \tag{5.31}$$

As variáveis e, $\mathbf{r}$ e $\mathbf{v}$ também estão presentes no sistema gradiente (5.25), dado em função de $\boldsymbol{\alpha}$. Utilizamos a derivada no tempo de cada uma destas variáveis para expandir o sistema (5.25):

$$\dot{e} = \mathbf{y}^T \dot{\boldsymbol{\alpha}} \tag{5.32}$$

$$\dot{\mathbf{r}} = \mathbf{B}\dot{\boldsymbol{\alpha}} \tag{5.33}$$

$$\dot{\mathbf{v}} = \dot{\boldsymbol{\alpha}} \tag{5.34}$$

Consideremos o sistema dinâmico aumentado, formado por (5.25) e (5.32)-(5.34). Utilizando a notação introduzida em (5.30) e (5.31), podemos escrever em notação vetorial:

$$\dot{\boldsymbol{\theta}} = -\mathbf{A}\mathbf{f}(\boldsymbol{\theta}), \tag{5.35}$$

no qual a matriz $\mathbf{A}$ é:

$$\mathbf{A} = \begin{pmatrix} \mathbf{Q} & \rho\mathbf{y} & \gamma\mathbf{B}^T & \beta\mathbf{I} \\ \mathbf{y}^T\mathbf{Q} & \rho\mathbf{y}^T\mathbf{y} & \gamma\mathbf{y}^T\mathbf{B}^T & \beta\mathbf{y}^T \\ \mathbf{B}\mathbf{Q} & \rho\mathbf{B}\mathbf{y} & \gamma\mathbf{B}\mathbf{B}^T & \beta\mathbf{B} \\ \mathbf{Q} & \rho\mathbf{y} & \gamma\mathbf{B}^T & \beta\mathbf{I} \end{pmatrix}.$$

A matriz $\mathbf{A}$ pode ser fatorada na forma $\mathbf{A} = \mathbf{S}\mathbf{D}$, em que:

$$\mathbf{S} := \left(\begin{array}{c|ccc} \mathbf{I} & \mathbf{y} & \mathbf{B}^T & \mathbf{I} \\ \hline \mathbf{y}^T & \mathbf{y}^T\mathbf{y} & \mathbf{y}^T\mathbf{B}^T & \mathbf{y}^T \\ \mathbf{B} & \mathbf{B}\mathbf{y} & \mathbf{B}\mathbf{B}^T & \mathbf{B} \\ \mathbf{I} & \mathbf{y} & \mathbf{B}^T & \mathbf{I} \end{array}\right), \mathbf{D} := \left(\begin{array}{c|ccc} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \rho & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \gamma\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \beta\mathbf{I} \end{array}\right). \tag{5.36}$$

Teorema 5.2. *Se* $\mathbf{Q}$ *é definida positiva então, para quaisquer constantes positivas* ρ*,* γ *e* β*, as trajetórias do sistema gradiente* (5.25) *convergem para a solução do problema de*

programação quadrática (5.20).

Prova: Seja $\mathbf{Q}$ simétrica definida positiva. Então a função objetivo do problema (5.20) é estritamente convexa, bem como a função objetivo do problema (5.24).

Por outro lado, como a matriz $\mathbf{A}$, definida em (5.36) pode ser estrita como o produto de $\mathbf{S}$ e $\mathbf{D}$, definidas em (5.36), e estas matrizes são respectivamente simétrica semi-definida positiva e diagonal por blocos definida positiva, então, pelo teorema 3.2, as trajetórias do sistema (5.25) convergem para o conjunto invariante $\mathcal{A} = \{\boldsymbol{\theta} : \mathbf{f}(\boldsymbol{\theta}) \in \mathcal{N}(\mathbf{B})\}$.

Segue que $\dot{\boldsymbol{\theta}} = \mathbf{0}$. Daí, tem-se que $\dot{\boldsymbol{\alpha}} = \mathbf{0}$, o que implica em $\dot{e} = 0$, $\dot{\mathbf{v}} = \mathbf{0}$ e $\dot{\mathbf{r}} = \mathbf{0}$. Logo, as trajetórias do sistema gradiente convergem para a solução do problema (5.20). ■

A hipótese de que a matriz $\mathbf{Q}$ é definida positiva, usada no teorema 5.2, é satisfeita através do uso kernels de Mercer nas implementações, que são definidos positivos. Além disso, note que a convergência do sistema gradiente não depende da escolha do parâmetro ν. Na prática, a escolha de ν é importante e alguns comentários são feitos no exemplo 5.5.3. Em abordagens convencionais de ν-SVM, métodos para a escolha de ν foram propostos por Chang e Lin [86], que obteve limites para os valores de ν, e Steinwart [87] propôs um método para obter o melhor valor de ν.

5.3 Análise de complexidade

Do mesmo modo que no capítulo 4, utilizamos as definições de complexidade dadas em [74], reproduzidas na seção 4.4, página 85, que possibilitam a comparação dos nossos sistemas com outras redes neurais para a resolução de problemas de programação quadrática quanto à complexidade.

Cada equação do sistema gradiente (5.8)–(5.9) executa $m + n + 2$ multiplicações e 2(m+n+1) adições a cada iteração e, conseqüentemente, a complexidade do modelo é $(3(m+n+1)+1)(m+n+1)$. Conseqüentemente a complexidade assintótica do modelo é igual a $\Theta((m+n)^2)$. Por outro lado, cada equação do circuito modelado por (5.26) executa $m^2 + 3m + 2$ multiplicações e $4m$ adições por iteração; a complexidade do modelo, neste caso, é $m^2 + 6m + 2$, e a complexidade assintótica do modelo é igual a $\Theta(m^2)$.

A rede em Leung et al. [88], que executa $4(m+n+1)+m$ adições e $(m+n+1)(5m+n+7)+m$ multiplicações por iteração, a complexidade do modelo é $(m+n+1)(5m+n+$

$11) + 2m$, resultando em complexidade assintótica do modelo igual a $\Theta((m+n)^2)$. Para o modelo ν-SVM, esta rede executa $6m^2 + 3m$ multiplicações e $4m + 3$ adições, resultando em complexidade do modelo igual a $6m^2 + 7m + 3$, e a complexidade assintótica do modelo é $\Theta(m^2)$.

O sistemas gradientes (5.8)–(5.9) e (5.26) apresentam a mesmas complexidades assintóticas que a rede primal-dual proposta em Leung et al. [88], que resolve problemas de programação quadrática e linear. Entretanto, a complexidade do modelo do último é mais elevada que a complexidade do modelo dos sistemas gradientes (5.8)–(5.9) e (5.26).

Tabela 5.1: Valores da complexidade dos sistemas gradientes (5.8)–(5.9) e (5.26) e outros modelos de redes neurais para a resolução de problemas de otimização quadrática e treinamento de SVMs.

Modelo	Complexidade do modelo	Complexidade assintótica do modelo	Separação
Sistema (5.8)–(5.9)	$3(m+n+1)^2 + (m+n+1)$	$\Theta((m+n)^2)$	Linear
Sistema (5.26)	$m^2 + 7m + 2$	$\Theta(m^2)$	Não linear
Leung et al. [88]	$(m+n+1)(5m+n+11) + 2m$	$\Theta((m+n)^2)$	Linear
	$6m^2 + 7m + 3$	$\Theta((m)^2)$	Não linear
Xia e Wang [89]	$4m + 2$	$\Theta(m)$	Linear
	$2m^2 + 5m$	$\Theta(m^2)$	Não linear

A dimensão do sistema gradiente (5.26) para o treinamento de ν-SVM é m, enquanto que a rede neural primal-dual, proposta por Leung et al. [88], também baseada em um sistema gradiente, possui dimensão igual a $2m + n + 1$. O sistema gradiente (5.8)–(5.9) apresenta dimensão $m + n + 1$, e a rede proposta em Leung et al. [88], quando aplicada ao mesmo problema, apresenta dimensão igual a $2m + n + 1$. Devido à menor dimensão dos sistemas gradientes (5.8)–(5.9) e (5.26) em relação à rede de Leung et al. [88], as implementações através de hardware são mais simples e baratas, pois a quantidade de componentes eletrônicos e cabeamento para conectar estes componentes é menor. No caso de uma implementação em computadores digitais, a menor dimensão dos sistemas (5.8)–(5.9) e (5.26) é também uma vantagem, pois o número de variáveis envolvidas nos cálculos é menor, exigindo uma menor quantidade de memória para armazená-las.

Como outra comparação, a rede proposta por Xia e Wang [89], que apresenta menor complexidade que as redes propostas em Tan et al. [82] e Anguita e Boni [80], executa $2m + 1$ multiplicações e adições quando aplicada ao problema de classificação linear, resultando em complexidade do modelo $4m + 2$. O sistema gradiente (5.8)–(5.9) apresenta

maior complexidade, porém como este sistema determina a solução do problema primal, a dimensão do vetor de pesos $\mathbf{w}$ não depende do número de elementos no conjunto de treinamento. No problema de separação não-linear, a complexidade do modelo da rede proposta em Xia e Wang [89] executa $m(m+2)$ produtos e $m(m+3)$ adições, conseqüentemente, a complexidade do modelo é $2m^2 + 5m$, que é maior que a complexidade do modelo do sistema gradiente (5.26) para classificação não-linear quando $m \geq 4$. Por exemplo, em um conjunto de treinamento com 100 elementos, o sistema (5.26) apresenta complexidade do modelo igual a 10702, enquanto que a complexidade do modelo da rede em [89] é 20500. Os resultados de complexidade estão resumidos na tabela 5.1.

5.4 Paralelização dos sistemas gradientes

Os sistemas gradientes (5.8)–(5.9) e (5.26) são adequados para implementação em paralelo utilizando clusters de PCs ou arquiteturas com múltiplos processadores com memória compartilhada. As equações podem ser distribuídas entre os processadores de um computador paralelo, reduzindo o tempo total de processamento.

Os sistemas (5.8)–(5.9) e (5.26) são resolvidos através de integração numérica. Para sistemas de grandes dimensões, a integração em paralelo é necessária para reduzir o tempo de computação. De acordo com Petcu [90], a formas de paralelização de algoritmos de integração numérica de equações diferenciais ordinárias podem ser classificadas como:

(i) Paralelização através do método ou equivalentemente, do espaço – consiste na execução em paralelo das etapas que compõem o algoritmo de integração.

(ii) Paralelização através do sistema ou equivalentemente, do tempo – envolve a paralelização de alguns procedimentos necessários à integração, como a avaliação do segundo membro da equação diferencial.

Neste trabalho, utilizamos a paralelização através do sistema. Esta forma de paralelização é mais adequada aos métodos de integração escolhidos: os métodos de Euler e de Runge-Kutta de segunda ordem. O método de Runge-Kutta de segunda ordem, é um método de dois estágios, sendo que o segundo estágio depende dos resultados produzidos pelo primeiro, e devem ser executados seqüencialmente. No entanto, a avaliação do segundo membro dos

sistemas gradientes (5.8)–(5.9) e (5.26) em cada estágio é feita em paralelo, através da distribuição das equações diferenciais que compõem os sistemas entre as unidades de processamento.

5.4.1 Paralelização do sistema gradiente para treinamento de SVMs para separação linear

A integração numérica do sistema gradiente (5.8)–(5.9) é executada através da paralelização do cálculo do vetor de resíduos $\mathbf{r}$ e da paralelização do produto da transposta da matriz de classes, denotada por $\mathbf{Z}^T$, pela não-linearidade hsgn. Estas são as tarefas que exigem maior tempo de computação, pois se o conjunto de treinamento possuir uma grande quantidade de entradas, então a matriz $\mathbf{Z}$ terá dimensão muito grande, o que justifica a utilização do paralelismo na integração numérica de (5.8)–(5.9).

O sistema gradiente (5.8)–(5.9) consiste de dois sistemas de equações diferenciais acoplados entre si através do vetor de resíduos $\mathbf{r}$. O sistema (5.8) tem dimensão $n+1$, e determina os coeficientes w_i e o bias b do hiperplano separador. O sistema (5.9) tem dimensão m, que é o número de elementos no conjunto de treinamento e determina as variáveis de folga ξ_i do SVM.

5.4.1.1 Particionamento e mapeamento do problema

O particionamento do problema é feito através da divisão do conjunto de treinamento em conjuntos menores, que são distribuídos uniformemente entre p unidades de processamento de um computador. Além disso, as $n+1$ equações do sistema (5.8) e as m equações do sistema (5.9) também são distribuídas entre os p processadores. Deste modo, se $k = 0, 1, \ldots, p-1$ é a identificação de cada unidade de processamento, são criadas as seguintes partições:

$\mathbf{Z}_l^{(k)} \rightarrow$ partição por linhas da matriz de classes no k-ésimo processador

$\mathbf{Z}_{lc}^{(k)} \rightarrow$ partição por linhas e colunas da matriz de classes no k-ésimo processador

$\mathbf{u}^{(k)} \rightarrow$ partição do vetor $\mathbf{w}$ no k-ésimo processador

$\boldsymbol{\xi}^{(k)} \rightarrow$ partição do vetor $\boldsymbol{\xi}$ no k-ésimo processador

A partição por linhas $\mathbf{Z}_l^{(k)}$, consiste das linhas da matriz $\mathbf{Z}$ mapeadas no k-ésimo processador. A partição por linhas e colunas $\mathbf{Z}_{lc}^{(k)}$ no k-ésimo processador, consiste em considerar na partição $\mathbf{Z}_l^{(k)}$ apenas as colunas correspondentes aos índices do vetor $\mathbf{w}^{(k)}$, o que permite reduzir a quantidade de operações necessárias em cada processador para o cálculo em paralelo do segundo membro de (5.8).

A dimensão de cada partição $\mathbf{Z}^{(k)}$ é dada pelo quociente entre o número de elementos do conjunto de treinamento m e o número de processadores usados. Denotando o resto do quociente m/p por $m \bmod p$, e a divisão inteira de m por p, por $m\backslash p$, então a dimensão de cada partição é dada por

$$\begin{cases} m\backslash p, \text{ se } k \neq 0 \\ m\backslash p + m \bmod p, \text{ se } k = 0 \end{cases} \tag{5.37}$$

Procedimento análogo é utilizado para mapear as partições $\mathbf{w}^{(k)}$ e $\boldsymbol{\xi}^{(k)}$ nos p processadores considerados.

O mapeamento de tarefas requer alguns cuidados quando o sistema computacional utilizado possui processadores heterogêneos. Neste caso, a maior carga deve ser atribuída ao processador mais veloz do conjunto utilizado na execução.

5.4.1.2 Execução de tarefas

Utilizando o particionamento definido na subseção anterior, a integração numérica do sistema (5.8)–(5.9) através do método de Runge-Kutta de segunda ordem exige duas avaliações do segundo membro do sistema gradiente a cada iteração, o que exige a execução das tarefas a seguir:

1. Cálculo do vetor de resíduos $\mathbf{r}$: utilizando as partições definidas acima, cada partição $\mathbf{r}^{(k)}$ do vetor de resíduos é calculada em paralelo através da fórmula

$$\mathbf{r}^{(k)} = \mathbf{Z}_l^{(k)}\mathbf{u} + \boldsymbol{\xi}^{(k)} - \mathbf{1}^{(k)}, \tag{5.38}$$

   sendo que $\mathbf{1}^{(k)}$ é o vetor de uns mapeado no k-ésimo processador, cuja dimensão é dada pelo número de linhas da partição $\mathbf{Z}_l^{(k)}$.

2. Determinação do produto da não-linearidade pela transposta da matriz $\mathbf{Z}$: de posse do vetor $\mathbf{r}^{(k)}$, este cálculo é feito através da fórmula $(\mathbf{Z}_{lc}^{(k)})^T \text{hsgn}(\mathbf{r}^{(k)})$.

De posse dos resultados obtidos nos ítens (1) e (2), e utilizando as partições definidas acima, ao k-ésimo processador da máquina paralela, é atribuída a seguinte partição do sistema gradiente (5.8)–(5.9):

$$\dot{\mathbf{u}}^{(k)} = -\mathbf{u}^{(k)} - \rho(\mathbf{Z}_{lc}^{(k)})^T \text{hsgn}(\mathbf{r}^{(k)}) \tag{5.39}$$

$$\dot{\boldsymbol{\xi}}^{(k)} = -c\boldsymbol{\xi}^{(k)} - \rho\text{hsgn}(\mathbf{r}^{(k)}) \tag{5.40}$$

Este procedimento proporciona uma redução considerável na quantidade de operações aritméticas executadas por cada processador em cada iteração do integrador. No que diz respeito à comunicação de resultados parciais, o cálculo de cada partição $\mathbf{r}^{(k)}$ do vetor de resíduos, exige que cada processador envie as partições $\mathbf{w}^{(k)}$ às demais unidades de processamento. Esta é a única operação de sincronização exigida, e a dimensão deste vetor não depende da quantidade de elementos no conjunto de treinamento, mas sim da dimensão dos vetores deste conjunto.

5.4.2 Paralelização do sistema gradiente para treinamento de SVMs para separação não-linear

A integração numérica em paralelo do sistema gradiente (5.25) segue procedimento análogo ao usado na integração em paralelo do sistema (5.8)–(5.9).

Consideremos a notação escalar em (5.26). Na avaliação do segundo membro deste sistema, a operação que exige maior tempo de computação é o produto da matriz $\mathbf{Q}$ pelo vetor $\boldsymbol{\alpha}$. A matriz $\mathbf{Q}$ é determinada a partir da função de kernel, e sua dimensão depende do número de elementos no conjunto de treinamento, que pode ser muito elevado, justificando a criação desta matriz em paralelo. A matriz $\mathbf{Q}$ é simétrica e, em geral, densa.

O produto interno $e = \mathbf{y}^T\boldsymbol{\alpha}$, presente no segundo membro de (5.26) também é executado em paralelo, o que é justificado pela dimensão do vetor $\boldsymbol{\alpha}$, que é elevada para grandes conjuntos de treinamento.

Com o objetivo de criar esta matriz $\mathbf{Q}$ utilizando o menor tempo possível de computação, procedemos do seguinte modo:

- A criação é feita em paralelo: cada processador cria a partição necessária para os cálculos locais;

- Como a matriz $\mathbf{Q}$ é simétrica, apenas o triângulo inferior é computado: cada unidade de processamento computa uma partição do triângulo inferior, e envia aos demais processadores para compor as partições do triângulo superior.

O procedimento descrito acima tem como vantagem a redução do número de operações aritméticas em cada processador para criar a matriz de kernel. Esta redução é conseguida através da paralelização, atribuindo a cada processador uma partição da matriz $\mathbf{Q}$, e explorando a simetria desta matriz.

Consideremos as seguintes partições da matriz $\mathbf{Q}$ e dos vetores $\mathbf{y}$ e $\boldsymbol{\alpha}$ mapeadas no k-ésimo processador:

$\boldsymbol{\alpha}^{(k)} \rightarrow$ partição do vetor $\boldsymbol{\alpha}$ no k-ésimo processador

$\mathbf{y}^{(k)} \rightarrow$ partição do vetor de classes $\mathbf{y}$ no k-ésimo processador

$\mathbf{Q}_c^{(k)} \rightarrow$ partição por colunas da matriz $\mathbf{Q}$ no k-ésimo processador

$\mathbf{Q}_{lc}^{(k)} \rightarrow$ partição por linhas e colunas da matriz $\mathbf{Q}$ no k-ésimo processador

Utilizando a notação definida em (5.37), a dimensão de cada partição por colunas $\mathbf{Q}_c^{(k)}$ da matriz $\mathbf{Q}$ é dada por

$$\begin{cases} m \times (m \backslash p), \text{ se } k \neq 0 \\ m \times (m \backslash p + m \bmod p), \text{ se } k = 0 \end{cases}$$

A partição por linhas e colunas $Q_{lc}^{(k)}$, é formada considerando na partição por colunas $\mathbf{Q}_c^{(k)}$, apenas as linhas cujos índices correspondem aos índices das partições do vetor $\boldsymbol{\alpha}$ mapeados no processador k.

As dimensões das partições acima são determinadas de modo análogo às partições definidas na subseção 5.4.1, dadas por (5.37). A integração numérica do sistema gradiente (5.26), exige o cumprimento das seguintes tarefas:

1. O produto $\mathbf{Q}\boldsymbol{\alpha}$ é feito parcialmente em cada processador através da fórmula $\mathbf{Q}_{lc}^{(k)}\boldsymbol{\alpha}^{(k)}$. Cada processador comunica aos demais o resultado parcial obtido. Em cada iteração, o k-ésimo processador soma o resultado computado por ele com os resultados parciais

enviados pelos demais processadores, isto é,

$$\boldsymbol{\pi} = \sum_{k=0}^{p-1} \mathbf{Q}_{lc}^{(k)} \boldsymbol{\alpha}^{(k)}.$$

2. O produto interno $e = \mathbf{y}^T \boldsymbol{\alpha}$: o k-ésimo, computa o produto interno entre as k-ésimas partições dos vetores $\mathbf{y}$ e $\boldsymbol{\alpha}$, dado por $(\mathbf{y}^{(k)})^T \boldsymbol{\alpha}^{(k)}$, e o resultado parcial obtido é enviado aos demais processadores. Cada processador calcula a soma do resultado parcial computado por ele com os resultados parciais recebidos dos demais processadores, isto é,

$$e = \sum_{k=0}^{p-1} (\mathbf{y}^{(k)})^T \boldsymbol{\alpha}^{(k)}.$$

Utilizando o particionamento e os produtos acima e, sem perda de generalidade, considerando a matriz $\mathbf{M}$ como sendo a matriz identidade, a partição do sistema gradiente (5.26) em cada processador é dada por:

$$\dot{\boldsymbol{\alpha}}^{(k)} = -\boldsymbol{\pi} - \rho \mathbf{y}^{(k)} \text{sgn}(e) - \gamma[\text{hsgn}(\mathbf{1}^T \boldsymbol{\alpha}^{(k)} - \mathbf{1}\nu) + \text{hsgn}(\boldsymbol{\alpha}^{(k)})] - \beta \text{ uhsgn}(\boldsymbol{\alpha}^{(k)} - \mathbf{1}/m). \tag{5.41}$$

A técnica de paralelização descrita acima proporciona um menor número de operações aritméticas a cada avaliação do segundo membro do sistema gradiente (5.26) em cada processador, o que reduz o tempo total de computação. A comunicação necessária para fins de sincronização dos processos paralelos, envolve a transmissão dos produtos parciais necessários para calcular o produto $\boldsymbol{\pi}$ e os produtos internos parciais necessários para calcular o produto interno e.

5.5 Exemplos numéricos

Para ilustrar a eficiência dos sistemas gradientes propostos, utilizamos os sistemas (5.8)–(5.9) e (5.26) com oito bases de dados do UCI Machine Learning Repository [91]. As bases de dados escolhidas são Iris Plants, Adult database, Wisconsin Breast Cancer Database (WBC), Johns Hopkins Ionosphere Database (Ion), Bupa Liver Disorders (Bupa), Pima Indians Diabetes (Pima), Sonar Mines vs. Rocks (Sonar) e Australian Credit Approval (Aus).

Cada atributo nas bases de dados foi reescalado no intervalo $[-1, 1]$ e elementos com o

conjunto de atributos incompletos foram removidos [92]. Cada base foi dividida em dois conjuntos: conjunto de treinamento e conjunto de teste. O conjunto de teste corresponde a 2/3 dos elementos de cada base, escolhidos aleatoriamente, e os elementos restantes são utilizados como conjunto de teste. As descrições das bases são apresentadas na tabela 5.2.

Tabela 5.2: Descrição das bases de dados do UCI Machine Learning Repository selecionadas para avaliação dos algoritmos propostos.

	Iris	Adult	BCW	Ion	Bupa	Pima	Sonar	Aus
Num. total elem.	100	48842	683	351	345	768	208	690
Num. Elem. classe 1	50	37155	444	126	145	500	111	383
Num. Elem. classe 2	50	11687	239	225	200	268	97	307
Atributos	5	14	10	34	6	8	8	14
Categóricos	0	8	0	0	0	0	0	8
Numéricos	1	6	10	34	6	8	8	6

Para melhor ilustrar a teoria apresentada neste capítulo, é dada atenção especial à base de dados Iris Plants, que consiste de três classes. A primeira classe é linearmente separável das demais, mas a segunda e a terceira não são linearmente separáveis. Cada classe possui 50 elementos com quatro atributos cada. Consideramos nesta análise as duas classes que não são linearmente separáveis e, com o objetivo de plotar as superfícies separadoras obtidas, apenas dois atributos (comprimento e largura da pétala) foram utilizados.

5.5.1 Detalhes das implementações numéricas

O problema do *chattering*

Devido à presença de termos descontínuos nos segundos membros dos sistemas gradientes (5.8)–(5.9) e (5.26), *chattering* numérico ocorre quando as soluções atingem a fronteira dos conjuntos viáveis dos problemas de otimização correspondentes, evitando que ocorra um movimento deslizante ideal nestas superfícies [47].

O chattering é particularmente crítico para o treinamento de ν-SVMs utilizando o sistema gradiente (5.26), pois a restrição (5.19) é difícil de ser satisfeita. Quando o conjunto de treinamento é grande, o comprimento do i-ésimo intervalo determinado por esta restrição é muito pequeno e, durante o treinamento utilizando o sistema (5.26), a presença de chattering impede que a solução correta seja obtida.

Este problema foi resolvido com sucesso em [25], através de uma mudança na escala do

problema (5.16). A função objetivo e todas as restrições deste problema foram multiplicadas por m, substituindo a variável $\boldsymbol{\alpha}$ pela variável $\bar{\boldsymbol{\alpha}} = m\boldsymbol{\alpha}$. Nesta nova escala, a restrição (5.19) torna-se

$$\mathbf{0} \leq \bar{\boldsymbol{\alpha}} \leq \mathbf{1},$$

que não depende do valor de m. Deste modo, os efeitos do chattering são minimizados e a solução correta é obtida facilmente dividindo a solução encontrada pelo sistema gradiente por m.

O problema do chattering é resolvido através da aproximação dos termos descontínuos por funções contínuas, utilizando a técnica da camada limite, descrita no capítulo 2. A não-linearidade sgn é substituída por

$$\text{ssgn}(a) = \begin{cases} a/\varepsilon, \text{ se } |a| \leq \varepsilon, \\ \text{sgn}(a), \text{ caso contrário} \end{cases}$$

sendo que ε é a espessura da camada limite. Note que a função ssgn é linear dentro da camada limite. Como $\text{hsgn}(a) = 0.5(\text{sgn}(a) - 1)$ e $\text{uhsgn}(a) = 0.5(\text{sgn}(a) + 1)$, a aplicação da técnica da camada limite, nestes casos, é trivial.

Em todos os experimentos deste capítulo, utilizamos uma camada limite com espessura $\varepsilon = 5 \times 10^{-2}$.

Método de integração numérica

A simulações foram feitas através de integração numérica dos sistemas gradientes (5.8)–(5.9) e (5.25), utilizando um método de Runge-Kutta de segunda ordem, com os coeficientes determinados por Cash e Karp [65], descrito no algoritmo 3.3.

O controle do passo de integração é feito através do algoritmo baseado em um controlador PID, descrito em [66]. Os parâmetros do controlador PID foram ajustados como sugerido em [66]:

$$\begin{aligned} P &= 0.075; \\ I &= 0.175; \\ D &= 0.01. \end{aligned} \tag{5.42}$$

5.5.2 SVM para separação linear

De acordo com Schölkopf e Smola [85], o parâmetro c determina o *trade off* entre minimizar o erro de treinamento e maximizar a margem, e não existe maneira de conhecer c a priori. De acordo com Cristianini e Shawe-Taylor [84], escolher c equivale a fixar um valor para $\mathbf{w}$ e minimizar $\sum_i \xi_i^2$. Neste caso, escolhemos c com o objetivo de minimizar o erro de treinamento. Este objetivo pode ser atingido escolhendo um valor elevado para o parâmetro c, que aumenta a influência do termo $\sum_i \xi_i^2$ na função objetivo do problema (5.4).

Com a base de dados Iris Plants, os parâmetros usados são: $c = 5$, $\rho = 10$ e uma matriz de aprendizado $\mathbf{M} = 10\mathbf{I}$, que reduz o tempo do transitório do sistema. Esta escolha de parâmetros garante convergência rápida para a solução do problema (5.2). As trajetórias do sistema na fase de treinamento são exibidas na figura 5.2. Os erros nos conjuntos de treinamento e teste são reportados na tabela 5.3.

O pacote SVM Light [93], que é baseado na estratégia de decomposição descrita por Osuna et al. [94], também foi utilizado neste exemplo, obtendo 5.88 % de erro no conjunto de teste, o que corresponde a duas classificações incorretas, e 6.06 % no conjunto de treinamento, o que corresponde a quatro classificações incorretas. O sistema gradiente (5.8)–(5.9) obteve o mesmo desempenho. Na figura 5.3, é mostrado o hiperplano obtido pelo sistema (5.8)–(5.9) utilizando o conjunto de teste. O desempenho de classificação deste sistema é equivalente ao do pacote SVM Light, com a vantagem adicional que o sistema (5.8)–(5.9) é adequado para implementação através de circuitos.

Este desempenho é também similar ao desempenho de classificação da rede proposta por Anguita et al. [81], que obtém um erro de classificação de 6.25 % na separação das classes dois e três do Iris Plants, em oposição ao sistema (5.8)–(5.9) que obteve 6.06% de erro de treinamento, com a vantagem computacional que, como o sistema proposto (5.8)–(5.9) resolve o problema primal (5.2), a dimensão do vetor de pesos obtido não depende da quantidade de elementos no conjunto de treinamento.

Testamos o sistema proposto (5.8)–(5.9) em outras sete bases de dados do repositório da UCI e comparamos os desempenhos de classificação obtidos com o desempenho obtido pelo SVM Light. Os resultados destes experimentos são apresentados na tabela 5.3. O sistema (5.8)–(5.9) apresenta erros de classificação nos conjuntos de teste menores ou iguais aos obtidos pelo SVM Light.

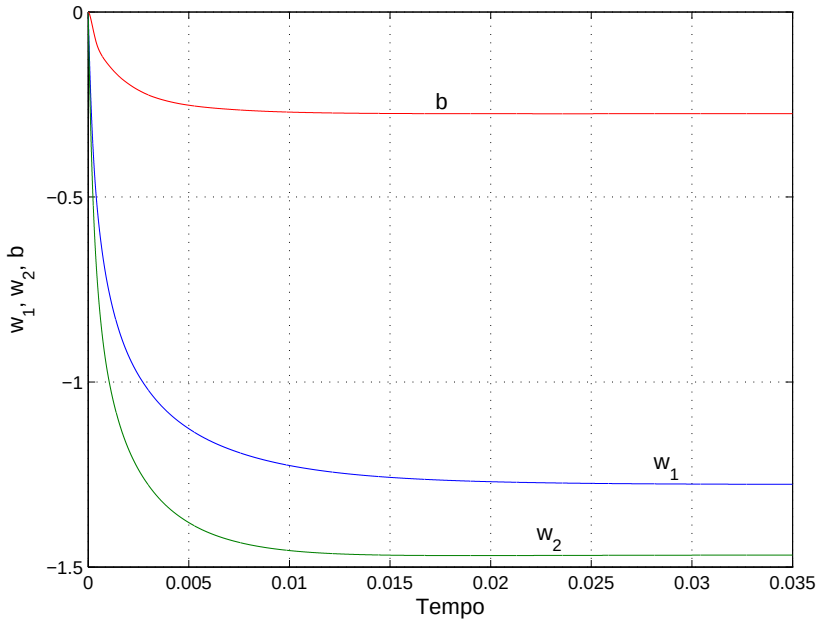


Figura 5.2: Trajetórias w_i, determinadas pelo sistema gradiente (5.8)–(5.9), para as duas classes não-linearmente separáveis da base de dados Iris.

Uma vantagem adicional da abordagem proposta é a escalabilidade; o sistema gradiente (5.8)–(5.9) pode ser facilmente implementado em computadores paralelos. A paralelização deste sistema foi utilizada em um cluster de PCs, equipado com 32 processadores Pentium III, distribuídos em 16 nós duais, com 512 MB de memória cada.

O algoritmo paralelo foi utilizado com a base de dados Adult; como o teorema 5.1 vale para quaisquer c e ρ positivos, mantemos $c = 5$ e escolhemos $\rho = 0.001$, que garante que a norma do vetor gradiente no segundo membro de (5.8)–(5.9) é pequena. As equações do sistema (5.8)–(5.9) foram distribuídas entre 2, 4 e 8 processadores do cluster de PCs. Os erros nos conjuntos de teste e treinamento, obtidos pela paralelização do sistema gradiente (5.8)–(5.9), são significativamente menores que os obtidos pelo SVM Light. Os resultados deste experimento são exibidos na tabela 5.3.

No que diz respeito aos tempos de treinamento utilizando 1, 2, 4 e 8 processadores, os tempos de processamento do sistema gradiente (5.8)–(5.9) são significativamente menores que os do SVM Light em todos os experimentos utilizando a base Adult. Utilizando apenas 1 processador, o tempo de treinamento é 33m26s, usando 2 processadores, o tempo de treinamento é reduzido para 30m28s, e para 15m33s e 8m29s, utilizando 4 e 8 processadores, respectivamente. Por outro lado, o tempo de treinamento do SVM Light, utilizando 1 pro-

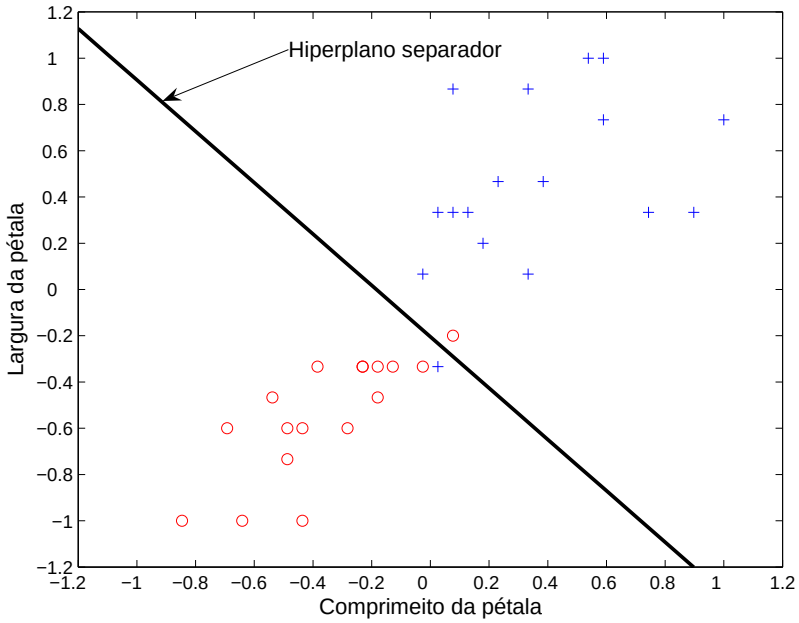


Figura 5.3: Conjunto de teste do Iris Plants e o hiperplano separador, mostrando a separação de classes, com dois erros de classificação, obtido pelo sistema gradiente (5.8)–(5.9)

Tabela 5.3: Medidas de desempenho do SVM para separação linear de duas classes, obtidas pelo sistema gradiente (5.8)–(5.9) e o pacote SVM Light, utilizando 8 bases de dados do repositório da UCI.

	Iris	Adult	BCW	Ion	Bupa	Pima	Sonar	Aus	Método
Margem	0.71	1.75	0.69	0.19	0.68	0.82	0.52	0.69	Sist grad
$(2/\|\mathbf{w}\|_2)$	0.43	0.167	0.67	0.17	0.68	0.44	0.19	1.0	SVM-Light
Erro treinamento %	6.06	17.23	2.86	2.56	30.4	21.3	10.79	12.39	Sist grad
	6.06	75.22	2.42	2.99	21.3	21.5	10.79	13.91	SVM-Light
Erro teste %	2.94	12.01	1.75	9.40	18.26	25.0	15.84	13.91	Sist grad
	5.88	75.21	3.96	9.40	27.83	25.78	20.29	15.65	SVM-Light

cessador da mesma máquina, com o mesmo exemplo, é 1h33m, que é consideravelmente maior que o tempo de treinamento obtido utilizando o sistema gradiente (5.8)–(5.9). Uma vantagem em utilizar o sistema (5.8)–(5.9), quando grandes bases de dados são consideradas, é que o tempo de treinamento pode ser reduzido através da distribuição das equações do sistema gradiente entre os processadores de uma máquina paralela. Os tempos de processamento obtidos neste experimento são exibidos na tabela 5.4, e o decréscimo do tempo de treinamento em função do número de processadores utilizados é mostrado na figura 5.4.

Observação 3. De acordo com Joachims [93], o tempo de treinamento utilizando o SVM Light, para a base de dados "Adult", é menor que o tempo de treinamento do algoritmo SMO (sequential minimal optimization), proposto por Platt [95], que por sua vez, é mais rápido

que o algoritmo padrão de *chunking*.

Tabela 5.4: Tempos de treinamento para a base de dados Adult, usando o sistema gradiente (5.8)–(5.9) em paralelo e o SVM Light.

Método	1 proc	2 procs	4 procs	8 procs
Sistema gradiente	33m26s	30m28s	15m33s	8m29s
SVM Light	1h33m	–	–	–

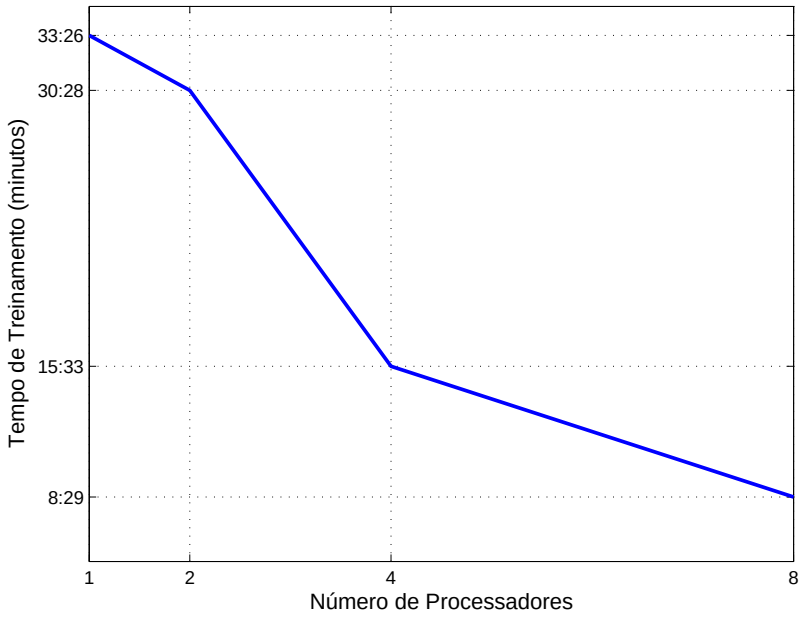


Figura 5.4: Decréscimo do tempo de treinamento com o aumento do número de processadores utilizados quando o sistema gradiente (5.8)–(5.9) é resolvido em paralelo.

Utilizando as medidas de desempenho introduzidas no apêndice A, calculamos a aceleração e a eficiência, obtidos pela paralelização do sistema gradiente, em relação ao tempo de processamento da implementação seqüencial deste sistema. Os cálculos da aceleração e da eficiência são feitos usando as definições A.2 e A.3, respectivamente.

A figura 5.5 mostra a curva de aceleração do sistema gradiente (5.8)–(5.9). O dados mostrados na figura 5.5, indicam que a paralelização proporcionou um ganho de desempenho em relação ao algoritmo seqüencial. No entanto, os valores da aceleração obtidos pela paralelização do sistema (5.8)–(5.9) estão muito abaixo dos valores ideais, que são os da aceleração linear. É desejável que a aceleração obtida pelo algoritmo paralelo esteja o mais próximo possível dos valores ideais.

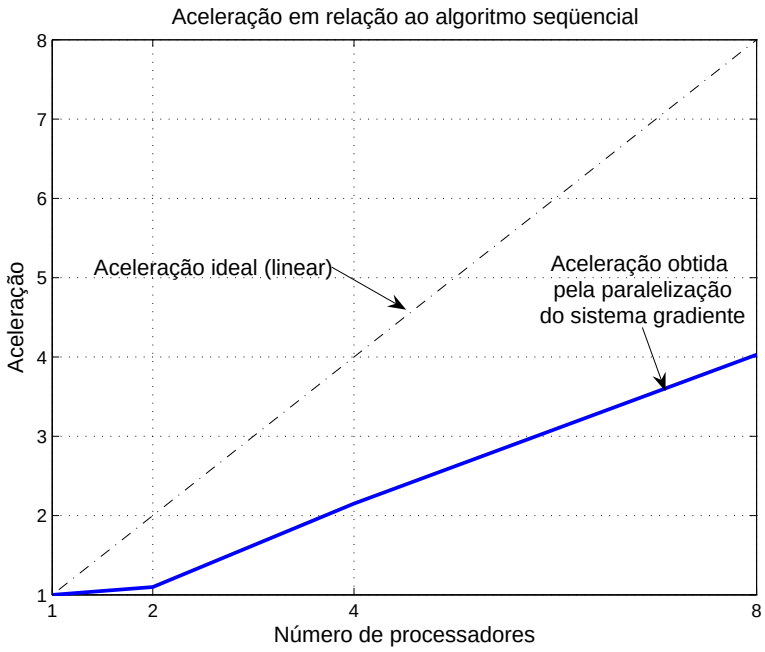


Figura 5.5: Curva de aceleração do sistema gradiente (5.8)–(5.9) para treinamento SVMs para separação linear, mostrada juntamente com a curva de aceleração ideal (linear)

A eficiência fornece uma medida do percentual da fração do tempo total de processamento que é efetivamente utilizada em computação, sem contar o tempo gasto em comunicação e sincronização de processos. A curva de eficiência do sistema gradiente (5.8)–(5.9) é mostrada na figura 5.6, mostrando uma variação entre 55% quando 2 unidades de processamento são usadas, e cerca de 50.5% para 8 unidades de processamento. Estes valores estão muito abaixo do valor ideal, que é 100%. Estes dados, exibidos na figura 5.6, indicam que uma fração considerável do tempo total de processamento é usado em tarefas de sincronização e comunicação de processos.

A partir dos dados de aceleração e eficiência, mostrados na figura 5.5 e 5.6, pode-se afirmar que, apesar da paralelização do sistema gradiente (5.8)–(5.9) ter proporcionado uma considerável melhora no desempenho do sistema, a fração do tempo total de processamento gasto em comunicação e sincronização de processos precisa ser reduzida, o que proporcionará uma elevação nos valores da aceleração e eficiência.

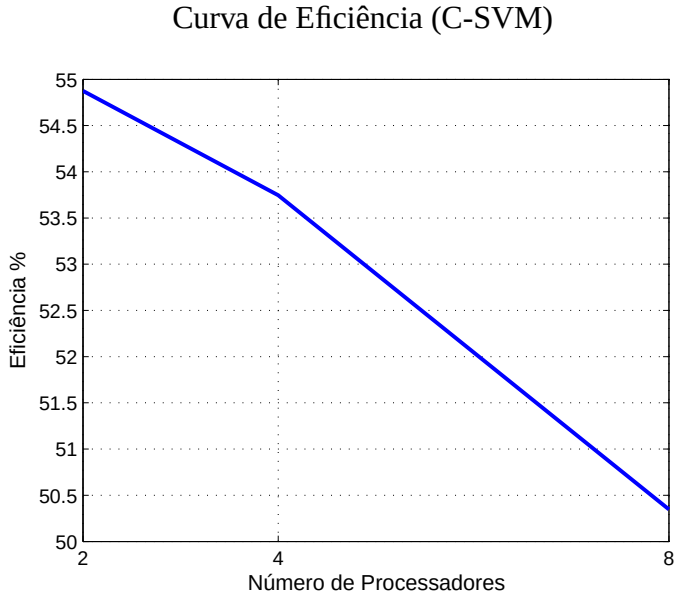


Figura 5.6: Curva de eficiência do sistema gradiente para treinamento de SVMs para separação linear, a eficiência diminui a medida que o número de processadores aumenta.

5.5.3 ν-SVM para separação não-linear

Utilizamos o sistema gradiente (5.26) para o treinamento do ν-SVM com as bases de dados descritas na tabela 5.2. O kernel utilizado é o Gaussiano, que satisfaz as condições de Mercer [84], e é dado por:

$$k(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2/\sigma^2). \tag{5.43}$$

Consideremos novamente a base de dados Iris Plants. Como na seção anterior, 2/3 da base é usada como conjunto de treinamento, que corresponde a 33 elementos de cada uma das classes consideradas, e os elementos restantes são utilizados como conjunto de teste. De acordo com Chang e Lin [86], neste exemplo, o problema (5.20) é viável se e somente se $\nu \leq 1$. Logo é preciso escolher um valor para ν no intervalo $[0, 1]$.

Seja $\nu = 0.1$, e os parâmetros de penalidade $\gamma = 1$, $\rho = 1$ e $\beta = 1$. O parâmetro do kernel Gaussiano é $\sigma = 1$. Uma matriz de aprendizado $\mathbf{M} = 100\mathbf{I}$ também foi usada neste exemplo, o que ajuda a reduzir o tempo do transitório. As trajetórias do sistema gradiente (5.26) são mostradas na figura 5.7 e as curvas de nível da superfície de decisão são exibidas na figura 5.8, utilizando o conjunto de teste, mostrando duas classificações incorretas. Os

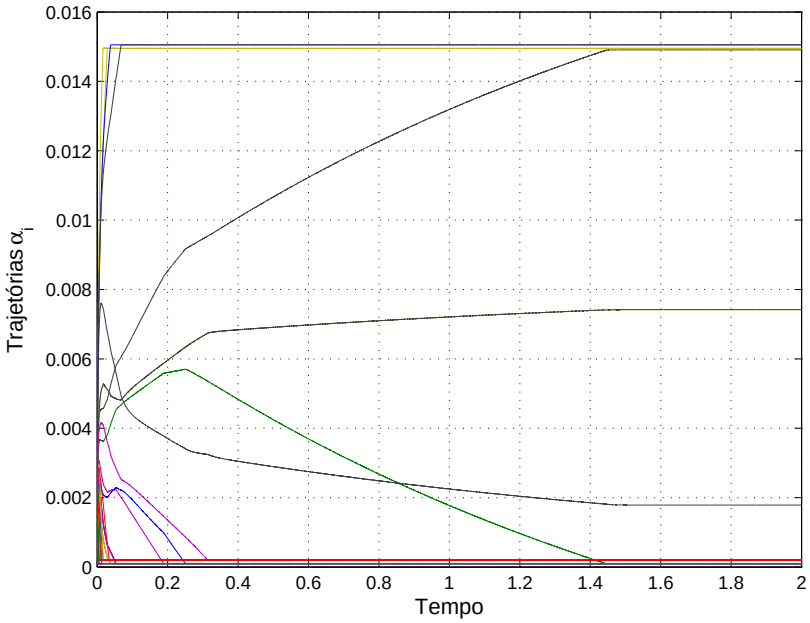


Figura 5.7: Trajetórias α_i, calculadas usando o sistema gradiente (5.26), com $\nu = 0.1$, $\rho = \beta = \gamma = 1$ e o kernel Gaussiano com $\sigma = 1$, para as duas classes não-linearmente separáveis da base de dados Iris Plants, mostrando a convergência com os parâmetros escolhidos adequadamente.

resultados deste experimento estão resumidos na tabela 5.5.

Também comparamos o desempenho do sistema gradiente (5.26) com o desempenho da rede em [89]. Neste caso, 41 elementos da segunda classe e 38 da terceira classe do Iris Plants foram usados para treinamento, o restante foi usado para teste. Os elementos do Iris usados neste exemplo são os mesmos usados por Xia e Wang [89]. O sistema gradiente (5.26) obteve 28 e 34 vetores de suporte, usando um kernel gaussiano com $\sigma = 0.5$ e $\sigma = 1.0$, respectivamente. Estes resultados são muito próximos dos resultados obtidos por Xia e Wang [89], utilizando um kernel polinomial, obtendo 25 e 35 vetores de suporte, para diferentes valores do parâmetro do kernel, com a vantagem que o sistema (5.26) apresenta menor complexidade de modelo para bases de dados com mais de 4 elementos, como discutido na seção 5.3.

Considerando ainda a base de dados Iris, o treinamento também foi feito utilizando o sistema gradiente (5.26) com o kernel polinomial dado por

$$k(\mathbf{z}_i, \mathbf{z}_j) = (\mathbf{z}_i^T \mathbf{z}_j + 1)^p, \tag{5.44}$$

em que p é um parâmetro. Para $p = 2$, os erros de treinamento e de teste são respectivamente

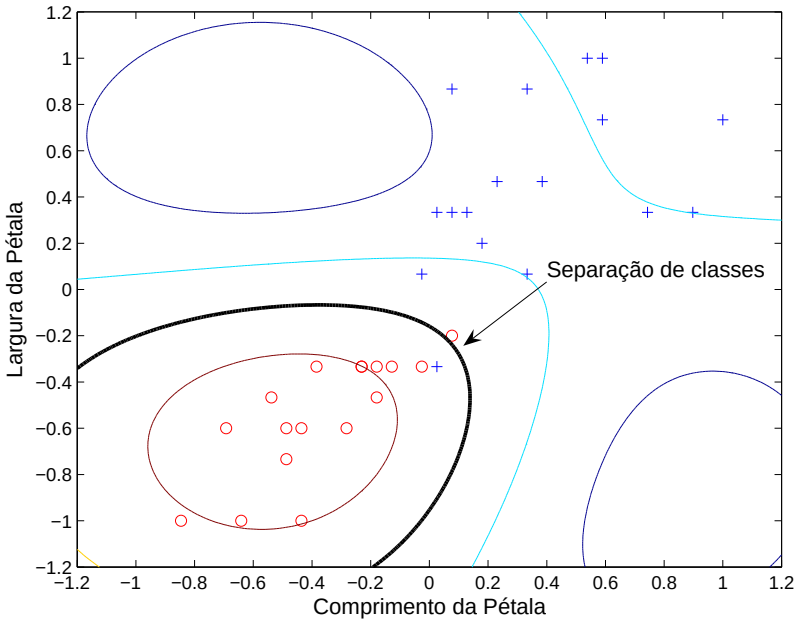


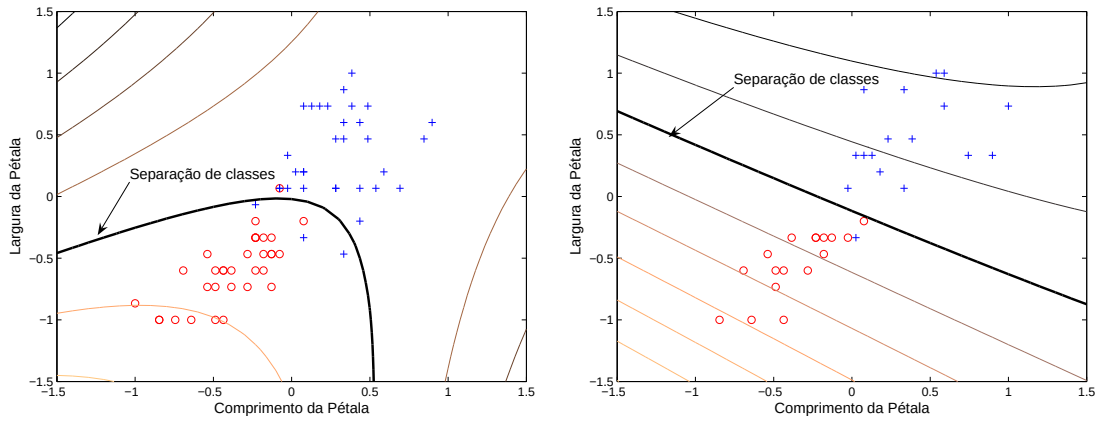
Figura 5.8: Conjunto de teste da base de dados Iris Plants e as curvas de nível da superfície de decisão obtidas pelo sistema gradiente (5.26), mostrando a separação das classes com uma classificação incorreta em cada classe.

6.06% e 0.34%. Estes valores correspondem a quatro classificações incorretas no conjunto de treinamento e apenas uma no conjunto de teste. As curvas de nível da superfície separadora, para os conjuntos de treinamento e de teste, são mostradas na figura 5.9.

Utilizando o kernel polinomial dado em (5.44), foram obtidos 4 vetores de suporte limites e 12 vetores de suporte. O número de vetores de suporte obtidos utilizando o kernel polinomial é maior que o obtido utilizando o kernel gaussiano. No entanto, o erro de teste utilizando o kernel polinomial é consideravelmente menor que o erro obtido utilizando o kernel gaussiano.

Utilizamos o sistema gradiente (5.26) com outras sete bases de dados do repositório da UCI. O sistema (5.26) apresentou desempenho de classificação igual ou superior ao obtido pelo pacote LIBSVM [96], que é baseado em um algoritmo tipo SMO que utiliza informação de segunda ordem para selecionar o *working set* [97], com a vantagem adicional que o sistema gradiente (5.26) também é adequado para implementação através de circuitos.

Do mesmo modo que o sistema (5.8)–(5.9), uma vantagem adicional do sistema (5.26) é a escalabilidade. Este sistema foi implementado em paralelo, utilizando uma máquina SGI Altix 350 com arquitetura de memória compartilhada, equipada com 12 CPUs Itanium, com palavra de 64 bits.



(a) (b)

Figura 5.9: Curvas de nível da superfície separadora para a base Iris, utilizando kernel polinomial – (a) No conjunto de treinamento, mostrando quatro classificações incorretas, o que corresponde a um erro de treinamento de 6.06%; (b) No conjunto de teste, mostrando apenas uma classificação incorreta, correspondendo a um erro de teste de 0.34%

A paralelização do sistema gradiente (5.26) foi utilizada com a base de dados Adult; as equações do sistema gradiente (5.26) foram distribuídas entre 1, 2, 4 e 8 processadores da máquina paralela. Utilizando 1 processador, o tempo de treinamento foi 7h38m, quando 2 processadores são usados, o tempo de treinamento necessário para obter o mesmo resultado é reduzido para 1h59m, com 4 processadores obtemos a mesma solução em 1h14m e com 8 processadores o tempo de treinamento é de apenas 46 minutos.

O tempo de treinamento utilizando o LIBSVM foi de 2h04m, que é maior que o tempo de treinamento do sistema gradiente utilizando dois ou mais processadores. Note que utilizando 1 processador, o tempo de treinamento do sistema (5.26) é consideravelmente maior que o tempo de treinamento utilizando o pacote LIBSVM. Entretanto, quando o sistema gradiente é distribuído entre dois ou mais processadores do computador paralelo, os tempos de processamento do sistema (5.26) são menores que os do LIBSVM, o que é uma vantagem importante quando grandes bases de dados são consideradas. Estes resultados são exibidos na tabela 5.7 e a curva mostrando o decréscimo do tempo de treinamento em função do número de processadores é exibida na figura 5.10.

Os dados discutidos no parágrafo anterior podem ser melhorados através da utilização de outros métodos de integração. Quando o sistema gradiente é resolvido através do método de Euler com controle de passo pela fórmula de Barzilai-Borwein (Euler-BB), descrito no algoritmo 3.2, na página 63, os tempos de processamento são consideravelmente menores.

Tabela 5.5: Dados estatísticos do treinamento do ν-SVM utilizando o sistema gradiente (5.26), resolvido por um método de Runge-Kutta de segunda ordem, e pelo pacote LIBSVM para a separação não-linear de duas classes do Iris Plants, utilizando $\nu = 0.1$, $\rho = \gamma = \beta = 1$ e um kernel Gaussiano com $\sigma = 1$

	Iris	Adult	BCW	Ion	Bupa	Pima	Sonar	Aus	Método
Número de vetores	7	30145	159	69	143	252	105	278	Sist grad
de suporte[1]	10	9679	155	74	121	256	106	278	LIBSVM
Número de vetores	4	0	16	6	0	0	0	2	Sist grad
de suporte limites[2]	6	1657	21	6	1	5	0	11	LIBSVM
Erro treinamento %	6.06	21.22	0.22	0	15.22	1.95	0	1.09	Sist grad
	6.06	7.66	0.88	0	13.48	0.39	0	1.74	LIBSVM
Erro teste %	5.88	21.55	3.95	4.27	28.70	33.20	14.49	20.00	Sist grad
	5.88	22.38	4.39	6.84	39.11	32.42	13.05	20.44	LIBSVM

[1] *Support vectors* na literatura em inglês
[2] *Bound support vectors* na literatura em inglês

Tabela 5.6: Dados estatísticos do treinamento do ν-SVM, utilizando a base de dados "Adult", obtidos através do sistema gradiente (5.26), resolvido através do método de Euler-BB, utilizando $\nu = 0.1$, $\rho = \gamma = \beta = 1$, e o kernel Gaussiano com $\sigma = 0.1$

Erro treinamento	Erro teste	Num. vetores suporte	Num. vetores suporte limites
14.11%	23.54%	3299	0

Utilizando 1 processador, o tempo de treinamento é 1h05m, enquanto que utilizando 8 processadores, o tempo de processamento é reduzido para apenas 10 minutos. Observe que estes tempos de processamento são significativamente menores que os obtidos pelo LIBSVM, mesmo sem utilizar a paralelização do sistema gradiente. Estes dados estão resumidos na tabela 5.7.

O desempenho de classificação obtido pelo sistema gradiente resolvido através do método de Euler-BB, é resumido na tabela 5.6. O erro de treinamento e o número de vetores de suporte obtidos são menores que os obtidos utilizando o método de Runge-Kutta. O número de vetores de suporte obtido também é menor que o obtido pelo LIBSVM. No entanto, o valor do erro de teste obtido quando o sistema gradiente (5.26) é resolvido pelo método de Euler-BB, é maior que os obtidos pelo LIBSVM e quando o sistema gradiente é resolvido pelo método de Runge-Kutta de segunda ordem.

O desempenho da paralelização do sistema gradiente (5.26) é medido através da aceleração e da eficiência, definidos em A.1 e A.3.

A curva de aceleração do sistema gradiente (5.26), resolvido pelo método de Runge-Kutta de segunda ordem, em relação ao tempo de processamento obtido pelo pacote LIB-

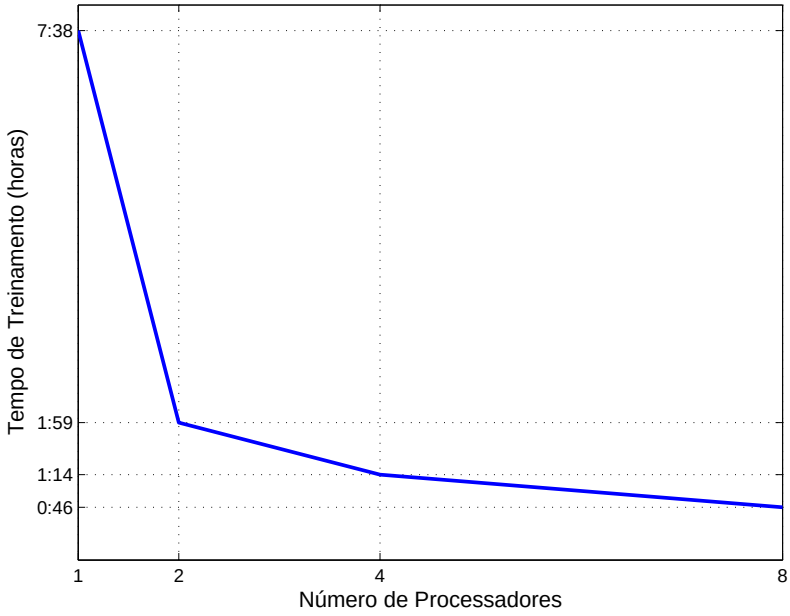


Figura 5.10: Curva mostrando o decréscimo do tempo de treinamento do ν-SVM em função do número de processadores, quando o sistema gradiente (5.26) é resolvido em paralelo.

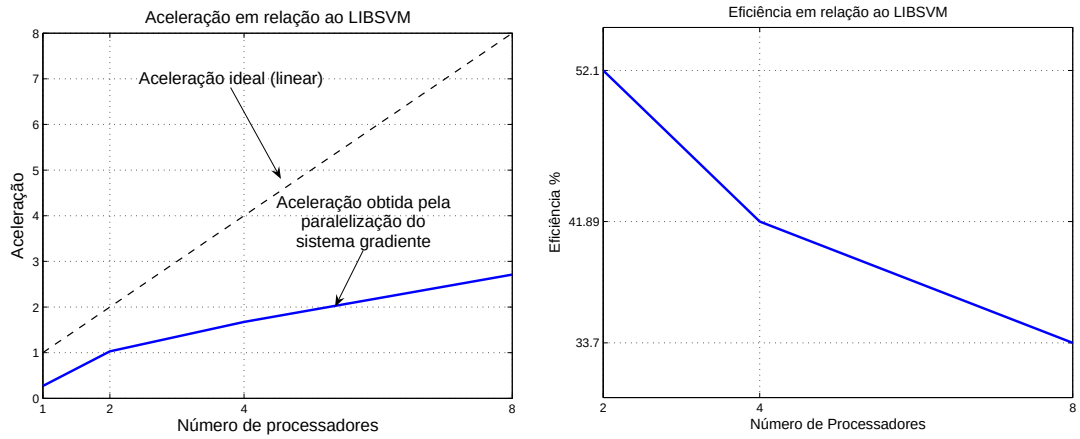
Tabela 5.7: Tempos de processamento do sistema gradiente (5.25), resolvido em paralelo pelos métodos de Runge-Kutta de segunda ordem e de Euler, com controle de passo por PID e pela fórmula de Barzilai-Borwein (Euler-BB), respectivamente, e o tempo de processamento obtido pelo LIBSVM, ambos utilizando um kernel Gaussiano com $\sigma = 1.0$. A base "Adult" foi usada, com $\nu = 0.1$

Modelo	Método integração	1 proc	2 procs	4 procs	8 procs
Sistema gradiente	Runge-Kutta	7h38m	1h59m	1h14m	0h46m
	Euler-BB	1h24m	0h47m	0h20m	0h10m
LIBSVM	–	2h04m	–	–	–

SVM, é mostrada na figura 5.11-(a). Observe que, quando 1 processador é utilizado, o valor da aceleração é menor que 0.5, pois o tempo de processamento da implementação serial do sistema gradiente (5.26), mostrado na tabela 5.7, é mais elevado do que o tempo de processamento obtido pelo LIBSVM. Quando 2, 4 e 8 processadores são usados, os valores da aceleração obtidos mostram que a paralelização proporcionou considerável melhora no desempenho, porém estes valores estão muito abaixo dos valores ideais, que são os da aceleração linear.

A curva de eficiência do sistema gradiente (5.26) em relação ao LIBSVM é mostrada na figura 5.11-(b). Observe que o valor da eficiência, quando 2 processadores são utilizados, é de 52%, e para 8 processadores é de 37%, o que indica que uma fração considerável do

tempo total de processamento, é despendida em tarefas de sincronização e comunicação de processos.



(a) (b)

Figura 5.11: Curvas de desempenho do sistema gradiente (5.26) para treinamento de ν-SVMs, resolvido em paralelo através do método de Runge-Kutta de segunda ordem, em relação ao LIBSVM - (a) Curva de aceleração obtida pela paralelização do sistema gradiente (5.26), e a aceleração ideal (linear); (b) Curva de eficiência

Os dados de aceleração e eficiência, para este exemplo, mostram que, apesar da paralelização ter proporcionado uma melhora significativa no desempenho do sistema, o tempo despendido em tarefas de sincronização e comunicação de processos precisa ser reduzido, o que proporcionará um aumento nos valores da aceleração e da eficiência obtidos pela paralelização do sistema gradiente (5.26).

No caso da paralelização do sistema gradiente (5.26), utilizando o método de Euler com controle de passo pela fórmula de Barzilai-Borwein, foram obtidos valores de aceleração acima dos valores ideais. Este fenômeno é conhecido como *aceleração superlinear*, e freqüentemente ocorre quando características do hardware utilizado colocam o algoritmo serial em desvantagem em relação ao algoritmo paralelo [98]. Um exemplo é quando o volume de dados de um problema é muito grande para ser armazenado na memória *cache*[1] de um único processador, mas quando os dados são particionados e distribuídos entre vários processadores, cada partição é suficientemente pequena para ser armazenada na memória cache de cada unidade de processamento [98], o que proporciona uma melhora significativa no desempenho do algoritmo paralelo. A curva de aceleração é mostrada na figura 5.12.

[1]Memória de alta velocidade, usada para armazenar cópias dos dados das áreas da memória principal acessadas com mais freqüência

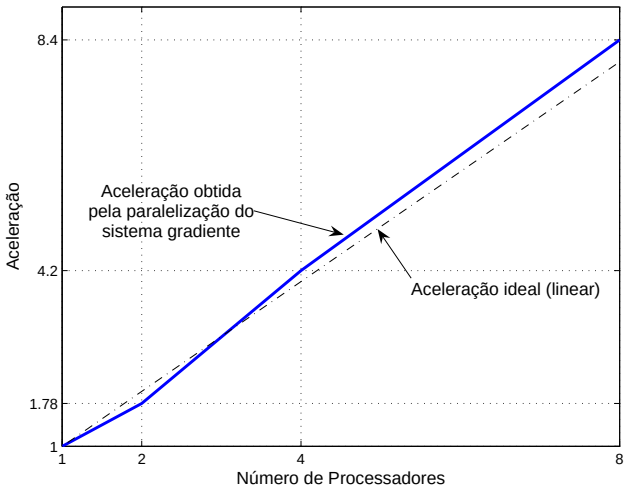


Figura 5.12: Curva de aceleração obtida pelo sistema gradiente (5.26), resolvido utilizando o método de Euler com controle de passo pela fórmula de Barzilai-Borwein (Euler-BB), em relação à formulação serial do sistema gradiente (5.26) resolvido pelo método de Euler-BB, mostrando aceleração superlinear

Outras propostas de algoritmos paralelos para treinamento de SVMs estão disponíveis na literatura. Um algoritmo foi proposto por Zanghirati e Zanni [99] que, de acordo com os próprios autores, é "efetivo no caso de support vector machines Gaussianos", enquanto que o sistema gradiente (5.26) acomoda qualquer kernel de Mercer. O algoritmo proposto por De Leone [100] foi desenvolvido para resolver problemas de programação quadrática com uma única restrição de igualdade e restrições em caixa nas variáveis de decisão, o que o torna inadequado para o treinamento de ν-SVMs e para resolver o problema primal (5.2). O algoritmo proposto por Dong et al. [101] também foi desenvolvido para resolver a mesma classe de problemas de programação quadrática, com uma única restrição de igualdade e restrições em caixa nas variáveis de decisão. Além disso, estes algoritmos, apesar de paralelizáveis, não são adequados para implementação através de circuitos, ao contrário dos sistemas gradientes (5.26) e (5.8)–(5.9). Além disso, os sistemas (5.26) e (5.8)–(5.9) também são adequados para a resolução de problemas gerais de programação quadrática com restrições lineares.

5.6 Least-Squares Support Vector Machines (LS-SVM)

O modelo LS-SVM é uma modificação da formulação original (5.2) do SVM, em que as restrições de desigualdade são substituídas por restrições de igualdade. O LS-SVM é

formulado pelo seguinte problema de programação quadrática [78]

$$\begin{aligned} \text{minimizar } \xi_{ls}(\mathbf{u}, \mathbf{e}, c) &= \frac{1}{2}\mathbf{u}^T\mathbf{u} + b\frac{1}{2}\sum_{i=1}^{n} e_i^2 \\ \text{sujeito a } \mathbf{u}^T\phi(\mathbf{z}_{A_i}) + c &= 1 - e_i,\ y_i = +1 \\ \mathbf{u}^T\phi(\mathbf{z}_{B_i}) + c &= -1 + e_i\ y_i = -1 \end{aligned} \tag{5.45}$$

O problema dual de (5.45) é dado pelo seguinte sistema de equações lineares, também conhecido como sistema linear KKT [78]:

$$\left[\begin{array}{c|c} 0 & \mathbf{y}^T \\ \hline \mathbf{y} & \mathbf{Q} + b^{-1}\,\mathbf{I} \end{array}\right] \left[\begin{array}{c} c \\ \hline \boldsymbol{\alpha} \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline \mathbf{1} \end{array}\right], \tag{5.46}$$

sendo que $\mathbf{Q}$ é uma matriz simétrica, cujos componentes são dados por $q_{ij} = y_i y_j K(\mathbf{z}_i, \mathbf{z}_j)$, sendo K definida por um kernel de Mercer $K(\mathbf{z}, \mathbf{z}_j) = \phi^T(\mathbf{z})\,\phi(\mathbf{z}_j)$. Segue que a função de kernel K é definida positiva e, conseqüentemente, $\mathbf{Q}$ também o é.

No modelo LS-SVM, o problema de obter a superfície de separação das classes equivale a resolver o sistema de equações lineares (5.46), que está na forma (3.54). Observe que, como $b > 0$, $\mathbf{y} \neq \mathbf{0}$ e $\mathbf{Q}$ é definida positiva, então a matriz de coeficientes do sistema KKT (5.46) é não-singular e, conseqüentemente, a solução é única.

O sistema (5.46) pode ser resolvido pelo sistema gradiente (3.54), e o teorema 5.1 garante que as trajetórias do sistema gradiente (3.54) convergem em tempo finito para a solução do sistema de equações lineares (5.46). A seguir, apresentamos um exemplo numérico da utilização do sistema gradiente (3.54) para resolver o sistema KKT (5.46).

Exemplo numérico

Consideremos novamente a base de dados Iris, do UCI Repository of Machine Learning [91]. Utilizamos a primeira e a segunda classes da base Iris, que são linearmente separáveis. A segunda e a terceira classes foram utilizadas para teste.

Neste exemplo, o sistema linear possui 101 equações e 101 incógnitas, e o sistema (3.54) possui 101 equações diferenciais. O parâmetro de regularização usado é $b = 10$. O kernel utilizado neste exemplo é o kernel Gaussiano, dado em (5.43). Como este kernel é de Mercer, então a matriz $\mathbf{Q}$ é definida positiva [78, 84]. A superfície que separa as classes é definida

em função do kernel utilizado, e é dada por:

$$\psi(\mathbf{z}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{z}, \mathbf{z}_i) + c. \tag{5.47}$$

As condições iniciais foram escolhidas arbitrariamente na origem e utilizamos o kernel gaussiano com $\sigma^2 = 1.0$. A matriz de aprendizado do sistema (3.54) é $\mathbf{M} = 10\mathbf{I}$. Algumas trajetórias resultantes são exibidas na figura 5.13, mostrando a convergência para a solução do sistema de equações lineares (5.46).

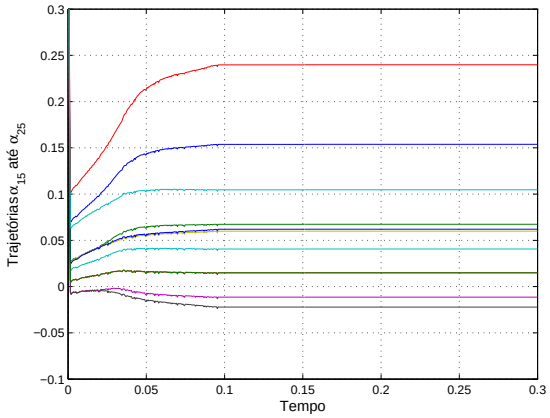
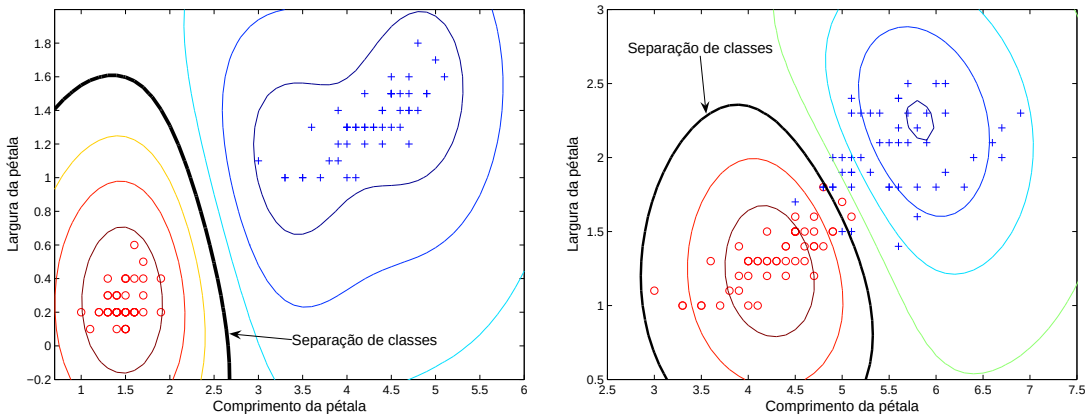


Figura 5.13: Algumas trajetórias, arbitrariamente escolhidas, do sistema gradiente (3.54) para o exemplo Iris, mostrando a convergência em tempo finito para a solução do sistema de equações lineares (5.46)

As curvas de nível da superfície separadora (5.47) para a fase de treinamento do LS-SVM através do sistema gradiente (3.54), são mostradas na figura 5.14-(a). O conjunto de treinamento é separável e a discriminação das classes foi feita sem erros. Na figura 5.14-(b), são exibidas as curvas de nível da superfície separadora (5.47) com a segunda e terceira classes, que não são linearmente separáveis. Neste caso, a separação apresenta 5 classificações incorretas.



(a) (b)

Figura 5.14: LS-SVM (a) Curvas de nível da superfície separadora, obtida pelo sistema gradiente (3.54), mostrando a fase de treinamento do LS-SVM utilizando duas classes linearmente separáveis do Iris. A classificação foi feita sem erros de treinamento; (b) Curvas de nível da superfície separadora, com o conjunto de teste, utilizando duas classes não-separáveis linearmente do Iris, sendo que uma delas não foi apresentada no treinamento. A separação apresenta 5 classificações incorretas.

Síntese do capítulo

Neste capítulo o treinamento de SVMs e LS-SVMs através de sistemas gradientes não-suaves é abordado. Três formulações distintas de support vector machines são consideradas; para cada formulação um sistema gradiente é proposto.

A análise de convergência para os sistemas gradientes para treinamento do C-SVM e ν-SVM é feita utilizando a forma Persidskii destes sistemas e o teorema geral 3.2, para sistemas Persidskii. Estes sistemas são implementados em computadores paralelos, proporcionando uma redução no tempo de treinamento, e tornando estes sistemas competitivos com outros algoritmos para o treinamento do C-SVM e ν-SVM.

Para o treinamento de LS-SVMs, que são modelados através de um sistema de equações lineares, cuja matriz de coeficientes é quadrada e não-singular para um kernel definido positivo, é utilizado o sistema gradiente (3.54).

No próximo capítulo, apresentamos a aplicação de sistemas gradientes para resolver o problema de restauração de imagens.

Capítulo 6

Restauração de imagens utilizando sistemas gradientes

A restauração de imagens é um problema de inversão, que tem como objetivo obter uma estimativa da imagem original, sem degradações, a partir de uma versão degradada por algum tipo de distorção, como embaçamento e ruído.

O problema de restauração de imagens é formulado matematicamente por um problema de otimização, cuja função objetivo mede o nível de degradação a que a imagem foi submetida, e restaurar uma imagem distorcida equivale a minimizar a medida de degradação. Em muitas situações práticas, as degradações em uma imagem podem ser descritas por modelos lineares [102]. Neste caso, minimizar a medida de degradação da imagem equivale a resolver um sistema de equações lineares mal-condicionado e usualmente de dimensões elevadas.

Diversas técnicas estão disponíveis para restaurar imagens degradadas, e os métodos iterativos, que incluem as redes neurais, apresentam diversas vantagens. A idéia ao utilizar redes neurais é tirar proveito da capacidade de processamento paralelo e da possibilidade de implementação através de hardware de baixo custo. Além disso, as redes neurais possuem capacidade de adaptar-se efetivamente à natureza do problema, e não necessitam do conhecimento prévio da distribuição dos dados a serem estimados [103].

O modelo de rede neural mais adequado para otimizar funções é a rede de Hopfield [9] e suas modificações, que servem como base para a maioria dos algoritmos de restauração de imagens baseados em redes neurais. O desenvolvimento de algoritmos para restauração

de imagens, baseados em redes de Hopfield modificadas teve início com o trabalho de Zhou et al. [104], que apresentaram dois algoritmos para restaurar imagens degradadas. O processo de restauração através dos algoritmos propostos em [104] consiste de dois estágios: no primeiro, os parâmetros da rede são estimados, e no segundo a imagem é restaurada iterativamente utilizando os algoritmos propostos.

Paik e Katsaggelos [105] utilizaram a rede de Hopfield para desenvolver diversos algoritmos para restauração de imagens, com diferentes métodos de atualização dos estados da rede. Figueiredo e Leitão [106] desenvolveram implementações de algoritmos iterativos para restauração de imagens, adequados para implementação em ambientes paralelos e distribuídos, baseados em redes de Hopfield modificadas. Perry e Guan [107] propuseram dois esquemas de particionamento de imagens e desenvolveram redes de Hopfield modificadas, baseadas nos esquemas de particionamento propostos, para processamento em tempo real, tirando proveito do paralelismo inerente a estas redes.

Mais recentemente, Sun [108, 109] utilizou uma rede de Hopfield para restaurar imagens utilizando os critérios de atualização dos neurônios da rede apresentados em [110], que são conhecidos como critérios EHE[1], e tem por finalidade selecionar, em cada iteração, o neurônios da rede que serão atualizados. Zenari e Achour [111] propuseram dois algoritmos baseados em uma uma modificação da rede de Hopfield para resolver o problema de restauração de imagens. Wu et al. [112], considerando imagens distorcidas, porém livres de ruído, desenvolveram um algoritmo seqüencial para restauração de imagens baseado em uma rede de Hopfield modificada, utilizando o método de determinação do passo do algoritmo proposto por Perry [113], e a regra de atualização de estados da rede proposta em [114], que é uma modificação da regra de atualização de Paik e Katsaggelos [105].

A técnica de *fusão de imagens* consiste em realçar dados úteis em imagens através de técnicas de fusão de dados [49]. O problema de fusão de dados consiste na integração de medidas de dados obtidas por diferentes sensores em diferentes instantes, quando a informação sobre os dados originais não está disponível ou possui incertezas, e é formulado através de um problema de programação quadrática com restrições lineares [115]. Xia e Wang [49] aplicaram a técnica de fusão de imagens para restaurar uma imagem corrompida por ruído, utilizando uma rede neural recorrente desenvolvida para resolver problemas gerais de programação convexa com restrições lineares.

[1]sigla de *eliminating highest error*

Um segmento de grande importância é a reconstrução de imagens a partir de projeções. Este tipo de aplicação está presente em muitos campos da ciência, como imagens geradas por tomografia computadorizada, raios-X e ultrasonografia [116]. Uma rede neural, baseada no modelo de Hopfield, aplicável ao problema de reconstrução de imagens a partir de projeções foi apresentada por Cichocki et al. [117]. Gopal e Hebert [118] utilizaram uma rede de Hopfield para restaurar imagens produzidas por um tomógrafo.

Consideramos neste trabalho apenas imagens degradadas por distorções lineares e utilizamos o sistema gradiente (3.54), na página 58, para resolver o problema de restauração de imagens. A principal característica do sistema gradiente (3.54), é que este sistema determina uma solução em norma L_1 mínima do sistema de equações lineares que modela o problema de restauração, que é menos sensível a ruídos, enquanto que a maioria dos algoritmos iterativos para restauração de imagens determina uma solução de mínimos quadrados.

Sob a perspectiva de redes neurais, o sistema gradiente (3.54) é um modelo de uma rede neural com função de ativação descontínua, e é uma alternativa viável à formulação baseada em redes de Hopfield, comumente adotada na literatura. Cada equação do sistema gradiente representa um neurônio da rede, e todos os neurônios são atualizados simultaneamente em cada iteração.

6.1 Tipos de degradação das imagens

Uma imagem é formada pela propagação de energia luminosa a partir de um objeto, que passa por um sistema de formação ou aquisição de imagens, como as lentes de uma câmera ou o próprio olho humano, construindo assim uma imagem do objeto original; um esquema deste processo é mostrado na figura 6.1.

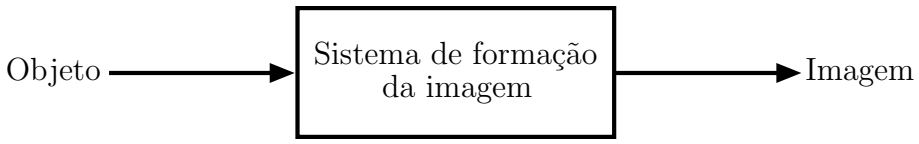
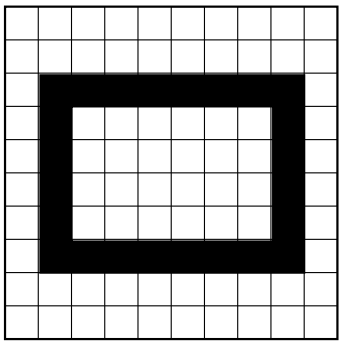


Figura 6.1: Diagrama ilustrando esquematicamente o processo de formação de uma imagem a partir de um objeto

Imagens digitais em tons de cinza são armazenadas sob a forma de matrizes, em que cada componente representa a intensidade do tom de cinza na posição correspondente. As

intensidades dos tons de cinza variam de 0 (mais escuro) até 255 (mais claro). Na figura 6.2, é mostrada uma imagem de tamanho 10×10 e sua representação matricial.



(a)

255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255
255	000	000	000	000	000	000	000	000	255
255	000	255	255	255	255	255	255	000	255
255	000	255	255	255	255	255	255	000	255
255	000	255	255	255	255	255	255	000	255
255	000	255	255	255	255	255	255	000	255
255	000	000	000	000	000	000	000	000	255
255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255

(b)

Figura 6.2: Representação de uma imagem digital: (a) Imagem de 10×10 pixels mostrada em um grid; (b) Matriz de tons de cinza correspondente à imagem, os componentes com valor 255, que são os tons mais claros, compõem o fundo da imagem e os componentes com valor 000 compõem as linhas pretas que formam o retângulo.

As degradações que ocorrem na imagem gerada são introduzidas pelo sistema de formação de imagens e/ou por fatores do ambiente em que é feita a aquisição da imagem, como variações de temperatura. As formas mais comuns de degradação são resolução finita dos sensores utilizados na aquisição da imagem, embaçamento devido ao movimento do sensor ou do alvo no momento da aquisição, refração e embaçamento provocado por problemas de foco [106]. As distorções em imagens podem ser classificadas em duas categorias [119]:

- Variantes no espaço — a distorção sofrida depende da região da imagem. Este tipo de distorção ocorre por problemas no sistema de aquisição ou por fatores externos;
- Invariantes no espaço — a distorção é a mesma em toda a imagem. Este tipo de distorção pode ser devido, por exemplo, a problemas de foco ou movimento da câmera.

Neste trabalho, consideramos distorções invariantes no espaço e lineares, pois os modelos matemáticos são consideravelmente simplificados, sendo adequados à aplicação das técnicas descritas nos capítulos anteriores e muitas distorções não-lineares podem ser aproximadas por distorções lineares e invariantes no espaço.

As distorções lineares são descritas matematicamente por uma classe de funções bidimensionais conhecida como *função de espalhamento de ponto* (PSF[2]). Esta denominação é devida ao fato que todo sistema ótico provoca algum grau de espalhamento (embaçamento)

[2]da denominação em inglês *point spread function*

em cada ponto de luz que o atravessa. O grau de espalhamento provocado determina a qualidade do sistema ótico utilizado [120].

O efeito da aplicação uma determinada PSF em uma imagem varia com o tipo e com a ordem da PSF. Quanto maior a ordem mais visível é o efeito da PSF quando aplicada a uma imagem. Quanto ao tipo, a PSF é selecionada de forma a produzir um tipo de degradação como, por exemplo, embaçamento da imagem original. A figura 6.3 mostra a imagem degradada através da PSF de dimensão 5×5, dada em (6.1), quando aplicada à imagem original.

$$\mathbf{PSF} = \begin{pmatrix} 0.1 & 0.0 & 0.1 & 0.0 & 0.1 \\ 0.0 & 0.1 & 0.0 & 0.1 & 0.0 \\ 0.1 & 0.0 & 0.1 & 0.0 & 0.1 \\ 0.0 & 0.1 & 0.0 & 0.1 & 0.0 \\ 0.1 & 0.0 & 0.1 & 0.0 & 0.1 \end{pmatrix} \tag{6.1}$$

O efeito provocado pela aplicação da PSF em (6.1) à imagem original é um exemplo de uma distorção invariante no espaço, pois provoca um embaçamento uniforme em toda a imagem, como pode ser visto na figura 6.3. A PSF é aplicada em cada ponto da imagem, e atua nos pontos vizinhos ao ponto central de aplicação, provocando o efeito de embaçamento, mostrado na figura 6.3. Quanto maior a dimensão da PSF, mais visível é a distorção produzida, pois uma PSF de dimensão maior atua em um número maior de pontos em torno do ponto em que é aplicada.

Como não existem sistemas de formação de imagens perfeitos, toda imagem formada por um sistema de formação de imagens é uma versão degradada da imagem original do objeto; o processo de restauração de imagens tem como objetivo obter uma estimativa da imagem original, livre de degradações. Na prática, devido à imperfeição dos instrumentos óticos, o que se considera como imagem original é, de fato, uma imagem tomada como de referência, que possui um nível de degradação suficientemente baixo que permite que seja considerada isenta de degradações.

Figura 6.3: Imagem degradada por embaçamento, provocado pela aplicação da PSF dada em (6.1) sobre a imagem original

6.2 Modelo matemático de degradação

Considere uma imagem degradada bidimensional $M \times N$, representada por $g(x, y)$. Se $x \leq 3$, esta notação indica que cada $g(x, y)$ é componente da seguinte matriz:

$$g(x,y) = \begin{pmatrix} g(1,1) & g(1,2) & g(1,3) \\ g(2,1) & g(2,2) & g(2,3) \\ g(3,1) & g(3,2) & g(3,3) \end{pmatrix}.$$

Seja $\hat{f}(x, y)$ a imagem original de ordem $M \times N$. O modelo matemático de degradação é dado pela convolução da imagem original com a PSF, mais um ruído aditivo, cuja formulação discreta para distorções lineares e invariantes no espaço é dada por

$$g(x,y) = \sum_{\xi=0}^{M-1} \sum_{\eta=0}^{N-1} h(x-\xi, y-\eta)\hat{f}(\xi,\eta) + n(x,y), \tag{6.2}$$

em que $h(x - \xi, y - \eta)$ é a PSF, $n(x, y)$ é o ruído, $x = 0, \ldots, M - 1$ e $y = 0, \ldots, N - 1$.

Organizando lexicograficamente as imagens, que consiste em empilhar as linhas de $g(x, y)$ e $\hat{f}(x, y)$ em vetores $\mathbf{g}$ e $\hat{\mathbf{f}}$, respectivamente, o modelo de degradação é dado por:

$$\mathbf{g} = \mathbf{H}\hat{\mathbf{f}} + \mathbf{n}, \tag{6.3}$$

sendo que $\mathbf{g} \in \mathbb{R}^{MN}$, $\hat{\mathbf{f}} \in \mathbb{R}^{MN}$, $\mathbf{n} \in \mathbb{R}^{MN}$ é o vetor de ruído e $\mathbf{H} \in \mathbb{R}^{MN \times MN}$ é a matriz de degradação, formada por um arranjo dos elementos da PSF, de tal maneira que o produto $\mathbf{H}\hat{\mathbf{f}}$ executa a convolução discreta entre a PSF e a imagem original $\hat{f}(x, y)$.

A PSF normalmente é um filtro passa-baixa [119], cujos elementos são estimativas da PSF do sistema que obteve a imagem ou que provocou a degradação. O ruído, representado pelo vetor $\mathbf{n}$, é considerado estático e distribuído uniformemente em toda a área da imagem. Considerando uma PSF quadrada $P \times P$, com $P << \min(M, N)$, para distorções invariantes no espaço, a matriz $\mathbf{H}$ toma a forma de uma matriz bloco-Toeplitz [119]. Uma matriz quadrada $\bar{\mathbf{H}}$ é Toeplitz, se o seu (i, j)-ésimo elemento é dado por $\bar{h}_{i-j}$, isto é

$$\bar{\mathbf{H}} = \begin{pmatrix} \bar{h}_0 & \bar{h}_{-1} & \dots & \bar{h}_{2-n} & \bar{h}_{1-n} \\ \bar{h}_1 & \bar{h}_0 & \bar{h}_{-1} & \dots & \bar{h}_{2-n} \\ \vdots & \bar{h}_1 & \bar{h}_0 & \ddots & \vdots \\ \bar{h}_{n-2} & \vdots & \ddots & \ddots & \bar{h}_{-1} \\ \bar{h}_{n-1} & \bar{h}_{n-2} & \cdots & \bar{h}_1 & \bar{h}_0 \end{pmatrix}. \tag{6.4}$$

Uma matriz $\mathbf{H}$ é dita bloco-Toeplitz, se é uma matriz de blocos, tal que o (i, j)-ésimo bloco de $\mathbf{H}$ é uma matriz Toeplitz dada por $\mathbf{H}_{i-j}$. Um exemplo de uma matriz bloco-Toeplitz é dado em (6.5).

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_0 & \mathbf{H}_{-1} & \mathbf{H}_{-2} & \mathbf{H}_{-3} \\ \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{H}_{-1} & \mathbf{H}_{-2} \\ \mathbf{H}_2 & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{H}_{-1} \\ \mathbf{H}_3 & \mathbf{H}_2 & \mathbf{H}_1 & \mathbf{H}_0 \end{pmatrix}. \tag{6.5}$$

Observe que, a matriz Toeplitz em (6.4) é completamente determinada apenas pela sua primeira linha e pela sua primeira coluna. Por outro lado, a matriz bloco-Toeplitz em (6.5) é completamente determinada apenas pelo seu primeiro bloco-linha e pelo seu primeiro bloco-coluna. Como cada bloco $\mathbf{H}_k$, $k = i - j$, é Toeplitz, conseqüentemente, a matriz bloco-Toeplitz em (6.5) é completamente determinada apenas pelos componentes da sua primeira linha e da sua primeira coluna. Este fato possibilita a minimização da quantidade de memória necessária para processar matrizes bloco-Toeplitz, já que apenas a primeira linha e a primeira coluna precisam ser armazenadas.

Caso a PSF não seja conhecida a priori, a estimativa da PSF é feita a partir do conheci-

mento prévio do tipo de degradação a que a imagem original é submetida, o que significa que quanto mais exato for o conhecimento dos processos que degradam a imagem original, melhor será o resultado produzido pela técnica de restauração utilizada.

Como as imagens são formadas a partir de energia luminosa, a representação matemática precisa satisfazer às seguintes restrições de não-negatividade [121]:

$$\hat{\mathbf{f}} \geq \mathbf{0} \tag{6.6}$$

$$\mathbf{g} \geq \mathbf{0} \tag{6.7}$$

$$h_{ij} \geq 0 \tag{6.8}$$

em que $h_{i,j}$ são os componentes da matriz bloco-Toeplitz $\mathbf{H}$.

O problema de restaurar imagens degradadas por ruído ou outras formas de degradação, como por exemplo embaçamento provocado pelo movimento da câmera ou do alvo no momento da aquisição, pode ser modelado como um problema de otimização [120], que em geral possui dimensões elevadas.

6.3 Medida de degradação

Uma das dificuldades em resolver o problema de restauração de imagens é o mal-condicionamento da matriz $\mathbf{H}$ [121]. Miller [122] propôs métodos baseados em mínimos quadrados para a resolução de uma classe problemas mal-condicionados, e que aplica-se ao problema de restauração de imagens. O problema em que estamos interessados é dado como segue [122]: encontrar um vetor $\mathbf{f}$ que satisfaz:

$$\|\mathbf{g} - \mathbf{H}\mathbf{f}\| \leq \varepsilon \tag{6.9}$$

$$\|\mathbf{D}\mathbf{f}\| \leq e \tag{6.10}$$

sendo que, no contexto de restauração de imagens, $\varepsilon > 0$ é uma estimativa da norma do vetor de ruídos [123], $e > 0$ é uma constante conhecida e $\mathbf{D}$ é um operador de suavização da imagem.

De acordo com Miller [122], resolver (6.9)-(6.10) equivale a resolver o seguinte problema

programação quadrática sem restrições

$$\text{minimizar } F(\mathbf{f}) = \frac{1}{2}(\|\mathbf{g} - \mathbf{H}\mathbf{f}\|^2 + \lambda\|\mathbf{D}\mathbf{f}\|^2). \tag{6.11}$$

Este procedimento também é conhecido como *regularização de Tikhonov* [124]. No problema de restauração de imagens, a função objetivo F do problema (6.11), fornece uma medida da degradação a que a imagem é submetida [119]. Neste contexto $\mathbf{f}$ é uma estimativa da imagem original $\hat{\mathbf{f}}$, $\lambda > 0$ é um parâmetro constante e $\mathbf{D}$ é um operador de suavização da imagem.

A escolha do parâmetro λ interfere diretamente na estimativa final $\mathbf{f}$. Se λ for muito pequeno a imagem restaurada apresentará maior nitidez, porém estará muito distorcida por ruído. Por outro lado, se λ for muito grande, a atenuação do ruído será mais eficiente, porém a imagem restaurada perderá nitidez e apresentará um grau de embaçamento maior.

Observe que, de acordo com a equação (6.3), o termo $\|\mathbf{g} - \mathbf{H}\mathbf{f}\|^2$ é o quadrado da norma euclideana do vetor de ruídos $\mathbf{n}$. Portanto, o problema de minimizar a função quadrática em (6.11), envolve a minimização do sinal de ruído introduzido na imagem, dado pelo primeiro termo desta função. O segundo termo é de regularização, e ajuda a reduzir os efeitos do mal condicionamento da matriz $\mathbf{H}$.

Expressando a função F na forma padrão de mínimos quadrados, o problema (6.11) pode ser escrito na forma [105]:

$$\text{minimizar } F(\mathbf{f}) = \frac{1}{2}(\mathbf{f}^T\mathbf{A}\mathbf{f} - \mathbf{b}^T\mathbf{f}), \tag{6.12}$$

em que $\mathbf{A} = \mathbf{H}^T\mathbf{H} + \lambda\mathbf{D}^T\mathbf{D}$ e $\mathbf{b} = \mathbf{H}^T\mathbf{g}$.

Observe que $\nabla F(\mathbf{f}) = \mathbf{A}\mathbf{f} - \mathbf{b}$, e que o mínimo de F em (6.12) ocorre quando $\mathbf{f}$ é solução do seguinte sistema de equações lineares:

$$\mathbf{A}\mathbf{f} = \mathbf{b}. \tag{6.13}$$

Logo, resolver o problema de otimização (6.12) equivale a resolver o sistema de equações lineares (6.13). Observe que como λ é uma constante positiva, se $\mathbf{D}$ for não-singular, então a matriz $\mathbf{A}$ também é não-singular, satisfazendo as condições do teorema 3.6. Portanto, o sistema de equações lineares (6.13), pode ser resolvido pelo sistema gradiente (3.54). Além

disso, a matriz $\mathbf{A}$ em (6.12) é mais bem condicionada do que $\mathbf{H}$ ou $\mathbf{H}^T\mathbf{H}$, desde que a matriz $\mathbf{D}$ seja escolhida adequadamente [123], e é mais adequada aos métodos iterativos para restauração de imagens.

6.4 Restauração através da solução em norma L_1 do sistema de equações lineares

A solução em norma L_1 do sistema de equações lineares (6.13) apresenta diversas vantagens em relação à solução de mínimos quadrados, como menor sensibilidade a ruídos e *outliers* [52], pois quando a norma L_2 é usada, dados contaminados com ruído são elevados ao quadrado, contribuindo mais para o erro total (quadrático), o que não acontece com a solução em norma L_1. Adicionalmente, a solução em norma L_1 é a mais indicada quando a distribuição do ruído é pouco conhecida [125].

Diferentemente da maioria dos algoritmos iterativos para restauração de imagens, que determinam uma solução de mínimos quadrados do sistema de equações lineares (6.13), optamos pela resolução utilizando o sistema gradiente (3.54), que determina uma solução em norma L_1 mínima do sistema de equações lineares (6.13).

Como o vetor $\mathbf{f}$ tem que satisfazer a restrição de sinal (6.6), precisamos encontrar uma solução não-negativa de norma mínima para o sistema (6.13). Como no problema de restauração de imagens a matriz $\mathbf{A}$, definida em (6.12), é simétrica, então o sistema gradiente (3.54) toma a forma:

$$\dot{\mathbf{f}} = -\mathbf{A}\,\text{sgn}(\mathbf{A}\mathbf{f} - \mathbf{b}), \tag{6.14}$$

sendo que os vetores $\mathbf{f}$ e $\mathbf{b}$ são definidos em (6.12).

A restrição de sinal (6.6) é implementada limitando a saída do integrador entre um limite mínimo lb e um máximo ub. Em processamento de imagens estes limites são $lb = 0$ e $ub = 255$, que é o intervalo dos valores da intensidade de tons de cinza em cada ponto da imagem.

A limitação da saída do integrador é feita através de um operador que: i) se o valor da saída do integrador está entre os limites lb e ub, então a saída é o próprio valor computado pelo algoritmo de integração; ii) caso o valor da saída do integrador seja menor que o limite inferior ou maior que o limite superior, o valor da saída é fixado no limite inferior ou no

limite superior, respectivamente [117]. Através desta abordagem, a restrição (6.6) não é violada durante o processo de otimização, e a implementação desta limitação pode ser feita através do operador de projeção utilizado em Xia e Wang [126]:

$$P(x) = \begin{cases} lb, \text{ se } x < lb \\ x, \text{ se } x \in [lb, ub] \\ ub, \text{ se } x > ub \end{cases} \tag{6.15}$$

6.5 Implementação do sistema de equações lineares para restauração de imagens

Os problemas de otimização que modelam o processo de restauração de imagens normalmente apresentam dimensões elevadas. Para ilustrar este fato, considere uma imagem degradada $g(x, y)$ e sua estimativa sem degradação $f(x, y)$, ambas de 512 por 512 pontos. Neste caso, a dimensão dos vetores $\mathbf{g}$ e $\mathbf{f}$ é 512^2, e a matriz de degradação $\mathbf{H}$ tem dimensão $512^2 \times 512^2$, isto é, a matriz $\mathbf{H}$ possui aproximadamente 68.7×10^9 elementos. Isto significa que a matriz de coeficientes $\mathbf{A}$ do sistema de equações lineares (6.13) também possui aproximadamente 68.7×10^9 elementos.

Este dado demonstra que a restauração de imagens é um problema que exige muito poder computacional, e o desenvolvimento de algoritmos paralelos para abordar este tipo de problema é de fundamental importância. O sistema gradiente (6.14) oferece uma alternativa bastante promissora neste sentido, pois além de sua estrutura simples, este sistema também é facilmente paralelizável.

O sistema gradiente (6.14) pode ser resolvido através de algoritmos de integração numérica. Para viabilizar a utilização de algoritmos de integração com passo adaptativo, do mesmo modo que no capítulo 3, utilizamos a técnica da camada limite para aproximar o termo descontínuo do sistema gradiente (6.14) por uma função contínua em uma vizinhança da superfície de descontinuidade, dada por $\{\mathbf{x} : \mathbf{Ax} = \mathbf{b}\}$. Dois algoritmos de integração foram utilizados para resolver o sistema gradiente: os métodos de Runge-Kutta e de Euler.

Método de Runge-Kutta de segunda ordem

O método Runge-Kutta de segunda ordem é um método de dois estágios, exigindo duas avaliações do segundo membro dos sistema de EDOs. Utilizamos este método com os coeficientes obtidos em [65], juntamente com o algoritmo de controle de passo via PID [66], obtendo resultados com precisão satisfatória.

O pseudo-código do método de Runge-Kutta de segunda ordem é dado no algoritmo 3.3, e os coeficientes de Cash-Karp são dados sob a forma de tableau em (3.61), ambos na página 64. A atualização do passo de integração é feita pela fórmula (3.62), com os parâmetros sugeridos em [66], dados em (5.42), na página 114.

Método de Euler com passo adaptativo

O método de Euler com controle de passo através da fórmula de Barzilai-Borwein [59] (Euler-BB), descrito na seção 3.9.1, é mais adequado à resolução de problemas de grande porte, pois exige apenas uma avaliação do segundo membro do sistema gradiente (6.14) em cada iteração. O custo computacional do cálculo do passo é baixo, pois exige apenas a determinação de dois produtos internos e duas operações de subtração de vetores, que podem ser feitos em paralelo.

Em termos de utilização de memória, no problema de processamento de imagens, considerando uma imagem de dimensão $M \times N$, e a utilização da precisão dupla nos cálculos, a quantidade adicional de memória utilizada é $16MN$ bytes. Para uma imagem de dimensão 512×512 pontos, a requisição adicional de memória é de apenas 4 megabytes, que é uma quantidade significativamente pequena.

O desempenho do sistema gradiente (6.14) é comparado com o desempenho do método dos gradientes conjugados, descrito a seguir.

Método dos gradientes conjugados

O método dos gradientes conjugados, proposto por Hestenes e Stiefel [127], é um dos métodos mais eficientes para a resolução de sistemas de equações lineares. Este método apresenta como principais atrativos o baixo custo computacional, e a alta taxa de convergência [128], sendo adequado à resolução de sistemas de equações lineares de grande porte. O método dos gradientes conjugados é freqüentemente utilizado para resolver sistemas de

Algoritmo 6.1 Método dos gradientes conjugados para a resolução de sistemas de equações lineares

$\mathbf{f}_0$ = condição inicial
$\mathbf{r}_0 = \mathbf{A}\mathbf{f}_0 - \mathbf{b}$ (resíduo inicial)
$\mathbf{s}_0 = \mathbf{r}_0$ (direção inicial de busca)
`Para` $k = 0, 1, 2, \ldots$
    $\alpha_k = \mathbf{r}_k^T \mathbf{r}_k / \mathbf{s}_k^T \mathbf{A}\mathbf{s}_k$ (calcula o passo do algoritmo)
    $\mathbf{f}_{k+1} = \mathbf{f}_k + \alpha_k \mathbf{s}_k$ (atualiza solução)
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{s}_k$ (calcula novo resíduo)
    $\beta_{k+1} = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} / \mathbf{r}_k^T \mathbf{r}_k$
    $\mathbf{s}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{s}_k$ (calcula nova direção de busca)
    $\mathbf{r}_k = \mathbf{r}_{k+1}$; $\mathbf{s}_k = \mathbf{s}_{k+1}$ (atualiza variáveis antigas)
    $k = k + 1$ (atualiza número da iteração)
`Fim`

equações lineares decorrentes de problemas processamento de imagens e sinais [129–133], que são mal-condicionados e, em geral, apresentam dimensões elevadas.

O método dos gradientes conjugados original, como proposto por Hestenes e Stiefel, é descrito no algoritmo 6.1. Em cada iteração do método dos gradientes conjugados, é determinada uma nova direção de busca $\mathbf{s}_{k+1}$ que satisfaz:

(i) $\mathbf{s}_{k+1}$ é não-ortogonal ao vetor de resíduos computado na k-ésima iteração, isto é, $\mathbf{s}_{k+1}^T \mathbf{r}_k \neq 0$;

(ii) $\mathbf{s}_{k+1}$ é $\mathbf{A}$-conjugada, isto é, $\mathbf{s}_i^T \mathbf{A}\mathbf{s}_{k+1} = 0$, sendo que $i = 1, \ldots, k$.

O parâmetro α_k, garante que a condição (i) é satisfeita. Por outro lado, o parâmetro β_k, garante que a nova direção de busca $\mathbf{s}_{k+1}$ é $\mathbf{A}$-conjugada, satisfazendo a condição (ii). Os parâmetros α_k e β_{k+1}, também garantem que os resíduos determinados pelo método dos gradientes conjugados, após $k-1$ passos, são mutuamente ortogonais [128, Teo. 10.2.3].

Quando a matriz de coeficientes $\mathbf{A}$ do sistema de equações lineares (6.13) é mal-condicionada, a taxa de convergência do método dos gradientes conjugados é severamente reduzida. Com o objetivo de aumentar a taxa de convergência, o algoritmo é implementado utilizando um pré-condicionador. O pré-condicionador é uma matriz não-singular $\mathbf{C}$, definida de tal maneira que as propriedades espectrais da matriz $\mathbf{C}^{-1}\mathbf{A}$ sejam melhores que as da matriz $\mathbf{A}$. Neste caso, ao invés do sistema (6.13), o método dos gradientes conjugados é aplicado a $\mathbf{C}^{-1}\mathbf{A}\mathbf{f} = \mathbf{C}^{-1}\mathbf{b}$. O método dos gradientes conjugados com pré-condicionamento é descrito no algoritmo 6.2.

Algoritmo 6.2 Método dos gradientes conjugados para a resolução de sistemas de equações lineares pré-condicionado por uma matriz definida-positiva $\mathbf{C}$

$\mathbf{f}_0$ = condição inicial
$\mathbf{r}_0 = \mathbf{A}\mathbf{f}_0 - \mathbf{b}$ (resíduo inicial)
$\mathbf{z}_0 = \mathbf{C}^{-1}\mathbf{r}_0$
$\mathbf{s}_0 = \mathbf{z}_0$ (direção de busca inicial)
`Para` $k = 0, 1, 2, \ldots$
    $\alpha_k = \mathbf{r}_k^T\mathbf{z}_k / \mathbf{s}_k^T\mathbf{A}\mathbf{s}_k$ (calcula passo do algoritmo)
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{s}_k$ (calcula novo resíduo)
    $\mathbf{z}_{k+1} = \mathbf{C}^{-1}\mathbf{r}_{k+1}$
    $\beta_{k+1} = \mathbf{r}_{k+1}^T\mathbf{z}_{k+1} / \mathbf{r}_k^T\mathbf{z}_k$
    $\mathbf{s}_{k+1} = \mathbf{z}_{k+1} + \beta_{k+1}\mathbf{s}_k$ (calcula nova direção de busca)
    $\mathbf{f}_{k+1} = \mathbf{f}_k + \alpha_k\mathbf{s}_{k+1}$ (atualiza solução)
    $\mathbf{r}_k = \mathbf{r}_{k+1}$; $\mathbf{s}_k = \mathbf{s}_{k+1}$; $\mathbf{f}_{k+1} = \mathbf{f}_k$; $\mathbf{z}_k = \mathbf{z}_{k+1}$ (atualiza variáveis antigas)
    $k = k + 1$ (incrementa o número da iteração)
`Fim`

No problema de restauração de imagens, a matriz de coeficientes $\mathbf{A}$ do sistema de equações lineares (6.13), é mal-condicionada. Esta matriz é formada a partir da aplicação da regularização de Tikhonov ao sistema bloco-Toeplitz (6.3), em que o produto da matriz mal-condicionada $\mathbf{H}$ pelo vetor $\mathbf{f}$, executa a convolução da PSF com a imagem original. Para problemas envolvendo matrizes Toeplitz, o pré-condicionador circulante é o mais adequado [133]. Para maiores detalhes sobre a aplicação do método dos gradientes conjugados para sistemas Toeplitz, ver por exemplo [134].

Uma matriz $\bar{\mathbf{C}}$ de dimensão $n \times n$ é dita circulante, se é Toeplitz e se $\bar{c}_k = c_{n-k}$, $k = 0, \ldots, n-1$. Um exemplo de uma matriz circulante $\bar{\mathbf{C}}$ de ordem 4×4 é dado em (6.16).

$$\bar{\mathbf{C}} = \begin{pmatrix} \bar{c}_0 & \bar{c}_3 & \bar{c}_2 & \bar{c}_1 \\ \bar{c}_1 & \bar{c}_0 & \bar{c}_3 & \bar{c}_2 \\ \bar{c}_2 & \bar{c}_1 & \bar{c}_0 & \bar{c}_3 \\ \bar{c}_3 & \bar{c}_2 & \bar{c}_1 & \bar{c}_0 \end{pmatrix} \tag{6.16}$$

As matrizes circulantes são diagonalizadas pela matriz de Fourier $\mathbf{F}$ [134], neste caso, a matriz $\bar{\mathbf{C}}$ pode ser escrita como:

$$\bar{\mathbf{C}} = \mathbf{F}^H\mathbf{\Lambda}\mathbf{F},$$

em que a matriz $\mathbf{\Lambda}$ é diagonal, formada pelos autovalores da matriz $\bar{\mathbf{C}}$, e $\mathbf{F}^H$ é a transposta hermitiana da matriz de Fourier $\mathbf{F}$. Como a matriz de Fourier $\mathbf{F}$ é unitária, então $\mathbf{F}^H = \mathbf{F}^{-1}$.

Conseqüentemente, a inversa dos pré-condicionadores circulantes pode ser obtida utilizando a transformada rápida de Fourier (FFT[3]). O primeiro pré-condicionador circulante para matrizes Toeplitz foi proposto por Strang [135]. Este pré-condicionador foi posteriormente estendido para matrizes quadradas quaisquer por Chan et al. [136]. O pré-condicionamento de matrizes bloco-Toeplitz e da matriz $\mathbf{A}$, resultante da regularização de Tikhonov, foi abordado, por exemplo, em [131] e [137].

Construção do pré-condicionador: observe que o sistema de equações lineares (6.13) pode ser fatorado na forma

$$\tilde{\mathbf{A}}^T(\tilde{\mathbf{A}}\mathbf{f} - \tilde{\mathbf{b}}) = \mathbf{0}, \tag{6.17}$$

sendo que $\tilde{\mathbf{A}} := [\mathbf{H}^T \quad \lambda^{1/2}\mathbf{D}^T]^T$ é uma matriz $2MN \times MN$, e $\tilde{\mathbf{b}} := [\mathbf{g}^T\ \mathbf{0}]^T$ é um vetor de dimensão $2MN$. As matrizes $\mathbf{H}$ e $\mathbf{D}$ são matrizes bloco-Toeplitz de dimensão $MN \times MN$, em que cada bloco possui dimensão $N \times N$, e λ é o parâmetro de regularização.

Em [137], é descrito um procedimento prático para obter pré-condicionadores circulantes por blocos para sistemas de equações lineares na forma (6.17), baseado no pré-condicionador circulante de Chan [138]. O pré-condicionador de Chan é uma aproximação circulante $\mathbf{C}$ de uma matriz quadrada $\mathbf{T}$, em relação à norma de Frobenius, isto é,

$$\mathbf{C} = \text{argmin}\, \mathcal{F}(\mathbf{X}) = \|\mathbf{X} - \mathbf{T}\|_F,$$

no conjunto de todas as matrizes circulantes $\mathbf{X}$.

Quando $\mathbf{T}$ é uma matriz Toeplitz de dimensão $N \times N$, o (i, j)-ésimo elemento do pré-condicionador de Chan [138], é dado por:

$$c_k = \begin{cases} \dfrac{(N-k)t_k + kt_{k-N}}{N}, \text{ se } 0 \leq k < N, \\ c_{N+k}, \text{ se } 0 < -k < N \end{cases} \tag{6.18}$$

sendo que $k = i - j$, e t_k é o elemento da k-ésima diagonal da matriz Toeplitz $\mathbf{T}$.

A fórmula (6.18) é utilizada em cada bloco das matrizes $\mathbf{H}$ e $\mathbf{D}$ em (6.17), resultando nas matrizes $\mathbf{C}_{\mathbf{H}}^{(1)}$ e $\mathbf{C}_{\mathbf{D}}^{(1)}$, em que cada bloco destas matrizes é uma matriz circulante. As matrizes $\mathbf{C}_{\mathbf{H}}^{(1)}$ e $\mathbf{C}_{\mathbf{D}}^{(1)}$ são as aproximações de nível 1 para $\mathbf{H}$ e $\mathbf{D}$, respectivamente [137].

Em seguida, cada bloco de $\mathbf{C}_{\mathbf{H}}^{(1)}$ e $\mathbf{C}_{\mathbf{D}}^{(1)}$ é tratado como um elemento destas matrizes,

[3]Sigla de Fast Fourier Transform

então, sob esta perspectiva, $\mathbf{C}_{\mathbf{H}}^{(1)}$ e $\mathbf{C}_{\mathbf{D}}^{(1)}$ têm dimensão $M \times M$. Aplicando a fórmula (6.18), obtemos as aproximações de nível 2 $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\mathbf{C}_{\mathbf{D}}^{(2)}$, para $\mathbf{H}$ e $\mathbf{D}$, respectivamente, dadas por:

$$\mathbf{C}_{\mathbf{H}}^{(2)} = \begin{cases} \dfrac{(M-k)[\mathbf{C}_{\mathbf{H}}^{(1)}]_k + k[\mathbf{C}_{\mathbf{H}}^{(1)}]_{k-M}}{M}, \text{ se } 0 \leq k < M, \\ [\mathbf{C}_{\mathbf{H}}^{(2)}]_{M+k}, \text{ se } 0 < -k < M \end{cases}$$

$$\mathbf{C}_{\mathbf{D}}^{(2)} = \begin{cases} \dfrac{(M-k)[\mathbf{C}_{\mathbf{D}}^{(1)}]_k + k[\mathbf{C}_{\mathbf{D}}^{(1)}]_{k-M}}{M}, \text{ se } 0 \leq k < M, \\ [\mathbf{C}_{\mathbf{D}}^{(2)}]_{M+k}, \text{ se } 0 < -k < M \end{cases}$$

sendo que $[\mathbf{T}]_k$ é o bloco que determina a k-ésima bloco-diagonal da matriz bloco-Toeplitz $\mathbf{T}$.

As matrizes $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\mathbf{C}_{\mathbf{D}}^{(2)}$ são circulantes por blocos, com blocos circulantes. Segundo Chan e Jin [139, Teorema 3], estas matrizes satisfazem:

$$\mathbf{C}_{\mathbf{H}}^{(2)} = (\mathbf{F} \otimes \mathbf{F})^H \mathbf{\Lambda}_{\mathbf{H}} (\mathbf{F} \otimes \mathbf{F}) \tag{6.19}$$

$$\mathbf{C}_{\mathbf{D}}^{(2)} = (\mathbf{F} \otimes \mathbf{F})^H \mathbf{\Lambda}_{\mathbf{D}} (\mathbf{F} \otimes \mathbf{F}) \tag{6.20}$$

sendo que o operador $\otimes$ indica o produto de Kronecker, e $\mathbf{\Lambda}_{\mathbf{H}}$ e $\mathbf{\Lambda}_{\mathbf{D}}$ são matrizes diagonais contendo os autovalores de $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\mathbf{C}_{\mathbf{D}}^{(2)}$, respectivamente.

A aproximação de nível 2 para a matriz $\lambda^{1/2}\mathbf{D}$ em (6.17) é dada por $\lambda^{1/2}\mathbf{C}_{\mathbf{D}}^{(2)}$. Segundo Chan et al. [137], o pré-condicionador $\mathbf{C}$ de nível 2, para o sistema (6.17) satisfaz

$$\mathbf{C}^T\mathbf{C} = (\mathbf{C}_{\mathbf{H}}^{(2)})^H \mathbf{C}_{\mathbf{H}}^{(2)} + \lambda (\mathbf{C}_{\mathbf{D}}^{(2)})^H \mathbf{C}_{\mathbf{D}}^{(2)}.$$

Utilizando (6.19) e (6.20), o pré-condicionador $\mathbf{C}$ é definido por [137]:

$$\mathbf{C} = (\mathbf{F} \otimes \mathbf{F})^H \mathbf{\Phi} (\mathbf{F} \otimes \mathbf{F}). \tag{6.21}$$

sendo que $\mathbf{\Phi} := (\mathbf{\Lambda}_{\mathbf{H}}^H \mathbf{\Lambda}_{\mathbf{H}} + \lambda \mathbf{\Lambda}_{\mathbf{D}}^H \mathbf{\Lambda}_{\mathbf{D}})^{1/2}$.

Observe que para construir o pré-condicionador $\mathbf{C}$ em (6.21), apenas os autovalores das matrizes $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\lambda\mathbf{C}_{\mathbf{D}}^{(2)}$, dados pelas matrizes diagonais $\mathbf{\Lambda}_{\mathbf{H}}$ e $\mathbf{\Lambda}_{\mathbf{D}}$, são necessários.

O pré-condicionador obtido é circulante por blocos, em que cada bloco é uma matriz circulante. Mostra-se que apenas a primeira coluna dos pré-condicionadores $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\mathbf{C}_{\mathbf{D}}^{(2)}$ são

necessárias para determinar as matrizes diagonais $\mathbf{\Lambda_H}$ e $\mathbf{\Lambda_D}$ [137]. Denotando por $[\mathbf{C}_{\mathbf{H}}^{(2)}]_{i,1}$ a primeira coluna da matriz $\mathbf{C}_{\mathbf{H}}^{(2)}$, os autovalores da matriz $\mathbf{C}_{\mathbf{H}}^{(2)}$ são determinados calculando-se a FFT em duas dimensões da primeira coluna desta matriz, isto é

$$\mathbf{\Lambda_H 1} = (\mathbf{F} \otimes \mathbf{F})^H [\mathbf{C}_{\mathbf{H}}^{(2)}]_{i,1},$$

sendo que $\mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbb{R}^{MN}$. Procedimento análogo é utilizado para determinar a matriz $\mathbf{\Lambda_D}$. De posse dos autovalores de $\mathbf{C}_{\mathbf{H}}^{(2)}$ e $\mathbf{C}_{\mathbf{D}}^{(2)}$, dados pelas matrizes $\mathbf{\Lambda_H}$ e $\mathbf{\Lambda_D}$, respectivamente, é possível determinar a matriz diagonal $\mathbf{\Phi}$, em (6.21).

Note que, em cada iteração do algoritmo 6.2, é efetuado o produto da inversa do pré-condicionador $\mathbf{C}$ por um vetor $\mathbf{r} \in \mathbb{R}^{MN}$. Este produto é feito através da fórmula

$$\mathbf{C}^{-1}\mathbf{r} = (\mathbf{F} \otimes \mathbf{F})^H \mathbf{\Phi}^{-1} (\mathbf{F} \otimes \mathbf{F})\mathbf{r},$$

que exige $O(2MN(\log(N) + \log(M)))$ operações, utilizando a FFT em duas dimensões [137].

6.6 Utilização da esparsidade das matrizes bloco-Toeplitz

A matriz bloco-Toeplitz $\mathbf{H}$ associada a uma imagem de dimensão 512×512 possui dimensão de $512^2 \times 512^2$. A quantidade de memória necessária para armazenar esta matriz, considerando que os dados de precisão dupla, é da ordem de 512 gigabytes. Uma matriz com estas dimensões é muito difícil de ser tratada computacionalmente, e devido a grande quantidade de memória exigida para o seu armazenamento, a sua criação explícita é inviável. No entanto, a matriz $\mathbf{H}$ possui as seguintes características:

(i) é esparsa;

(ii) é completamente determinada pela PSF utilizada.

Estas duas características permitem resolver de forma eficiente o problema da requisição de memória, e também da quantidade de operações aritméticas necessárias para a realização de cada iteração do algoritmo de integração utilizado.

A matriz bloco-Toeplitz $\mathbf{H}$ é gerada a partir da fórmula discreta de convolução da imagem com a PSF, e um procedimento prático para a geração desta matriz é descrito em [121].

Por este procedimento, a PSF não é centrada na origem, e tem como vantagem gerar matrizes Toeplitz $\mathbf{H}_i$ triangulares de banda; a matriz bloco-Toeplitz $\mathbf{H}$ também é triangular de banda por blocos. A conseqüência de não considerar a PSF centrada na origem é apenas um deslocamento na imagem $\mathbf{g}$, dada em (6.3), no valor da dimensão da PSF [121, pg. 218].

Como ilustração, considerando uma PSF de ordem 2 aplicada a uma imagem quadrada de 3×3 pontos, a matriz bloco-Toeplitz correspondente tem dimensão 9×9. O número de blocos não-nulos $\mathbf{H}_i$ é dado pela dimensão da PSF, e cada um dos blocos é uma matriz Toeplitz. Neste caso, a matriz bloco-Toeplitz é

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 \end{bmatrix}$$

Os primeiros elementos da primeira coluna de cada bloco $\mathbf{H}_i$ são os elementos da i-ésima linha da matriz que implementa a PSF, os demais elementos são nulos. A dimensão de cada bloco $\mathbf{H}_i$ é dada pela largura da imagem e o i-ésimo bloco não-nulo da matriz $\mathbf{H}$ é

$$\mathbf{H}_i = \begin{bmatrix} h_{i,1} & 0 & 0 \\ h_{i,2} & h_{i,1} & 0 \\ 0 & h_{i,2} & h_{i,1} \end{bmatrix} \tag{6.22}$$

sendo que $h_{i,j}$ são os elementos da matriz PSF, também conhecida como máscara da PSF.

Observe que a matriz $\mathbf{H}$ é esparsa e os blocos $\mathbf{H}_i$ também são esparsos. Considerando uma PSF de dimensão 3 aplicada à uma imagem de dimensão 512×512 resulta em uma matriz $\mathbf{H}$ triangular de banda por blocos, de ordem $512^2 \times 512^2$ que possui três bloco-diagonais não-nulos. Os blocos não-nulos $\mathbf{H}_i$ que compõem as diagonais de $\mathbf{H}$ também são matrizes triangulares de banda, com três diagonais.

Devido às dimensões da matriz $\mathbf{H}$ e da matriz $\mathbf{D}$, a criação da matriz $\mathbf{A}$ em (6.14) é inviável. No entanto, o vetor de resíduos $\mathbf{r}$ pode ser escrito como segue:

$$\mathbf{r} = \mathbf{H}^T(\mathbf{Hf}) + \lambda \mathbf{D}^T(\mathbf{Df}) - \mathbf{H}^T\mathbf{g}.$$

Utilizando a mesma notação, o sistema gradiente (6.14) pode ser escrito como:

$$\dot{\mathbf{f}} = -\mathbf{H}^T(\mathbf{H}\text{sgn}(\mathbf{r})) - \lambda \mathbf{D}^T(\mathbf{D}\text{sgn}(\mathbf{r})).$$

A estratégia adotada para realizar os cálculos discriminados acima é resolver primeiro os produtos entre parênteses, o que permite executar apenas produtos matriz-vetor. A esparsidade das matrizes $\mathbf{H}$ e $\mathbf{D}$ é explorada a partir da identificação das posições dos blocos não-nulos. A discussão a seguir é feita para a matriz $\mathbf{H}$, mas também é válida para a matriz $\mathbf{D}$, que também é uma matriz bloco-Toeplitz. Através da inspeção da matriz $\mathbf{H}$, concluimos que o produto desta matriz pelo vetor $\mathbf{f}$ é um vetor de blocos $\mathbf{y}$ cujo i-ésimo bloco é dado por:

$$\mathbf{y}_i = \sum_l \mathbf{H}_{|i-l|}\mathbf{f}_l, \quad l = -\min(0, d-i-1), \ldots, i \tag{6.23}$$

sendo que $i = 0, \ldots, M-1$ é o índice dos blocos-linha da matriz $\mathbf{H}$, l é o índice dos blocos-coluna de $\mathbf{H}$, M é a altura da imagem, d é a dimensão da PSF. O vetor de blocos $\mathbf{y}$ tem dimensão MN e a dimensão de cada bloco $\mathbf{y}_i$ é N.

A variação dos índices l em (6.23) mapeia corretamente os blocos não-nulos na matriz $\mathbf{H}$. Deste modo, apenas os produtos matriciais dos blocos não-nulos pela partição correspondente do vetor $\mathbf{f}$ são executados. A esparsidade dos blocos $\mathbf{H}_i$ é explorada utilizando as subrotinas BLAS de multiplicação de matrizes triangulares de banda por um vetor. Os blocos $\mathbf{H}_i$ são armazenados em uma forma compacta, em que apenas as diagonais não-nulas do triângulo inferior são armazenadas em uma matriz de ordem $d \times N$, sendo que N é a largura da imagem, e é suportada pelas subrotinas BLAS. Utilizando este sistema de armazenamento, o i-ésimo bloco, definido em (6.22), é armazenado como:

$$\begin{bmatrix} h_{i,1} & h_{i,j} & h_{i,1} \\ h_{i,2} & h_{i,2} & 0 \end{bmatrix} \tag{6.24}$$

Os produtos são feitos utilizando a subrotina `dtbmv` do BLAS, que é específica para a execução de produtos matriz-vetor, em que a matriz é triangular de banda e armazenada na forma compacta acima. Esta forma de armazenamento reduz ainda mais a quantidade de memória necessária para o armazenamento dos dados, e a utilização do BLAS aumenta o desempenho, tirando proveito da estrutura dos blocos $\mathbf{H}_i$. Este procedimento proporciona uma redução significativa na quantidade de memória necessária para o armazenamento dos dados na implementação do sistema gradiente (6.14). Apesar dos produtos em (6.23) serem executados em cada iteração, a quantidade de operações necessárias é consideravelmente reduzida, pois o somatório no segundo membro de (6.23) é composto por no máximo d parcelas, que em geral é um número pequeno.

6.7 Paralelização do sistema gradiente

O sistema gradiente (6.14) tem uma estrutura muito simples e é facilmente paralelizável. Do mesmo modo que os sistemas do capítulo 5, a paralelização é feita através da distribuição das equações do sistema gradiente (6.14) entre os diversos processadores de uma máquina paralela.

A avaliação do segundo membro do sistema envolve duas etapas, e cada uma delas pode ser realizada em paralelo:

1. Cálculo do vetor de resíduos $\mathbf{r} = \mathbf{A}\mathbf{f} - \mathbf{b}$;

2. Cálculo do produto da matriz $\mathbf{A}$ pela não-linearidade $\text{sgn}(\mathbf{r})$.

A implementação numérica do sistema gradiente (6.14) é feita através da integração em paralelo deste sistema. A forma de paralelização adotada enquadra-se, segundo a classificação dada por Petcu [90], na categoria de *paralelização através do sistema*. Neste caso, a avaliação do segundo membro do sistema gradiente (6.13) é feita em paralelo, o que resulta na distribuição das equações que compõem o sistema entre os processadores solicitados.

Um dos principais problemas a serem enfrentados na implementação numérica do sistema gradiente (6.14) é o armazenamento da matriz $\mathbf{A}$, que possui dimensões muito elevadas. Para uma imagem de 1024 por 1024 pontos, a matriz $\mathbf{A}$ correspondente é quadrada e tem dimensão $1.048.576 \times 1.048.576$. Considerando que a precisão dupla é usada nas imple-

mentações, o armazenamento desta matriz exige 8 terabytes de memória, o que inviabiliza a sua criação explícita.

No entanto, a matriz $\mathbf{A}$ é completamente determinada pela PSF e pelo operador de suavização que compõe a matriz $\mathbf{D}$, que são de dimensões pequenas, permitindo que o problema do armazenamento da matriz $\mathbf{A}$ seja resolvido, em paralelo, através do procedimento descrito na seção anterior, em que apenas os blocos $\mathbf{H}_i$ e $\mathbf{D}_i$ são armazenados, utilizando a forma compacta (6.24). Ao invés de criar explicitamente a matriz $\mathbf{A}$, os blocos $\mathbf{H}_i$ e $\mathbf{D}_i$ são utilizados diretamente nos cálculos.

6.7.1 Particionamento do problema

A paralelização do sistema gradiente (6.14) exige a definição do tamanho de cada partição e, em seguida, o mapeamento das partições criadas nos processadores disponíveis. Seja $k = 0, 1, \ldots, p-1$, a identificação de cada unidade de processamento e p o número de processadores solicitados na máquina paralela. O vetor $\mathbf{f}$ é de ordem MN e a matriz $\mathbf{A}$ é de ordem $MN \times MN$, sendo que M é a altura e N a largura da imagem.

Para que a estrutura das matrizes bloco-Toeplitz $\mathbf{H}$ e $\mathbf{D}$ possa ser explorada adequadamente através do procedimento descrito na seção 6.6, o particionamento do sistema gradiente (6.14) é obtido a partir do particionamento do vetor $\mathbf{f}$. O vetor $\mathbf{f}$ é a estimativa da imagem original organizada lexicograficamente, que é particionado de tal maneira que cada partição, mapeada em um processador diferente, corresponde a um determinado número de linhas da imagem a ser obtida. Este particionamento é ilustrado na figura 6.4, considerando um particionamento para três processadores.

A dimensão de cada partição é determinada através do quociente entre o número de linhas M da imagem e o número de processadores p solicitados para o processamento paralelo. Denotando por $M \backslash p$ a divisão inteira de M por p, a dimensão de cada partição do vetor $\mathbf{f}$ é dada por:

$$P_{img} = \begin{cases} M \backslash p, \text{ se } k \neq 0 \\ M \backslash p + M \bmod p, \text{ caso contrário} \end{cases} \tag{6.25}$$

Caso o quociente M/p tenha resto igual a zero, as dimensões das partições, definidas em (6.25), garantem uma distribuição uniforme da carga de processamento na máquina paralela. Caso contrário, a dimensão da partição mapeada no processador identificado por $k = 0$,

Figura 6.4: Particionamento da imagem para processamento paralelo através do sistema gradiente (6.14) utilizando três processadores, em que cada bloco é mapeado em um processador diferente

é maior que a dimensão das partições mapeadas nos demais processadores; neste caso o processador $k = 0$ recebe uma carga maior de processamento que os demais. Considerando a organização lexicográfica da imagem $\mathbf{f}$, cada bloco-linha da matriz $\mathbf{A}$ possui N linhas e, portanto, a cada partição da imagem correspondem $P_{img}N$ linhas na matriz $\mathbf{A}$.

6.7.2 Mapeamento das partições nos processadores

Sejam as partições da matriz $\mathbf{A}$ e dos vetores $\mathbf{f}$ e $\mathbf{r}$, mapeadas no k-ésimo processador, definidas como segue:

$$\mathbf{A}_l^{(k)} \rightarrow \text{ partição por linhas da matriz } \mathbf{A};$$
$$\mathbf{f}^{(k)} \rightarrow \text{ partição do vetor } \mathbf{f};$$
$$\mathbf{r}^{(k)} \rightarrow \text{ partição do vetor } \mathbf{r};$$
$$\mathbf{b}^{(k)} \rightarrow \text{ partição do vetor } \mathbf{b}.$$

A partição por linhas $\mathbf{A}_l^{(k)}$ é uma matriz que contém as linhas de $\mathbf{A}$ que são mapeadas no k-ésimo processador. Utilizando este particionamento, o cálculo do vetor de resíduos é feito pelo k-ésimo processador através da fórmula:

$$\mathbf{r}^{(k)} = \mathbf{A}_l^{(k)}\mathbf{f} - \mathbf{b}^{(k)},\ k = 0, \ldots, p-1. \tag{6.26}$$

O cálculo do resíduo é efetuado em paralelo e o produto da matriz $\mathbf{A}_l^{(k)}$ pelo vetor $\mathbf{f}$ é efetuado utilizando as fórmulas dadas na seção 6.6. No cálculo do resíduo $\mathbf{r}$, a operação que demanda maior tempo de computação é o produto da matriz $\mathbf{A}$ pelo vetor $\mathbf{f}$. O particionamento descrito acima é justificado, pois o número de operações aritméticas por iteração é distribuído entre os processadores, reduzindo consideravelmente o o tempo de computação.

Após o cálculo das partições $\mathbf{r}^{(k)}$, é feita a sincronização dos processos, em que cada cada processador envia aos demais a partição computada localmente. Estes dados são utilizados para compor em cada processador o vetor de resíduos $\mathbf{r}$. A k-ésima partição do sistema de equações (6.14), mapeada no k-ésimo processador é dada por:

$$\dot{\mathbf{f}}^{(k)} = -\mathbf{M}^{(k)}\mathbf{A}_l^{(k)}\,\mathrm{sgn}(\mathbf{r}). \tag{6.27}$$

O k-ésimo processador integra numericamente a partição correspondente do sistema gradiente (6.14), dada por (6.27). A cada iteração, é necessário que os resultados parciais calculados localmente por cada processador sejam enviados aos demais processadores envolvidos.

O número de operações aritméticas necessárias para completar cada iteração é dividido entre os processadores solicitados para o processamento paralelo, proporcionando uma distribuição bastante uniforme da carga de processamento.

O algoritmo de controle de passo do método de Runge-Kutta é executado seqüencialmente no processador mestre, identificado como o processador zero. Os cálculos envolvidos são muito simples, envolvendo apenas grandezas escalares, por esta razão, não há vantagens em paralelizar este algoritmo. A paralelização deste algoritmo introduziria mais pontos de sincronização em cada iteração do integrador, e o tempo despendido em sincronização seria maior que o tempo despendido nos cálculos. A paralelização, neste caso, traria prejuízos ao desempenho total da implementação. Com a implementação serial do algoritmo de controle de passo, o único ponto de sincronização existente no cálculo do passo corresponde à comunicação do novo passo de integração pelo processador mestre aos demais processadores do grupo.

Por outro lado, o passo de integração do método de Euler, determinado através do método de Barzilai-Borwein, é calculado em paralelo. O cálculo do passo depende diretamente da solução computada nas duas últimas iterações, e do vetor gradiente computado na iteração anterior. Como estes valores estão distribuídos entre as unidades de processamento da

máquina paralela, o cálculo do passo do algoritmo 3.2 exige que todos os processos sejam sincronizados. O cálculo do passo representa um ponto de sincronização adicional ao algoritmo, mas os dados sendo transmitidos são apenas grandezas escalares de precisão dupla, que ocupam 8 bytes de memória, e a transmissão destes dados não representa uma carga adicional significativa ao algoritmo.

6.8 Exemplos numéricos

A implementação em paralelo do sistema gradiente (6.14) foi feita através da codificação dos algoritmos de integração em FORTRAN 90, utilizando as bibliotecas MPI, para a criação e gerenciamento dos processos paralelos, e as bibliotecas BLAS para a execução de operações envolvendo vetores e matrizes.

O aplicativo gerado pode ser usado tanto em máquinas de memória compartilhada como em máquinas de memória distribuída. Os experimentos apresentados a seguir foram realizados em uma máquina SGI Altix 350, cujas características são resumidas na tabela 1.2.

Para fins de comparação, o sistema de equações lineares (6.13) também foi resolvido através do método dos gradientes conjugados sem pré-condicionamento, descrito no algoritmo 6.1, e utilizando pré-condicionamento, descrito no algoritmo 6.2. A qualidade das imagens restauradas pelo sistema gradiente (6.14) também foi comparada com as imagens restauradas através dos filtros de Wiener e Lucy-Richardson [140].

Critério de parada

Nos exemplos a seguir, a integração numérica do sistema gradiente (6.14) é executada até que um ponto estacionário do sistema, dentro de uma determinada tolerância, seja atingido. Este critério é implementado através do erro normalizado cometido na função objetivo a cada iteração, dado por

$$\frac{|E(\mathbf{f}_{k+1}) - E(\mathbf{f}_k)|}{|E(\mathbf{f}_k)|} \leq 5 \times 10^{-4}, \tag{6.28}$$

em que $E(\mathbf{f})$ é definida em (3.52). Observe que, se

$$\frac{|E(\mathbf{f}_{k+1}) - E(\mathbf{f}_k)|}{|E(\mathbf{f}_k)|} = 0,$$

então $E(\mathbf{f}_{k+1}) = E(\mathbf{f}_k)$, o que implica que o sistema gradiente (6.14) atingiu um ponto estacionário, que é um ponto de mínimo de E, justificando o uso de (6.28) como critério de parada.

Freqüentemente utiliza-se como critério de parada o erro normalizado cometido na variável $\mathbf{f}$ a cada iteração [102], dado por

$$\frac{\|\mathbf{f}_{k+1} - \mathbf{f}_k\|}{\|\mathbf{f}_k\|} \leq tol. \tag{6.29}$$

No entanto, este critério mostrou-se inadequado ao método de Runge-Kutta, pois o algoritmo de controle de passo mantém o erro definido acima dentro de uma determinada tolerância, o que torna este critério inadequado para decidir o momento de parar o processo de integração.

Um critério de parada para algoritmos iterativos para restauração de sinais foi proposto por Walsh e Marcellin [141]. No entanto, segundo os próprios autores, este critério é inadequado quando a matriz de coeficientes do sistema de equações lineares que modela o processo de restauração não é formada explicitamente. Este é o caso do problema de restauração de imagens, que devido às elevadas dimensões desta matriz, em geral não é possível formá-la explicitamente.

Outro critério de parada para algoritmos iterativos destinados ao problema de restauração de sinais foi proposto por Trussell [142]. No entanto, para o sistema gradiente (6.14), este critério mostrou-se inadequado, pois utiliza diretamente a equação (6.3). Como a matriz $\mathbf{H}$ é mal condicionada, a taxa de convergência é consideravelmente lenta. Experimentos realizados no exemplo 6.8.1 abaixo mostram que o resíduo definido por $\|\mathbf{Hf} - \mathbf{g}\|^2$ decai a uma taxa de cerca de 0.1% a cada vinte interações, o que inviabiliza a utilização deste critério de parada.

Formas de degradação das imagens

As imagens são degradadas através de diferentes formas de embaçamento e por um ruído aditivo. A matriz $\mathbf{D}$ em (6.14) usualmente corresponde a um filtro passa-alta, e nos experimentos que seguem utilizamos o operador laplaciano, definido pela equação (6.30).

$$\nabla^2 = \frac{\partial^2}{\partial i^2} + \frac{\partial^2}{\partial j^2}. \tag{6.30}$$

O operador laplaciano é um operador derivativo e uma aproximação discreta para este operador, utilizada nos exemplos, é dada pela matriz em (6.31). O método para obter uma aproximação discreta para o operador de Laplace é descrito em [120].

$$\mathbf{L} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \tag{6.31}$$

O operador laplaciano é comumente utilizado na literatura [104]. A aplicação do operador de Laplace à imagem, resulta na atenuação de variações suaves de níveis de cinza e acentua variações bruscas ou descontinuidades nos níveis de cinza, tendendo a produzir imagens com contornos e outras descontinuidades, tais como ruído, mais acentuados [120, pg. 129]. Por outro lado, através do ajuste do parâmetro λ, é possível obter uma imagem restaurada com menor nível de ruído, porém menos nítida.

A escolha do parâmetro de regularização $\boldsymbol{\lambda}$

A escolha do parâmetro λ é importante, pois se λ é grande, o ruído será suprimido da imagem restaurada, porém a imagem resultante perde nitidez. Por outro lado, se λ for pequeno, a imagem resultante é mais nítida, mas o ruído, não é suprimido de forma eficiente.

Diversas formas de ajustar este parâmetro estão disponíveis na literatura. Galatsanos e Katsaggelos [143] propuseram dois métodos para estimar a variância do ruído presente na imagem e escolher o parâmetro λ. O problema de escolher o parâmetro de regularização λ também foi abordado por Miller [122].

Para ilustrar a influência da escolha deste parâmetro nos resultados obtidos, as simulações foram feitas utilizando diferentes valores constantes do parâmetro de regularização, e também utilizando a escolha adaptativa do parâmetro λ, proposta por Kang e Katsaggelos [144], em que este parâmetro é considerado como um funcional de $\mathbf{f}$ e é determinado iterativamente, baseado nos dados disponíveis em cada iteração do algoritmo de restauração utilizado.

Kang e Katsaggelos [144] obtiveram duas fórmulas para a atualização iterativa de λ: uma é linear; a outra quadrática. Realizamos experimentos preliminares com ambas as fórmulas, e a quadrática apresentou um desempenho superior à linear. Por esta razão, realizamos dois

experimentos com a fórmula quadrática, dada por

$$\lambda(\mathbf{f}) = \gamma\|\mathbf{Df}\|^2 - 1 + \sqrt{(1-\gamma\|\mathbf{Df}\|^2)^2 + 2\gamma\|\mathbf{Hf}-\mathbf{g}\|^2}, \tag{6.32}$$

sendo que $\gamma = 3/4\|\mathbf{g}\|^2$ garante a convexidade da função de degradação (6.11) para cada λ [144].

Medidas de qualidade das imagens restauradas

Em ambientes de simulação, onde a imagem original está disponível, a qualidade final das restaurações obtidas comumente é medida, em decibéis (db), através da relação sinal-ruído (RSR), dada por

$$\text{RSR} = 10 \times \log_{10}\left(\frac{\|\hat{\mathbf{f}}\|^2}{\|\hat{\mathbf{f}} - \mathbf{f}^*\|^2}\right), \tag{6.33}$$

sendo que $\hat{\mathbf{f}}$ é a imagem original sem degradações ou imagem de referência, e $\mathbf{f}^*$ é a imagem obtida pelo algoritmo de restauração. A relação sinal ruído mede a qualidade da imagem final obtida após encerrado o processo de restauração.

A outra medida de qualidade utilizada é a variação na relação sinal-ruído (ΔRSR), também em decibéis, que mede a diferença entre as relações sinal-ruído calculadas antes e depois da restauração, e é dada por:

$$\Delta\text{RSR} = 10 \times \log_{10}\left(\frac{\|\hat{\mathbf{f}} - \mathbf{g}\|^2}{\|\hat{\mathbf{f}} - \mathbf{f}^*\|^2}\right), \tag{6.34}$$

sendo que g é a imagem degradada.

A variação na RSR é largamente utilizada na literatura [105, 145], e é uma medida objetiva da melhora da qualidade da imagem restaurada $\mathbf{f}^*$, em relação à imagem degradada g. Um valor negativo deste índice indica que houve deterioração em relação à imagem degradada, e um valor positivo, indica que o método de restauração obteve uma estimativa da imagem original melhor que a imagem degradada. Quanto maior o valor da ΔRSR, melhor é a qualidade da imagem obtida em relação à imagem degradada.

A RSR e a variação na RSR freqüentemente não refletem a qualidade visual das imagens obtidas pelos algoritmos de restauração, porém estes índices servem como base para a comparação de diferentes técnicas de restauração de imagens [103].

Condições iniciais

Escolhemos como condição inicial a própria imagem degradada $\mathbf{g}$, isto é, $\mathbf{f}_0 = \mathbf{g}$. A opção pela imagem degradada como condição inicial é justificada pelo fato que a estimativa $\mathbf{f}$ da imagem original, determinada pelo sistema (6.14), é uma versão da imagem $\mathbf{g}$, com as degradações atenuadas, o que indica que $\mathbf{g}$ é o vetor conhecido mais próximo da solução procurada.

6.8.1 Imagem de 112×92 pontos

A fotografia mostrada na figura 6.5-(a) foi distorcida por um embaçamento por movimento de 9 pixels e 45 graus de ângulo, dado pela PSF em (6.35), e por um ruído branco gaussiano de variância igual a 10^{-3} e média zero.

(a) (b) (c)

Figura 6.5: Exemplo 6.8.1– (a) Imagem original de 112×92 pixels (Fonte: Olivetti Research Laboratory, Cambridge, UK); (b) Imagem degradada através de embaçamento por movimento com ângulo de 45 graus e 9 pixels; (c) Imagem degradada por embaçamento por movimento e ruído branco gaussiano de variância igual a 10^{-3}

A imagem original é mostrada na figura 6.5-(a). A imagem embaçada pela PSF (6.35) é mostrada em 6.5-(b) e, finalmente, na figura 6.5-(c), é mostrada a imagem obtida quando o ruído branco gaussiano é adicionado à imagem embaçada da figura 6.5-(b).

Neste exemplo, a dimensão da matriz $\mathbf{A}$, em (6.14), é 10304×10304 e, conseqüentemente, o sistema gradiente (6.14) possui 10304 equações e 10304 variáveis a serem calculadas.

$$\mathbf{P}_{mb} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.029203 & 0.075514 \\ 0 & 0 & 0 & 0 & 0.029203 & 0.099706 & 0.029203 \\ 0 & 0 & 0 & 0.029203 & 0.099706 & 0.029203 & 0 \\ 0 & 0 & 0.029203 & 0.099706 & 0.029203 & 0 & 0 \\ 0 & 0.029203 & 0.099706 & 0.029203 & 0 & 0 & 0 \\ 0.029203 & 0.099706 & 0.029203 & 0 & 0 & 0 & 0 \\ 0.075514 & 0.029203 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.35}$$

A restaurações feitas pelo sistema gradiente (6.14) são mostradas na figura 6.6, utilizando diferentes valores para o parâmetro de regularização λ. O sistema gradiente foi resolvido em paralelo através do método de Runge-Kutta de segunda ordem, descrito na seção 6.5, utilizando 1, 2, 4 e 8 processadores.

Os resultados obtidos para diferentes números de processadores são os mesmos obtidos pela implementação serial, que utiliza apenas um processador. O sistema gradiente foi implementado utilizando o operador de projeção 6.15 para limitar a saída do integrador e satisfazer a restrição de não negatividade sobre o vetor $\mathbf{f}$. A espessura da camada limite é $\varepsilon = 5 \times 10^{-2}$. A matriz $\mathbf{D}$ é uma matriz bloco-Toeplitz, formada utilizando o operador laplaciano dado em (6.30).

Observe que para valores pequenos do parâmetro de regularização, a imagem restaurada é mais nítida, entretanto o ruído não é suprimido adequadamente, como pode ser visto nas figuras 6.6-(a) a 6.6-(d). Por outro lado, à medida que λ é aumentado, a imagem restaurada torna-se mais suave e com menos ruído, havendo perda de nitidez no resultado final, como pode ser observado nas figuras 6.6-(e) e 6.6-(f). Quando o parâmetro de regularização é obtido ao longo das iterações do integrador, o ruído no resultado final é maior que utilizando $\lambda = 0.1$ e $\lambda = 2.0$, porém a imagem obtida é mais nítida. Estas observações são quantificadas através dos valores das RSRs, aumentam à medida que o valor do parâmetro de regularização também aumenta.

No entanto, os valores negativos das variações na RSR, mostrados nas tabelas 6.1 e 6.2 indicam que houve deterioração, em relação à imagem degradada, na imagem obtida pelo sistema gradiente (6.14). A exceção é para o caso em que $\lambda = 2.0$, obtendo $\Delta\text{RSR} =$

Figura 6.6: Exemplo 6.8.1. Restaurações obtidas pela implementação paralela do sistema gradiente (6.14) para diferentes valores de λ e utilizando 4 processadores– (a) $\lambda = 10^{-6}$, imagem restaurada com RSR $= 9.62$db e ΔRSR $= -8.48$db; (b) $\lambda = 10^{-4}$, imagem restaurada com RSR $= 9.51$db e ΔRSR $= -8.58$db; (c) $\lambda = 10^{-3}$, imagem restaurada com RSR $= 9.66$db e ΔRSR $= -8.43$db; (d) $\lambda = 10^{-2}$, imagem restaurada com RSR $= 10.75$db e ΔRSR $= -7.34$db; (e) $\lambda = 10^{-1}$, imagem restaurada com RSR $= 12.70$ db e ΔRSR $= -5.39$db; (f) $\lambda = 2.0$, imagem restaurada com RSR $= 18.94$db e ΔRSR $= 0.84$db ; (g) λ determinado iterativamente através da equação (6.32), imagem restaurada com $RSR = 10.54$db e ΔRSR $= -7.64$db

0.84db, indicando uma sensível melhora na qualidade imagem obtida, em relação à imagem degradada.

As restaurações feitas pelo método dos gradientes conjugados, descrito no algoritmo 6.1, e utilizando os mesmos valores para λ que o sistema gradiente (6.14), são mostradas na figura 6.7. O critério de parada utilizado para o método dos gradientes conjugados é dado em (6.29), com $tol = 5 \times 10^{-4}$.

Em comparação com as restaurações feitas utilizando método dos gradientes conjugados, mostradas na figura 6.7, para os mesmos valores de λ utilizados com o sistema gradiente (6.14), os valores da RSR e ΔRSR obtidos pelo sistema gradiente são maiores, em todos os experimentos realizados. Observe nas tabelas 6.1 e 6.2, que todos os valores das ΔRSR obtidos pelo método dos gradientes conjugados são negativos.

Tabela 6.1: Exemplo 6.8.1 – valores da relação sinal-ruído (RSR) e da variação na RSR (ΔRSR) das imagens restauradas pelo sistema gradiente (6.14) e pelo método dos gradientes conjugados, com valores de λ entre 10^{-6} e 10^{-3}

Método \ λ	$\lambda = 10^{-6}$	$\lambda = 10^{-4}$	$\lambda = 10^{-3}$	λ \ Índice
Sistema gradiente	9.62db -8.48db	9.51db -8.58db	9.66db -8.43db	RSR ΔRSR
Gradientes conjugados	2.16db -15.93db	4.00db -14.09db	6.94db -11.16db	RSR ΔRSR

Tabela 6.2: Exemplo 6.8.1 – valores da relação sinal-ruído (RSR) e da variação na RSR (ΔRSR) das imagens restauradas pelo método dos gradientes conjugados e pelo sistema gradiente (6.14), com valores de λ maiores que 10^{-3} e λ variável

Método \ λ	$\lambda = 10^{-2}$	$\lambda = 10^{-1}$	$\lambda = 2.0$	λ variável	λ \ Índice
Sistema gradiente	10.75db -7.34db	12.70db -5.39db	18.94db **0.84db**	10.54db -7.65db	RSR ΔRSR
Gradientes conjugados	9.77db -8.32db	11.40db -6.69db	12.03db -6.06db	11.90db -6.19db	RSR ΔRSR

Tabela 6.3: Exemplo 6.8.1 – valores da relação sinal-ruído (RSR) e da variação na RSR (ΔRSR) das imagens restauradas pelos filtros de Wiener e Lucy-Richardson

Índice \ Método	Filtro de Wiener	Filtro de Lucy-Richardson
RSR ΔRSR	13.48db -4.61db	15.43db -2.67db

Observe também, na tabela 6.2, que para $\lambda = 2.0$, o sistema gradiente obteve $\Delta\text{RSR} = 0.84$db, indicando que o resultado obtido é sensivelmente melhor que a imagem degradada, ao passo que, para o mesmo valor de λ, o método dos gradientes conjugados obteve $\Delta\text{RSR} = -6.06$db, indicando que houve deterioração em relação à imagem degradada.

Os valores maiores das RSRs obtidos pelo sistema gradiente (6.14), mostrados nas tabelas 6.1 e 6.2, indicam um resultado final de melhor qualidade, em relação ao resultado obtido pelo método dos gradientes conjugados.

A melhor qualidade das imagens restauradas pelo sistema gradiente (6.14) é devido ao fato que este sistema minimiza a norma L_1 do vetor de resíduos, ao passo que o método dos gradientes conjugados minimiza a norma L_2. Como discutido anteriormente, a solução em norma L_1 apresenta diversas vantagens em relação à solução de mínimos quadrados, como menor sensibilidade a ruídos e *outliers* [146]. Além disso, a solução em norma L_1 é a mais indicada quando a distribuição do ruído é pouco conhecida [125], que é o caso

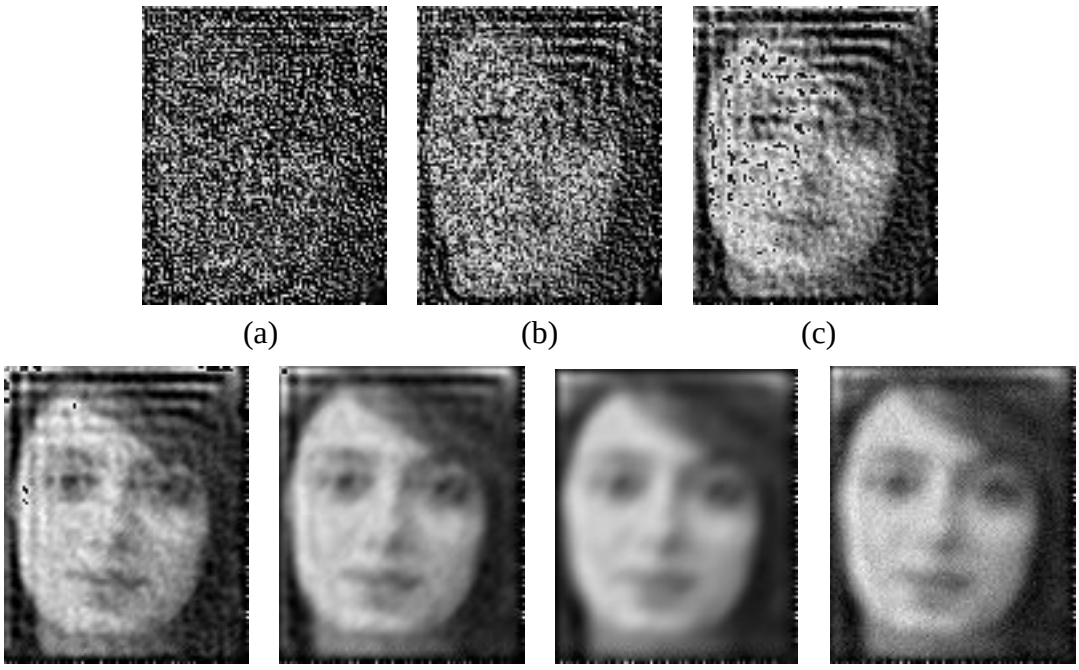


Figura 6.7: Exemplo 6.8.1. Restaurações obtidas pelo método dos gradientes conjugados– (a) $\lambda = 10^{-6}$, imagem restaurada com RSR $= 2.16$db e ΔRSR $= -15.93$db; (b) $\lambda = 10^{-4}$, imagem restaurada com RSR $= 4.00$ db e ΔRSR $= -14.095$db; (c) $\lambda = 10^{-3}$, imagem restaurada com RSR $= 6.94$db e ΔRSR $= -11.16$db; (d) $\lambda = 10^{-2}$, imagem restaurada com RSR $= 9.77$db e ΔRSR $= -8.32$db; (e) $\lambda = 0.1$, imagem restaurada com RSR $= 11.40$db e ΔRSR $= -6.69$db; (f) $\lambda = 2.0$, imagem restaurada com RSR $= 12.03$db e ΔRSR $= -6.06$db; (g) Com λ determinado iterativamente através da equação (6.32), imagem restaurada com RSR $= 11.90$db e ΔRSR $= -6.19$db

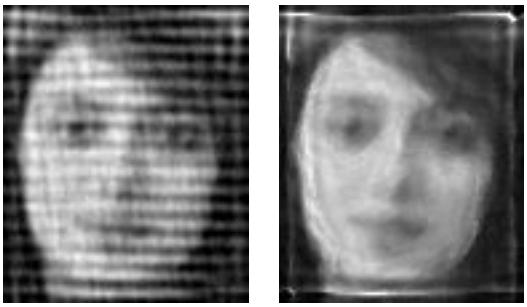
deste exemplo, pois nenhuma informação sobre a distribuição do ruído é utilizada no sistema gradiente (6.14) e no método dos gradientes conjugados.

Comparamos também as restaurações obtidas pelo sistema gradiente, mostradas na figura 6.6, com as obtidas utilizando o pacote de processamento de imagens do MATLAB, utilizando os filtros de Wiener e Lucy-Richardson. Na restauração através filtro de Wiener, foram utilizadas as seguintes informações:

- matriz de autocorrelação da imagem original e do ruído;
- espectros de energia da imagem original e do ruído.

No caso do filtro de Lucy-Richardson, a imagem degradada foi pré-processada por um filtro de mediana para a suavização do ruído. As figuras 6.8-(a) e 6.8-(b) mostram o resultado obtido pela restauração da imagem degradada (fig. 6.5-(c)), utilizando o filtro de Wiener e o

filtro de Lucy-Richardson, respectivamente.



(a) (b)

Figura 6.8: Exemplo 6.8.1– (a) Restauração obtida pelo filtro de Wiener, utilizando matrizes de autocorrelação da imagem e do ruído, $RSR = 13.48$db; (b) Restauração obtida com o filtro de Lucy-Richardson, em que a imagem degradada foi pré-processada por um filtro de mediana para suavização de ruído, $RSR = 15.43$db

Observe, na tabela 6.3, que os valores da variação na RSR são negativos, indicando que o resultado obtido apresenta qualidade inferior à imagem degradada. A qualidade dos resultados obtidos pelos filtros de Wiener e Lucy-Richardson é superior a dos obtidos pelo sistema gradiente (6.14) (fig. 6.6), quando $\lambda < 2.0$, pois apresentam valores da ΔRSR maiores. Entretanto, para $\lambda = 2.0$, o valor da $\Delta\text{RSR} = 0.84$db obtido pelo sistema gradiente, contra $\Delta\text{RSR} = -4.61$db e $\Delta\text{RSR} = -2.67$db, obtidos pelo filtro de Wiener e pelo filtro de Lucy-Richardson, respectivamente, mostram que a qualidade da restauração obtida pelo sistema gradiente é superior às obtidas por estes filtros.

Observe que, a melhor qualidade da imagem obtida pelo sistema gradiente (6.14) (fig. 6.6-(f)), em relação às obtidas pelos filtros de Wiener e Lucy-Richardson (fig. 6.8), foi consequida apenas aumentando o valor do parâmetro de regularização λ. A imagem degradada (fig. 6.5-(c)), submetida ao sistema gradiente, não sofreu qualquer tipo de pré-processamento, e nenhuma informação sobre a imagem original e o ruído foi utilizada. Por outro lado, informações estatísticas sobre a imagem original e o ruído foram utilizadas no filtro de Wiener, e a imagem degradada (fig. 6.5-(c)) teve o ruído atenuado por um filtro de mediana antes de ser submetida ao filtro de Lucy-Richardson.

Análise de desempenho

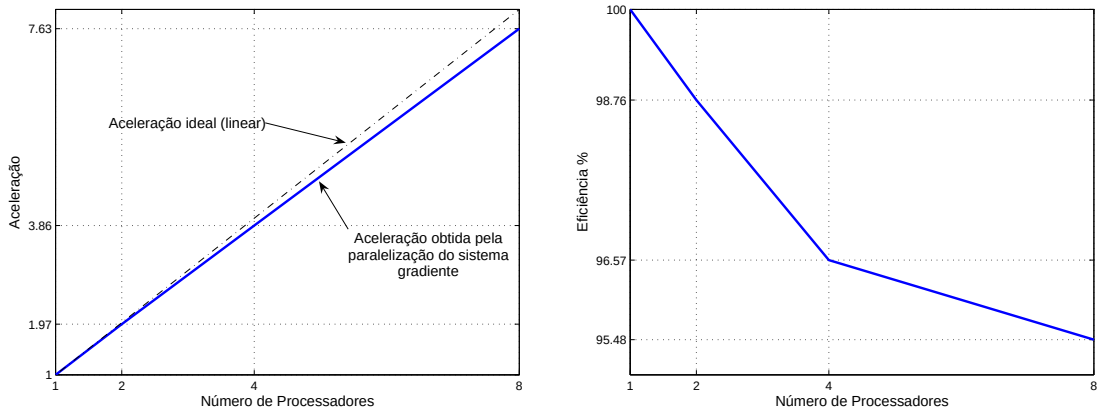
Para fins de análise de desempenho, consideramos o parâmetro de regularização $\lambda = 10^{-2}$. A tabela 6.4 mostra os tempos de processamento necessários para o sistema gradiente

(6.14) restaurar a imagem degradada, utilizando diferentes números de processadores. Observe que à medida que o número de processadores utilizados é aumentado, o tempo de processamento é reduzido consideravelmente. A aceleração e a eficiência do algoritmo paralelo em relação ao seqüencial são mostrados na tabela 6.4.

Tabela 6.4: Exemplo 6.8.1 – tempos de processamento, aceleração e eficiência em relação ao algoritmo serial da implementação paralela do sistema gradiente (6.14) com $\lambda = 10^{-2}$

Número de processadores	1 proc	2 procs	4 procs	8 procs
Tempo (min.)	195.09	98.76	50.50	25.54
Aceleração	1.00	1.97	3.86	7.63
Eficiência	100%	98.76 %	96.57 %	95.48 %

Os dados de aceleração e eficiência são exibidos na figura 6.9 em relação ao número de processadores. A figura 6.9-(a) mostra um crescimento aproximadamente linear da aceleração com o aumento do número de processadores.



(a) (b)

Figura 6.9: Exemplo 6.8.1. Curvas de desempenho — (a) Curva de aceleração; (b) Curva de eficiência

A aceleração quase linear apresentada na figura 6.9-(a) indica que o algoritmo paralelo não sofreu queda de desempenho considerável nos experimentos realizados. Entretanto, sabe-se que o desempenho de algoritmos paralelos sofre queda de desempenho a partir de um determinado número de processadores, devido ao aumento do tempo despendido em tarefas de comunicação e sincronização entre os processos.

Neste exemplo, a deterioração do desempenho é perceptível quando mais de 2 processadores são usados, e aumenta gradativamente com o aumento do número de processadores.

Este fato é observado na figura 6.9-(a), onde a medida que o número de processadores é aumentado, os valores da aceleração obtidos pelo sistema gradiente (6.14), estão mais distantes dos valores ideais.

A curva de eficiência é mostrada na figura 6.9-(b) e apresenta uma variação inferior a 5 pontos, o que indica que a fração do tempo total de processamento gasto em tarefas de sincronização e comunicação de processos é pequena. Estes dados mostram que a paralelização do sistema gradiente (6.14) proporcionou um ganho de desempenho considerável em relação ao algoritmo serial.

Comparando com o método dos gradientes conjugados, os tempos de processamento do sistema gradiente são significativamente maiores. Os tempos de processamento do método dos gradientes conjugados, em todos os experimentos deste exemplo, não excede 6 minutos. Entretanto, a qualidade das restaurações obtidas pelo método dos gradientes conjugados é, em todos os experimentos realizados, inferior à qualidade das imagens restauradas utilizando o sistema gradiente (6.14).

6.8.2 Imagem de 1024×1024 pontos

A imagem da figura 6.10-(a) foi distorcida por um embaçamento gaussiano de variância igual a 10.0, cuja PSF tem dimensão 20×20, e por ruído branco gaussiano de média zero e variância igual a 10^{-2}.

Neste exemplo, o sistema gradiente (6.14) possui $1.048.786$ equações e $1.048.786$ variáveis. A espessura da camada limite e o valor do parâmetro de regularização usados são respectivamente $\varepsilon = 10^{-1}$ e $\lambda = 10^{-2}$, sendo que a simulação é repetida aumentando valor do parâmetro de regularização para $\lambda = 2.0$.

O sistema gradiente (6.14) foi resolvido utilizando o método de Euler, com controle automático do passo pela fórmula de Barzilai-Borwein. O algoritmo 3.2 foi executado em paralelo utilizando 1, 2, 4 e 8 processadores.

O método dos gradientes conjugados foi implementado em paralelo neste exemplo. Apesar deste método ser de difícil paralelização, os produtos de matriz por vetor e os produtos internos presentes no algoritmo podem ser paralelizados, o que proporciona uma considerável redução no tempo de processamento. O particionamento e a paralelização destas operações, bem como a técnica de esparsidade aplicados ao método dos gradientes conjugados são os

(a)

(b)

(c)

Figura 6.10: Exemplo 6.8.2 — (a) Imagem original de 1024×1024 pixels; (b) Imagem distorcida por embaçamento gaussiano de variância igual a 10.0 e dimensão 20×20; (c) Imagem degradada por embaçamento gaussiano de variância igual a 10.0 e dimensão 20×20 e por ruído branco gaussiano de média zero e variância igual a 10^{-2}

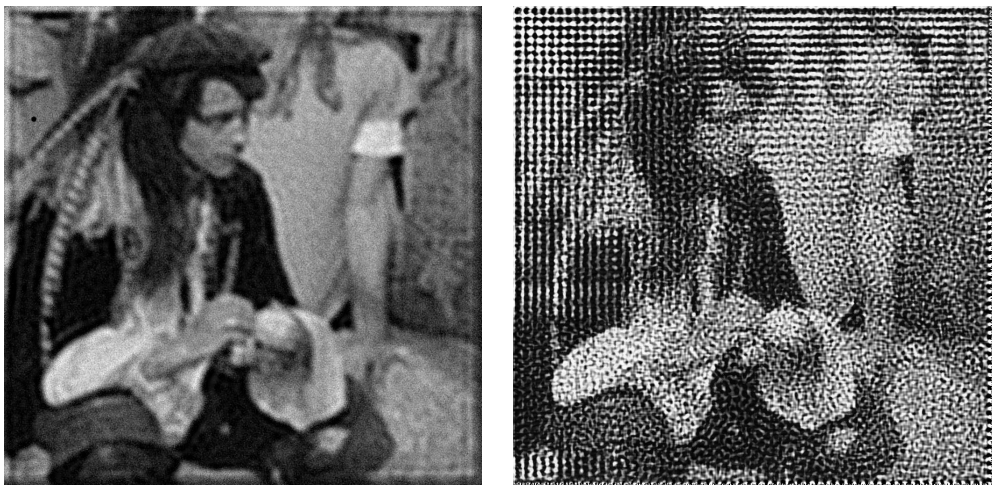
mesmos aplicados ao sistema gradiente (6.14).

Quando o parâmetro de regularização é aumentado para para $\lambda = 2.0$, o método dos gradientes conjugados pré-condicionado, descrito no algoritmo 6.2, também foi utilizado, sem paralelização, porém utilizando as mesmas técnicas para explorar a esparsidade das matrizes $\mathbf{H}$ e $\mathbf{D}$. O pré-condicionamento foi feito utilizando o pré-condicionardor circulante por blocos, descrito na seção 6.5.

Na figura 6.11-(a), é mostrado o resultado obtido pelo sistema gradiente (6.14), com $\lambda = 10^{-2}$ e utilizando quatro processadores. A relação sinal-ruído da imagem obtida é

RSR $= 8.90$db, e o valor da variação na RSR é ΔRSR $= -1.94$db, o que indica que houve deterioração, em relação à imagem degradada (fig. 6.10-(c)), no processo de restauração.

A imagem restaurada pelo método dos gradientes conjugados paralelizado, sem pré-condicionamento, com $\lambda = 10^{-2}$ e utilizando 4 processadores, é mostrada na figura 6.11-(b). Neste caso, o valor da relação sinal-ruído é RSR $= 3.16$db e a variação na RSR é ΔRSR $= -3.15$db, indicando que a imagem obtida apresenta qualidade inferior à imagem degradada (fig. 6.10-(c)).



(a) (b)

Figura 6.11: **Exemplo 6.8.2 (restaurações com $\lambda = 10^{-2}$)** — (a) Imagem restaurada pelo sistema gradiente (6.14), RSR = 8.80db; (b) Imagem restaurada pelo método dos gradientes conjugados, RSR =3.16db

Comparando os resultados obtidos pelo sistema gradiente (6.14) e pelo método dos gradientes conjugados, o maior valor da ΔRSR obtido pelo sistema gradiente, mostra que a deterioração ocorrida no processo de restauração é menor que a deterioração ocorrida no resultado obtido pela restauração através do método dos gradientes conjugados. Adicionalmente, o maior valor da relação sinal ruído obtido pelo sistema gradiente (6.14), mostra que a imagem obtida pelo sistema gradiente (6.14) apresenta qualidade superior à imagem obtida pelo método dos gradientes conjugados. Neste caso particular, isto pode ser confirmado através de inspeção visual. Os valores da RSR e da ΔRSR obtidos neste experimento estão resumidos na tabela 6.5.

Este fato também foi verificado no exemplo anterior, cuja razão é o fato de que o sistema gradiente (6.14) determina uma solução em norma L_1 mínima do sistema de equações linea-

Tabela 6.5: Exemplo 6.8.2 – valores da relação sinal-ruído (RSR) e da variação na RSR (ΔRSR) das imagens restauradas pelo sistema gradiente (6.14) e pelo método dos gradientes conjugados

Método \ λ	$\lambda = 10^{-2}$	$\lambda = 2.0$	Índice
Sistema gradiente	8.80db -1.94db	13.09db 2.35db	RSR ΔRSR
Gradientes conjugados	3.16db -3.15db	8.25db -2.49db	RSR ΔRSR

res (6.14). A solução em norma L_1 é menos sensível a ruídos e outliers que a solução de mínimos quadrados, determinada pelo método dos gradientes conjugados.

A imagem degradada (fig. 6.10-(c)) também foi restaurada utilizando os filtros de Wiener e Lucy-Richardson. Como no exemplo anterior, as matrizes de autocorrelação e os espectros de energia da imagem original do ruído foram utilizados no filtro de Wiener. No caso do filtro de Lucy-Richardson, a imagem degradada foi pré-processada por um filtro de mediana, para suavização do ruído.

A figura 6.12-(a) mostra a restauração obtida através do filtro de Wiener, obtendo uma imagem restaurada com relação sinal-ruído igual a 14.14db. A imagem mostrada na figura 6.12-(b) é o resultado da restauração através do filtro de Lucy-Richardson, que resultou em uma imagem restaurada com relação sinal-ruído igual a 13.02db.

Os valores da RSR e ΔRSR obtidos pelos filtros de Wiener e Lucy-Richardson, exibidos na tabela 6.6, mostram que os resultados obtidos pelo sistema gradiente, quando $\lambda = 10^{-2}$, apresentam qualidade inferior em relação aos obtidos por estes filtros. Os valores positivos da ΔRSR, obtidos pelos filtros de Wiener e Lucy-Richardson, mostram que as imagens obtidas apresentam melhor qualidade que a imagem degradada.

Quando o valor do parâmetro de regularização é aumentado para $\lambda = 2.0$, a restauração obtida pelo sistema gradiente, mostrada na figura 6.13-(a), também apresenta melhor qualidade que a obtida pelo método dos gradientes conjugados, mostrada na figura 6.13-(b). A imagem obtida pelo sistema gradiente (fig. 6.13-(a)) apresenta variação na RSR positiva, mostrada na tabela 6.5, ao passo que a imagem obtida pelo método dos gradientes conjugados (fig. 6.13-(b)) apresenta variação na RSR negativa, o que indica que houve deterioração em relação à imagem degradada (fig. 6.10-(c)).

Em relação aos filtros de Wiener e de Lucy-Richardson, para $\lambda = 2.0$, a imagem restau-

(a) (b)

Figura 6.12: Exemplo 6.8.2 — (a) **Restauração obtida pelo filtro de Wiener**, utilizando as matrizes de autocorrelação e os espectros de energia da imagem original e do ruído, RSR = 14.14db; (b) **Restauração obtida através do filtro de Lucy-Richardson**, antes utilizando um filtro de mediana para suavização do ruído na imagem degradada, RSR = 13.02db

rada obtida pelo sistema gradiente (6.14) apresenta $\Delta\text{RSR} = 2.35\text{db}$. Este valor é menor que a variação da RSR obtida pelo filtro de Wiener. Por outro lado, a qualidade da restauração obtida pelo sistema gradiente é superior à obtida pelo filtro de Lucy-Richardson, pois o valor da variação na RSR obtido pelo sistema gradiente é maior que o valor deste índice obtido pelo filtro de Lucy-Richardson, que é $\Delta\text{RSR} = 2.28\text{db}$.

Quando as matrizes de autocorrelação e os espectros de energia da imagem original e do ruído não são usados no filtro de Wiener, o valor da variação na RSR da imagem obtida, mostrada na figura 6.14-(a), é menor que os obtidos pelo sistema gradiente (6.14) e pelo método dos gradientes conjugados (tabela 6.6) para $\lambda = 10^{-2}$ e $\lambda = 2.0$ (tabelas 6.5 e 6.6).

No caso da restauração pelo filtro de Lucy-Richardson, quando a imagem degradada (fig. 6.10-(c)) não é pré-processada, a imagem restaurada, mostrada na figura 6.14-(b), apresenta variação da RSR menor que a obtida pelo sistema gradiente com $\lambda = 2.0$ (tabelas 6.5 e 6.6). O valor negativo deste índice indica que a imagem obtida apresenta qualidade inferior em relação à imagem degradada, ao passo que a variação na RSR da imagem restaurada pelo sistema gradiente (fig. 6.13-(a)), é positivo, indicando uma melhor qualidade em relação à imagem degradada.

Em relação aos filtros de Wiener e Lucy-Richardson, as soluções obtidas pelo sistema gradiente (6.14) apresentam melhor qualidade, dependendo apenas do ajuste adequado do

(a) (b)

Figura 6.13: **Exemplo 6.8.2 (restaurações com $\lambda = 2.0$)** — (a) Imagem restaurada pelo sistema gradiente (6.14), RSR = 13.09db; (b) Imagem restaurada pelo método dos gradientes conjugados com, RSR = 8.25db

parâmetro de regularização λ. Nenhum tipo de pré-processamento à imagem degradada, e nenhuma informação estatística sobre a imagem original e o ruído são necessários.

Tabela 6.6: Exemplo 6.8.2 – valores da relação sinal-ruído (RSR) e da variação na RSR (ΔRSR) das imagens restauradas pelos filtros de Wiener e Lucy-Richardson

Índice / Método	RSR	ΔRSR
Filtro de Wiener	$14.14^{(1)}$db 5.73db	$3.40^{(1)}$db -5.02db
Filtro de Lucy-Richardson	$13.02^{(2)}$db 10.16db	$2.28^{(2)}$db -0.56db

(1) dados obtidos utilizando as matrizes de autocorrelação e os espectros de energia da imagem original e do ruído
(2) imagem degradada foi pré-processada com um filtro de mediana

Análise de desempenho

Esta análise tem por objetivo estabelecer comparação entre o desempenho do sistema gradiente, resolvido através do método de Euler com passo adaptativo, e o método dos gradientes conjugados, utilizados para restaurar a imagem degradada da figura 6.10-(a), em termos de velocidade de processamento.

As comparações são efetuadas utilizando os tempos de processamento dos dois algorit-

(a) (b)

Figura 6.14: Exemplo 6.8.2 — (a) **Restauração obtida pelo filtro de Wiener**, sem as utilizar matrizes de autocorrelação e os espectros de energia da imagem original e do ruído, RSR = 5.73db; (b) **Restauração obtida através do filtro de Lucy-Richardson**, sem o pré-processamento da imagem degradada, RSR = 10.16db

mos e os conceitos de *aceleração* e *eficiência,* comumente usados na literatura como medidas de desempenho de algoritmos paralelos. Para este fim, os dois algoritmos considerados foram executados utilizando 1, 2, 4 e 8 processadores da máquina paralela.

Tempos de processamento – Os tempos de processamento necessários para obter as restaurações das imagem da figura 6.10-(b) são mostrados na tabela 6.7. Nota-se que os tempos de processamento do sistema gradiente (6.14) são consideravelmente menores que os do método dos gradientes conjugados. Além disso, o resultado obtido pelo sistema gradiente (6.14) apresenta qualidade superior ao obtido pelo método dos gradientes conjugados.

Este resultado ilustra uma vantagem do sistema gradiente (6.14), que é a flexibilidade na escolha de um método para obter uma solução numérica. No exemplo 6.8.1, utilizamos o método de Runge-Kutta de segunda ordem, obtendo restaurações de boa qualidade, porém com taxa de convergência reduzida. Neste exemplo, utilizamos o método de Euler com determinação de passo através do método de Barzilai-Borwein, o que proporcionou tempos de convergência significativamente menores que os tempos observados utilizando o método dos gradientes conjugados.

Quando o valor do parâmetro de regularização é aumentado para $\lambda = 2.0$, os tempos de processamento do sistema gradiente (6.14) e do método dos gradientes conjugados sofrem uma considerável redução em relação aos tempos obtidos com $\lambda = 10^{-2}$.

Tabela 6.7: Exemplo 6.8.2 – Tempos de processamento do método dos gradientes conjugados, sem pré-condicionamento, e do sistema gradiente (6.14), com $\lambda = 10^{-2}$

Método \ Num. procs.	1 proc	2 procs	4 procs	8 procs
Sistema gradiente	58h35m	45h09m	24h57m	16h24m
Gradientes conjugados	279h25m	117h29m	88h18m	65h27m

Tabela 6.8: Exemplo 6.8.2 – Tempo de processamento do método dos gradientes conjugados e do sistema gradiente (6.14), com $\lambda = 2.0$ e utilizando 1 processador

Método	Pré-condicionamento	Tempo de processamento
Sistema gradiente	não	5h52m
Gradientes conjugados	sim	7h07m
	não	29h07m

O método dos gradientes conjugados com pré-condicionamento (algoritmo 6.2) é utilizado, com $\lambda = 2.0$. O tempo de processamento é significativamente menor que o tempo de processamento do método dos gradientes conjugados sem pré-condicionamento (algoritmo 6.1), porém é maior que o tempo de processamento do sistema gradiente (6.14), que não utiliza pré-condicionamento. Os tempos de processamento observados neste experimento estão registrados na tabela 6.8.

Outros experimentos com valores de λ maiores foram realizados. Os tempos obtidos pelo sistema gradiente (6.14) são significativamente menores que os tempos de processamento obtidos pelo método dos gradientes conjugados em todos os experimentos realizados.

Aceleração e eficiência – o ganho obtido ao paralelizar o sistema gradiente (6.14) foi medido através da *aceleração* e da *eficiência*, definidos na página 186. Os resultados que seguem são relativos à simulação com $\lambda = 10^{-2}$.

A tabela 6.9 e a figura 6.15 mostram os valores da aceleração e da eficiência obtidos pela paralelização do sistema gradiente em relação ao algoritmo serial.

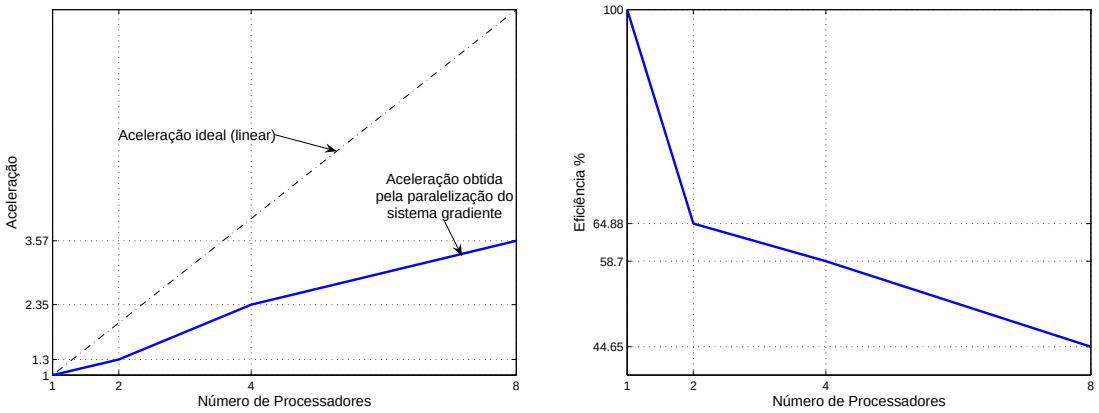
Na figura 6.15-(a) é mostrada a curva de aceleração, indicando que o ganho obtido pela paralelização do sistema gradiente em relação à implementação seqüencial é significativo. No entanto, os valores de aceleração estão muito abaixo dos valores ideais, que são os da aceleração linear.

A curva de eficiência é mostrada na figura 6.15-(b), e apresenta uma tendência de decréscimo acentuada à medida que o número de processadores é incrementado, o que indica que a fração de tempo utilizada em tarefas de sincronização e comunicação de processos aumenta rapidamente para este exemplo. Observe também que os valores obtidos da eficiên-

cia, variando de 64%, com dois processadores, até 44%, com 8 processadores, estão muito abaixo do valor ideal, que é de 100%.

Tabela 6.9: Exemplo 6.8.2 – Medidas de desempenho da implementação em paralelo do sistema gradiente (6.14), com $\lambda = 10^{-2}$, em relação ao sistema gradiente seqüencial

Num. processadores / Medida desempenho	1 proc	2 procs	4 procs	8 procs
Aceleração	1.00	1.30	2.35	3.57
Eficiência	100%	64.8 %	58.70 %	44.65 %



(a) (b)

Figura 6.15: Exemplo 6.8.2. Curvas de desempenho da paralelização do sistema gradiente (6.14), com $\lambda = 10^{-2}$, em relação ao sistema gradiente seqüencial — (a) Curva de aceleração; (b) Curva de eficiência

Os dados de aceleração e eficiência obtidos neste exemplo mostram que, apesar da paralelização do sistema gradiente ter proporcionado ganhos significativos de desempenho, a fração do tempo total de processamento, gasto em tarefas de sincronização e comunicação entre os processos paralelos é grande. A redução do tempo total gasto em tarefas de sincronização é necessária, afim de otimizar o uso dos recursos computacionais disponíveis. Isto proporcionará um aumento nos valores obtidos da aceleração e da eficiência.

Síntese do capítulo

Neste capítulo o problema de restauração de imagens degradadas por distorções lineares e invariantes no espaço foi apresentado e resolvido utilizando um sistema gradiente, que obtém uma solução em norma L_1 mínima do sistema de equações lineares que modela o problema.

O sistema gradiente foi resolvido através de duas técnicas numéricas distintas – um método de Runge-Kutta de segunda ordem com controle automático do passo através de um controlador PID, e o método de Euler com passo determinado através da fórmula de Barzilai-Borwein. Ambos os métodos produziram soluções numéricas com boa precisão. No entanto, o método de Euler com passo adaptativo mostrou-se mais adequado quando o sistema gradiente é de grande porte.

Os tempos de convergência obtidos com o método de Euler, com passo controlado pelo método de Barzilai-Borwein, são significativamente menores que os tempos de convergência obtidos com o método dos gradientes conjugados, além de obter soluções de melhor qualidade. Este fato, aliado à fácil paralelização dos sistemas gradientes, mostra que redes neurais formuladas matematicamente por sistemas gradientes são uma alternativa viável à resolução numérica de problemas de otimização de grande porte.

Capítulo 7

Conclusões e trabalhos futuros

Sistemas gradientes não-suaves para a resolução de problemas de otimização com restrições lineares foram propostos e analisados neste trabalho. Estes sistemas são vistos como modelos de redes neurais recorrentes com funções de ativação descontínuas, e são adequados a um grande número de aplicações, algumas apresentadas neste trabalho.

A análise dos sistemas gradientes propostos foi feita através da forma Persidskii dos sistemas e funções de Lyapunov do tipo diagonal e do tipo Lure-Persidskii. Um resultado geral de convergência para uma classe de sistemas Persidskii com segundo membro descontínuo é apresentada no capítulo 3. Este é um resultado chave, que possibilita a análise de convergência de uma ampla classe de sistemas gradientes aplicados à resolução de problemas de otimização.

Sistemas gradientes para resolução de diversas variantes de problemas de otimização foram apresentados no capítulo 3. Uma nova metodologia de análise de convergência, utilizando a forma Persidskii dos sistemas gradientes foi proposta, e sistemas gradientes que resolvem diversas variantes de problemas de otimização foram analisados.

No capítulo 4 um sistema gradiente foi formulado a partir de um de um problema de programação linear com variáveis limitadas para resolver o problema KWTA. Este problema de programação linear é resultado de uma modificação ao modelo proposto em [70], baseado em um problema de programação inteira. O sistema gradiente proposto no capítulo 4 apresenta vantagens importantes em relação ao circuito KWTA de Urahama e Nagao, como maior resolução e escalabilidade.

O treinamento de SVMs e LS-SVMs para o problema de classificação binária foi abor-

dado no capítulo 5. O treinamento de SVMs é feito através da resolução de um problema de programação quadrática com restrições lineares e consideramos duas variantes de SVMs: o C-SVM, que foi utilizado para separação linear de duas classes e o ν-SVM, que foi utilizado para a separação não linear. O LS-SVM é formulado através de um sistema de equações lineares quadrado, que foi resolvido pelo sistema gradiente destinado à resolução de sistemas de equações lineares, apresentado no capítulo 3.

No capítulo 6 o problema de restauração de imagens degradadas por distorções lineares e invariantes no espaço foi considerado. Este problema é formulado através um sistema de equações lineares mal-condicionado e, freqüentemente, de grande porte, que foi resolvido eficientemente utilizando o sistema gradiente para resolução de sistemas de equações lineares, apresentado no capítulo 3.

Contribuições

Neste trabalho, sistemas gradientes não-suaves obtidos a partir de um método de penalização exata são propostos e analisados.

Do ponto de vista teórico, uma nova metodologia de análise de convergência, baseada na forma Persidskii dos sistemas gradientes foi proposta. Para isso, foi provado um teorema geral de convergência para sistemas Persidskii com o segundo membro descontínuo.

Do ponto de vista de implementação, a possibilidade de paralelização destes sistemas foi amplamente explorada; foi mostrado que o desempenho dos sistemas gradientes é significativamente melhorado pela paralelização. Nas aplicações consideradas, devido à paralelização e/ou o uso de técnicas eficientes de integração e controle do passo, os sistemas gradientes propostos obtiveram um desempenho superior ao de métodos consagrados na literatura.

Nos tópicos abaixo, as contribuições e vantagens dos resultados obtidos neste trabalho são destacadas.

Sistemas Persidskii generalizados

No capítulo 3 foi proposto o sistema Persidskii generalizado, e um resultado de convergência global, estabelecido pelo teorema 3.2, foi provado para este sistema.

O teorema 3.2 generaliza o teorema de convergência absoluta apresentado em Persidskii

[19], que exige que as funções envolvidas sejam contínuas em todo o seu domínio. Este resultado simplifica consideravelmente as análises de convergência dos sistemas propostos neste trabalho, bastando para isso escrever os sistemas gradientes na forma Persidskii (3.12) e utilizar o teorema 3.2.

Análises de convergência

A resolução de problemas de programação não-linear utilizando sistemas dinâmicos com segundo membro descontínuo através de modos deslizantes foi abordada em Korovin e Utkin [147], onde os parâmetros de penalidade são matrizes diagonais, que são determinadas através do método do controle hierárquico, o que resolve o problema da determinação destes parâmetros. Este problema também foi abordado por Utkin [7], onde problemas de otimização convexa com restrições são resolvidos através de sistemas gradientes, obtidos utilizando um método de penalização exata. No entanto, a análise de convergência apresentada em [7] não fornece indicações sobre o ajuste dos parâmetros de penalidade.

Por outro lado, a técnica de análise utilizando a forma Persidskii dos sistemas gradientes apresenta as seguintes vantagens:

- Simplicidade das provas — os sistemas gradientes para resolução dos problemas propostos neste trabalho podem ser escritos, equivalentemente, numa forma Persidskii que satisfaz às condições do teorema 3.2;

- Ajuste de parâmetros— a convergência global é garantida para quaisquer valores positivos dos parâmetros de penalidade, ou resultam em condições explícitas em função apenas dos parâmetros do problema de otimização, como é o caso dos sistemas gradientes para a resolução de problemas de programação linear.

Sistema gradiente KWTA – problemas de programação linear

Os sistemas gradientes proposto para resolver o problema KWTA apresentam as seguintes propriedades:

- Escalabilidade — os sistemas são facilmente paralelizáveis, sendo adequados à resolução de problemas de grande porte;

- Implementação através de hardware — os sistemas gradientes podem ser implementados através de circuitos analógicos, utilizando apenas resistores, capacitores, amplificadores e switches;

- Menor complexidade — a complexidade dos sistemas gradientes apresentados é menor que a de diversos sistemas que resolvem a mesma classe de problemas, o que é uma vantagem em termos de implementação numérica, já que o número de operações aritméticas executadas a cada iteração é menor. No caso de implementação através de hardware, a quantidade de componentes necessária é menor, assim como o cabeamento necessário para conectar os componentes [12];

- Convergência global em tempo finito — depende apenas do ajuste adequado dos parâmetros de penalidade, e condições explícitas de convergência em tempo finito foram obtidas. A análise de convergência utilizando a forma Persidskii do sistema gradiente resultou em limites inferiores, que os parâmetros de penalidade devem satisfazer, dependentes apenas dos parâmetros do problema de programação linear que originou o sistema gradiente;

- Maior resolução, em comparação com o circuito de Urahama e Nagao [70].

Os ítens destacados acima também são aplicáveis aos sistemas gradientes para a resolução de três variantes de problemas de programação linear, analisados em [20] e [21], cujos resultados foram apresentados no capítulo 3.

Sistemas gradientes para treinamento de SVMs – programação quadrática

Os sistemas para treinamento de SVMs, propostos no capítulo 5 apresentam as seguintes vantagens:

- Convergência global — a análise através da forma Persidskii, utilizando o teorema 3.2, garante convergência global e independente do ajuste de parâmetros;

- Menor complexidade — os sistemas gradientes foram comparados com outras redes neurais que resolvem problemas de programação quadrática, e a análise mostra que os sistemas propostos apresentam menor complexidade, o que é uma vantagem sob o ponto de vista de implementação;

- Escalabilidade — os sistemas são facilmente paralelizáveis e a análise de desempenho apresentada na seção 5.5 mostra que os ganhos obtidos pela paralelização dos sistemas é considerável em relação às versões seqüenciais correspondentes e a outros dois algoritmos seqüenciais, considerados como referência, para treinamento de SVMs;

- Maior velocidade de processamento— os sistemas propostos apresentam melhor desempenho que algoritmos consagrados na literatura para treinamento de SVMs, desde que implementados com um algoritmo eficiente de integração e controle de passo e/ou sejam paralelizados;

- Desempenho de classificação equivalente ao de métodos já consagrados na literatura.

Os sistemas gradientes propostos para o treinamento de SVMs também são adequados à resolução de problemas gerais de programação quadrática com restrições lineares.

Sistema gradiente para sistemas de equações lineares – restauração de imagens

No capítulo 3 um sistema gradiente destinado à resolução de sistemas subdeterminados de equações lineares foi analisado. Este sistema determina uma solução em norma L_1 do sistema de equações lineares e apresenta as seguintes propriedades gerais:

- Convergência em tempo finito – este resultado foi obtido através da forma Persidskii do sistema gradiente e uma função de Lyapunov do tipo diagonal e independe do ajuste de parâmetros

- Menor sensibilidade a ruído – as soluções em norma L_1 são robustas a ruído e outliers;

- Escalabilidade – é facilmente paralelizável, sendo adequado à resolução de sistemas de equações lineares de grande porte.

Na aplicação de restauração de imagens:

- Nenhuma informação estatística sobre a imagem original e o ruído são necessárias;

- Qualidade das imagens restauradas: i) é superior à qualidade das imagens obtidas pelo filtro Lucy-Richardson, desde que o parâmetro de regularização seja ajustado

adequadamente; ii) supera qualidade da imagem obtida pelo filtro de Wiener, desde que este filtro não utilize informações estatísticas sobre a imagem original e o ruído; iii) é superior à qualidade das imagens obtidas pelo método dos gradientes conjugados em todos os experimentos;

- Desempenho – sem utilizar qualquer técnica de pré-condicionamento, o desempenho do sistema gradiente é superior ao do método dos gradientes conjugados com pré-condicionamento.

Conclusão

Neste trabalho, foi mostrado que sistemas gradientes obtidos a partir de um método de penalização exata são métodos competitivos, pois resolvem eficientemente problemas de otimização de grande porte. A velocidade de convergência depende do ajuste adequado de parâmetros, da técnica de integração usada e do método de controle do passo.

No aspecto teórico, a análise através da forma Persidskii é uma metodologia de análise de convergência aplicável a uma ampla classe de problemas de otimização, resultando em condições de convergência tratáveis.

Perspectiva de trabalhos futuros

Neste trabalho utilizamos sistemas gradientes obter a solução de diversas variantes de problemas de otimização, e mostramos que o uso de sistemas gradientes para a resolução de problemas de grande porte é viável. No entanto, podemos destacamos a seguir alguns ítens relacionados a este trabalho que podem ser objetos de maiores investigações.

Melhora do desempenho dos algoritmos paralelos

Apesar da implementação em paralelo dos sistemas gradientes apresentados ter proporcionado considerável melhora no desempenho, o muito tempo é gasto em tarefas de comunicação de processos. A redução na fração de tempo utilizada em tarefas de comunicação é necessária para um melhor desempenho da implementação.

Resolução de problemas de otimização não convexos com restrições não lineares

Os problemas de otimização tratados neste trabalho possuem restrições lineares, o que facilita a formulação das formas Persidskii dos sistemas gradientes correspondentes, facilitando a análise de convergência. A viabilidade das técnicas de análise utilizadas neste trabalho a sistemas dinâmicos destinados à resolução problemas de otimização com restrições não lineares exige maior investigação.

Utilização de outros métodos de integração numérica

Dois métodos de integração numérica foram utilizados para resolver os sistemas gradientes– um método de Runge-Kutta de segunda ordem e o método de Euler com passo selecionado através do método de Barzilai-Borwein, sendo que o segundo método exige tempo de computação consideravelmente menor que o primeiro. Outros métodos de integração podem ser usados, como métodos de integração implícitos, métodos de ordem variável e os métodos de Adams com passo variável.

Utilização de outras técnicas para eliminação ou atenuação do chattering

Neste trabalho, a técnica utilizada para eliminação do chattering foi a camada limite com espessura constante. Diversas técnicas para atenuação ou eliminação do chattering estão disponíveis na literatura, e podem ser aplicadas aos sistemas gradientes estudados neste trabalho, um exemplo é a camada limite com espessura variante no tempo.

Restauração de imagens coloridas

As imagens tratadas neste trabalho são imagens em preto e branco, pois o modelo matemático de restauração de imagens é adequado ao sistema gradiente (3.51). O processamento de imagens coloridas não foi considerado neste trabalho, pois o modelo matemático de representação destas imagens, em princípio, não é adequada aos modelos desenvolvidos neste trabalho.

Implementação física, utilizando circuitos VLSI

Os sistemas gradientes apresentados neste trabalho são adequados para implementação utilizando VLSI para processamento em tempo real. As implementações são potencial-

mente simples e baratas, pois podem ser feitas utilizando apenas resistores, amplificadores e switches [12].

Apêndice A

Conceitos básicos de computação paralela

Apresentamos a seguir uma breve introdução aos conceitos de processamento paralelo utilizados nos capítulos que seguem. Uma introdução mais detalhada à computação paralela é apresentada por Grama et al. [98]. Roosta [148] e Freeman e Phillips [149] apresentam abordagens mais direcionada a algoritmos paralelos.

Arquitetura de máquinas paralelas

Existem hoje diversas arquiteturas de máquinas paralelas e a classificação mais utilizada foi proposta por Flynn [150], conhecida como *taxonomia de Flynn*. Segundo Flynn, existem quatro categorias básicas de máquinas:

1. Single Instruction Stream (SISD) – é o modelo de von Neumann, nesta categoria encaixam-se os microcomputadores pessoais e máquinas com apenas um processador.

2. Single Instruction Stream - multiple data stream (SIMD) – inclui máquinas que suportam paralelismo em *arrays*, que consiste em máquinas com vários processadores conectados em forma de *grid*. Estas máquinas possuem uma unidade de controle que recebe e interpreta instruções, para em seguida enviá-las aos processadores conectados em grid para processamento paralelo.

3. Multiple instruction stream - single data stream (MISD) – poucas máquinas foram

construídas nesta categoria e nenhuma teve sucesso comercial ou algum impacto na computação científica.

4. Multiple instruction stream - multiple data stream (MIMD) – é a categoria mais abrangente, cobrindo máquinas com vários processadores e que suportam paralelismo de processos.

Em uma máquina com múltiplos processadores, cada processador é identificado por um número, iniciando em 0 e terminando em $p-1$, em que p é o número de processadores existentes na máquina. As máquinas que possuem múltiplos processadores pertencem, segundo a taxonomia de Flynn, à categoria MIMD. É esta classe de máquinas utilizadas neste trabalho, como exemplos pode-se citar o cluster Itautec e a SGI Altix 350, descritas nas tabelas 1.1 e 1.2. As máquinas de múltiplos processadores, de acordo com a forma de acesso à memória, podem ser classificadas de três formas

- *Sistemas de memória compartilhada* – a memória é global e acessível a todos os processadores. Esquematicamente, esta arquitetura é mostrada na figura A.1.

- *Sistemas de memória distribuída ou de memória local* – a memória é local, cada processador possui uma memória separada e não é acessível aos demais processadores. O esquema desta arquitetura é mostrado na figura A.2.

- *Sistemas híbridos* – possuem memória distribuída e compartilhada. Nestes sistemas, os processadores são divididos em grupos e, cada grupo de processadores possui seu espaço de memória, que não é acessível aos demais grupos.

Como exemplos de sistemas de memória compartilhada podemos citar o SGI Altix 350, o Cray SV1. Como exemplos de máquinas de memória distribuída, encontram-se o IBM SP2. Na categoria de sistemas híbridos, encontra-se o cluster Itautec, que 16 possui nós com dois processadores cada e memória de cada nó é compartilhada entre estes dois processadores.

Software

O paralelismo em máquinas paralelas é comumente gerenciado através de APIs (Application Programming Interfaces) específicas.

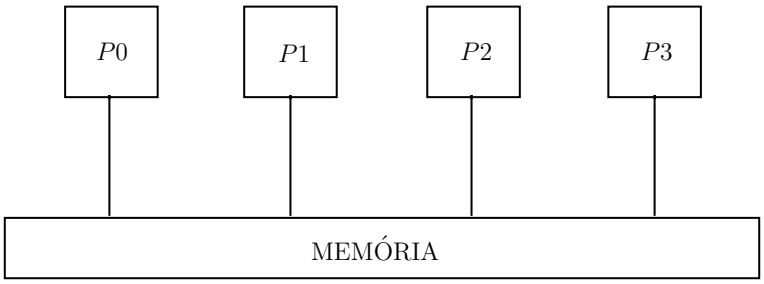


Figura A.1: Arquitetura de memória compartilhada – esquema de uma máquina com quatro processadores, representados pelos blocos $P0$ a $P4$, e um único barramento de memória, que é acessível por todos os processadores.

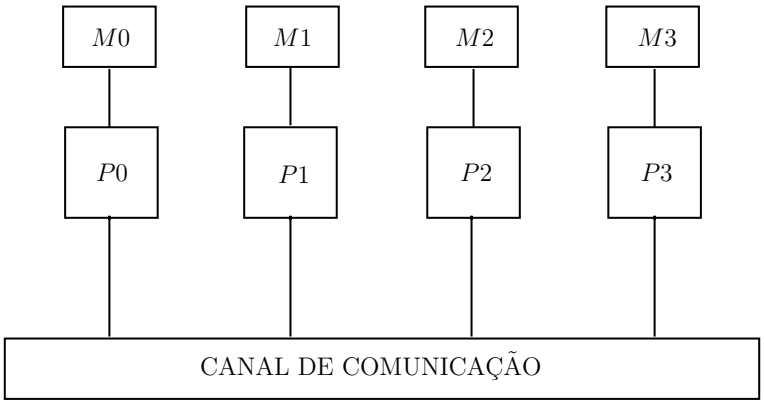


Figura A.2: Arquitetura de memória distribuída – esquema de uma máquina com quatro processadores, identificados por $P0$ a $P4$ e as memórias correspondentes, representadas por $M0$ a $M1$; a comunicação entre os processadores é feita pelo canal de comunicação, em geral uma rede TCP/IP.

Dependendo da arquitetura da máquina, o paralelismo pode ser obtido através da criação de processos ou *threads* paralelos, definidos como segue:

- Um *processo* consiste de um conjunto de instruções que executam uma tarefa, e não compartilham recursos alocados pelo sistema operacional, como o espaço de memória. Um processador é o dispositivo físico de hardware que executa os processos.

- *Threads* são partes de um processo, cada processo possui pelo menos um thread, que é chamado de thread mestre e diversos threads podem existir em um processo. Os threads podem compartilhar entre eles os recursos alocados pelo sistema operacional para o processo, como o espaço memória, o que os torna adequados à programação paralela em sistemas de memória compartilhada.

Em sistemas de memória distribuída, atualmente as APIs mais utilizadas são as de pas-

sagem de mensagem, como MPI (Message Passing Interface) e PVM (Parallel Virtual Machine). Estas APIs criam, em cada processador, processos que executam as tarefas de forma independente. A API de passagem de mensagens é responsável por gerenciar a comunicação entre os processos criados e também por finalizá-los.

Em sistemas de memória compartilhada, o paralelismo pode ser gerenciado através do OpenMP (Open Multi Processing), que consiste de um conjunto de especificações e interfaces para paralelizar um programa. O OpenMP funciona criando um único processo com vários *threads* paralelos, o número de threads é definido pelo usuário e é recomendável que seja igual ao número de processadores disponíveis, pois assim cada thread é executado por um processador, evitando a execução de mais de um thread por algum processador. O OpenMP é destinado exclusivamente a sistemas de memória compartilhada e a comunicação entre os threads é feita através da leitura direta da área de memória em que está armazenada a informação desejada. As APIs de passagem de mensagens também podem ser utilizadas em máquinas de memória compartilhada, o que torna os programas que as utilizam mais portáveis.

Medidas de desempenho

O objetivo do paralelismo é reduzir o tempo necessário para completar uma determinada tarefa computacional. O desempenho de um algoritmo paralelo é medido pela *aceleração* e pela *eficiência*, definidos a seguir.

Definição A.1. Seja t_p o tempo de execução de um programa paralelo em p processadores e t_s o tempo de execução do algoritmo serial mais rápido utilizando 1 processador, a *aceleração em p processadores* é definida pela relação

$$S_p = \frac{t_0}{t_p}$$

A aceleração, segundo a definição A.1, mede o ganho obtido em utilizar um programa paralelo ao invés de um programa serial executado em apenas um processador. Outra definição de aceleração é dada em função da versão seqüencial do algoritmo paralelo.

Definição A.2. Seja t_p o tempo de execução de um algoritmo paralelo em p processadores e

t_1 o tempo de execução do mesmo algoritmo em 1 processador, a *aceleração em p processadores* é definida por

$$S_p = \frac{t_1}{t_p}$$

De acordo com a definição A.2, a aceleração é uma medida do ganho obtido pela paralelização do algoritmo, e mede diretamente os efeitos provocados pela comunicação entre processos e sincronização. Obviamente, os processadores usados para medir a aceleração nas definições A.1 e A.2 têm que ser iguais.

Idealmente, a aceleração deve crescer linearmente, com $S_p = p$, no entanto isto raramente é verificado na prática, pois a medida que o número de processadores p é aumentado, aumenta também o tempo gasto com sincronização e comunicação entre os processos. Um gráfico típico de aceleração, em comparação com a aceleração ideal, é mostrado na figura A.3. Outra medida de desempenho associada à aceleração é a eficiência, definida a seguir.

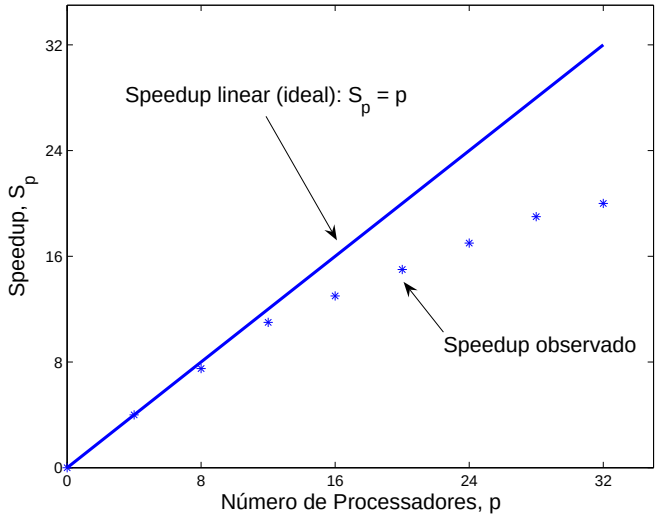


Figura A.3: Curva de speedup ideal (linha cheia) e um exemplo de uma curva de speedup usualmente observado na prática

Definição A.3 (Eficiência). A eficiência em p processadores é determinada pela relação

$$E_p = 100 \times \frac{S_p}{p}$$

A eficiência mede o percentual a fração de tempo em que os recursos totais de com-

putação recursos de computação que são utilizados, sem contar o tempo total gasto com comunicação e sincronização. Idealmente, quando a aceleração é linear, a eficiência é igual a 100% para todo p, o que raramente é verificado na prática. Em implementações reais, o que se verifica é um decréscimo da eficiência a medida que o número de processadores utilizados é incrementado, provocado pelo aumento tempo gasto em operações de sincronização e comunicação entre os processos. Uma curva de eficiência usualmente obtida na prática é mostrada na figura A.4. No desenvolvimento de algoritmos paralelos, deseja-se que a aceleração seja mantida o mais próximo possível a p e a eficiência o mais próximo possível a 100%.

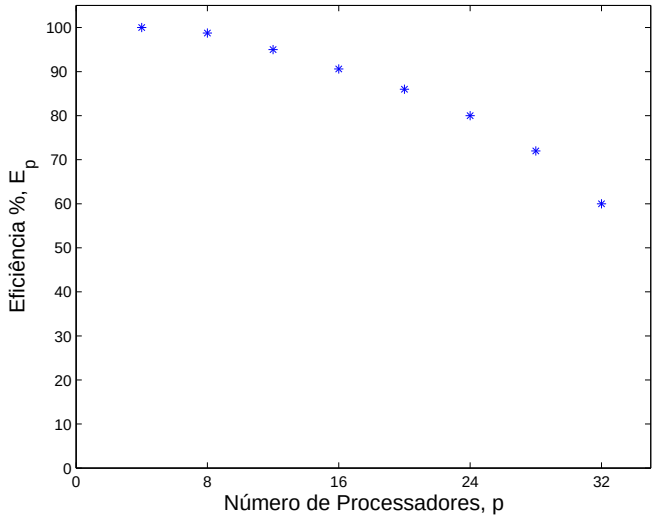


Figura A.4: Curva de eficiência usualmente obtida na implementação de algoritmos paralelos

As medidas de desempenho apresentadas acima são dadas em função do número de processadores utilizados no processamento e são os parâmetros que serão utilizados para avaliar o desempenho dos programas paralelos desenvolvidos neste trabalho.

Apêndice B

Software utilizado

A lista dos softwares usados no desenvolvimento das implementações dos algoritmos deste trabalho é apresentada abaixo. Todos são de código livre ou estão disponíveis gratuitamente na Internet, exceto os softwares proprietários da Silicon Graphics.

1. **Sistema operacional:** RedHat Enterprise Linux + SGI ProPack.

2. **Compilador:** Intel FORTRAN Compiler 9.0, disponibilizado pela própria Intel em `http://www.intel.com`. Versões não comerciais gratuitas estão disponíveis para a plataforma LINUX, e versões de teste, válidas por 30 dias, estão disponíveis para a plataforma Windows.

3. **Bibliotecas MPI:** Message Passing Toolkit (MPT), proprietário da Silicon Graphics. É compatível com as especificações 1.0, 1.1 e 1.2 do MPI, além de possuir alguns recursos do MPI-2. Uma alternativa de código livre ao MPT é o software MPICH 2, que é uma implementação das especificações MPI-1 e MPI-2, englobando as especificações implementadas pelo MPT. O MPICH 2 é fornecido gratuitamente em `http://www-unix.mcs.anl.gov/mpi/mpich2/`.

4. **Bibliotecas BLAS (Basic Linear Algebra Subroutines):** Goto BLAS, disponível gratuitamente em `http://www.tacc.utexas.edu/resources/software`, implementadas por Katsushike Goto. O BLAS foi usado para efetuar todas as operações envolvendo vetores e matrizes, maximizando o desempenho dos aplicativos, além de proporcionar uma melhor utilização de memória através do uso de subrotinas do Sparse BLAS, incorporadas no Goto BLAS.

5. **Cálculo da FFT:** FFTW 3.12 (Fastest Fourier Transform in the West), disponível gratuitamente em `http://www.fftw.org`. Utilizado para determinar a inversa do pré-condicionador circulante, usado no método dos gradientes conjugados.

Referências

[1] PYNE, I. B., “Linear programming on an electronic analogue computer”. *Trans. AIEE* v. 75, pp. 139–143, May 1956.

[2] RYBASHOV, M. V., “Gradient methods of solving linear and quadratic programming problems on electronic analog computers”. *Automation and Remote Control* v. 26, n. 12, pp. 2151–2162, Dec 1965.

[3] RYBASHOV, M. V., “The gradient method of solving convex programming problems on electronic analog computers”. *Automation and Remote Control* v. 26, n. 11, pp. 1886–1898, Nov 1965.

[4] RYBASHOV, M. V., “Stability of gradient systems”. *Automation and Remote Control* v. 9, pp. 1386–1393, 1975.

[5] KARPINSKAYA, N. N., “Method of "penalty" functions and the foundations of Pyne’s method”. *Automation and Remote Control* v. 28, n. 1, pp. 124–129, Jan 1967.

[6] LUENBERGER, D. G., *Linear and Nonlinear Programming*. 2nd ed., Reading, Addison-Wesley, 1984.

[7] UTKIN, V. I., *Sliding Modes in Control and Optimization*. Berlin, Springer-Verlag, 1992.

[8] KOROVIN, S. K., UTKIN, V. I., “Method of piecewise smooth penalty functions”. *Automation and Remote Control* v. 37, n. 1, pp. 94–105, Jan 1976.

[9] 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, n. 10, pp. 3088–3092, May 1984.

[10] TANK, D. W., HOPFIELD, J. J., “Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit”. *IEEE Transactions on Circuits and Systems* v. CAS-33, n. 5, pp. 533–541, May 1986.

[11] FORTI, M., NISTRI, P., “Global convergence of neural networks with discontinuous neuron activations”. *IEEE Transactions on Circuits and Systems* v. 50, n. 11, pp. 1421–1435, Nov 2003.

[12] CICHOCKI, A., UNBEHAUEN, R., *Neural Networks for Optimization and Signal Processing*. New York, John Wiley and Sons, 1993.

[13] RODRÍGUEZ-VÁZQUEZ, A., DOMÍNGUEZ-CASTRO, R., RUEDA, A., HUERTAS, J., SANCHEZ-SINENCIO, E., "Nonlinear switched-capacitor neural networks for optimization problems". *IEEE Transactions on Circuits and Systems* v. 37, n. 3, pp. 384–398, 1990.

[14] GLAZOS, M. P., HUI, S., ŻAK, S. H., "Sliding modes in solving convex programming problems". *SIAM Journal of Control and Optimization* v. 36, n. 2, pp. 680–697, Mar 1998.

[15] CHONG, E. K. P., HUI, S., ŻAK, S. H., "An analysis of a class of neural networks for solving linear programming problems". *IEEE Transactions on Automatic Control* v. 44, n. 11, pp. 1995–2006, 1999.

[16] KENNEDY, M. P., CHUA, L. O., "Neural networks for nonlinear programming". *IEEE Transactions on Circuits and Systems* v. 35, n. 5, pp. 554–562, May 1988.

[17] FORTI, M., NISTRI, P., QUINCAMPOIX, M., "Generalized neural network for nonsmooth nonlinear programming problems". *IEEE Transactions on Circuits and Systems* v. 51, n. 9, pp. 1741–1754, Sep 2004.

[18] FORTI, M., GRAZZINI, M., NISTRI, P., PANCIONI, L., "Generalized approach for convergence of neural networks with discontinuous or non-lipschitz activations". *Phisica D* v. 214, n. 1, pp. 88–99, Feb 2006.

[19] PERSIDSKII, S. K., "Problem of absolute stability". *Automation and Remote Control* v. 12, pp. 1889–1895, 1969.

[20] FERREIRA, L. V., *Utilização de redes neurais para a resolução de problemas de programação linear*. Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2002.

[21] FERREIRA, L. V., KASZKUREWICZ, E., BHAYA, A., "Convergence analysis of neural networks that solve linear programming problems". In: *Procedings of the International Joint Conference on Neural Networks*, v. 3, pp. 2476–2481, Honolulu, Hawaii, USA, May 12-17 2002.

[22] FERREIRA, L. V., KASZKUREWICZ, E., BHAYA, A., "Solving systems of linear equations using gradient systems with discontinuous righthand sides: application to LS-SVM". *IEEE Transactions on Neural Networks* v. 16, n. 2, pp. 501–505, Mar. 2005.

[23] ZHANG, X.-S., *Neural Networks in Optimization*. Dordrecht, Kluwer Academic Publishers, 2000.

[24] FERREIRA, L. V., KASZKUREWICZ, E., BHAYA, A., "Synthesis of a k-winners-take-all neural network using linear programming with bounded variables". In: *Procedings of the International Joint Conference on Neural Networks*, v. 3, pp. 2360–2365, Portland, OR, USA, July 20-24 2003.

[25] FERREIRA, L. V., KASZKUREWICZ, E., BHAYA, A., "Support vector classifiers via gradient systems with discontinuous righthand sides". In: *Procedings of the International Joint Conference on Neural Networks*, v. 4, pp. 2997–3002, Budapest, Hungary, July 25-29 2004.

[26] FERREIRA, L. V., KASZKUREWICZ, E., BHAYA, A. Support vector classifiers via gradient systems with discontinuous righthand sides. To appear in the Neural Networks Journal, 2006.

[27] STRANG, G., *Linear Algebra and Its Applications*. 3rd ed., New York, Academic Press, 1988.

[28] PERESSINI, A. L., SULLIVAN, F. E., UHL JR., J. J., *The Mathematics of Nonlinear Programming*. Berlin, Springer-Verlag, 1987.

[29] LANG, S., *Calculus of Several Variables*. 3rd ed., Berlin, Springer-Verlag, 1987.

[30] DEM'YANOV, V. F., VASIL'EV, L. V., *Nondifferentiable Optimization*. New York, Optimization Software, Inc., 1985.

[31] CLARKE, F. H., *Optimization and Nonsmooth Analysis*. New York, John Wiley and Sons, 1983.

[32] SMIRNOV, G. V., *Introduction to the Theory of Differential Inclusions*. Rhode Island, American Mathematical Society, 2002.

[33] KHALIL, H. K., *Nonlinear Systems*. New York, Macmillan Publishing Company, 1992.

[34] LASALLE, J. P., *The Stability of Dynamical Systems*. Philadelphia, SIAM, 1976.

[35] SHEVITZ, D., PADEN, B., "Lyapunov stability of nonsmooth systems". *IEEE Transactions on Automatic Control* v. 39, n. 9, pp. 1910–1914, Sep 1994.

[36] PADEN, B. E., SASTRY, S. S., "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators". *IEEE Transactions on Circuits and Systems* v. CAS-34, n. 1, pp. 73–82, Jan 1997.

[37] FILIPPOV, A. F., "Differential equations with discontinuous righthand sides". *American Mathematical Society Translations* v. 42, n. 2, pp. 191–231, 1964.

[38] FILIPPOV, A. F., *Differential Equations with Discontinuous Righthand Sides*. Dordrecht, Kluwer Academic Publishers, 1988.

[39] UTKIN, V. I., "Variable structure systems with sliding modes". *IEEE Transactions on Automatic Control* v. AC-22, n. 2, pp. 212–221, Apr 1977.

[40] DE CARLO, R. A., ŻAK, S. H., MATTHEWS, G. P., "Variable structure control of multivariable systems: A tutorial". *Proceedings of the IEEE* v. 76, n. 3, pp. 212–232, Mar 1988.

[41] EDWARDS, C., SPURGEON, S. K., *Sliding mode control: Theory and Applications*. London, Taylor & Francis, 1998.

[42] HUNG, J. H., HUNG, J. C., "Chatter reduction in variable structure control". In: *Proceedings of the 20th International Conference on Industrial Electronics, Control and Instrumentation, 1994*, v. 3, pp. 1914–1918, Sep 5-9 1994.

[43] UTKIN, V. I., GULDNER, J., SHI, J., *Sliding mode control in electromechanical systems*. Boca Ratton, CRC Press, 1999.

[44] FURUTA, K., "Sliding mode control of a discrete system". *Systems & Control Letters* v. 14, pp. 145–152, 1990.

[45] FURUTA, K., PAN, Y., "Variable structure control with sliding sector". *Automatica* v. 36, pp. 211–228, 2000.

[46] BANDYOPADHYAY, B., JANARDHANAN, S., *Discrete-time Sliding Mode Control*, v. 323, *Lecture Notes in Control and Information Sciences*. Berlin, Springer-Verlag, 2006.

[47] YOUNG, K. D., UTKIN, V. I., ÖZGÜNER, U., "A control engineer's guide to sliding mode control". *IEEE Transactions on Control Systems Technology* v. 7, n. 3, pp. 328–342, May 1999.

[48] BERTSEKAS, D. P., "Necessary and sufficient conditions for a penalty method to be exact". *Mathematical Programming* v. 9, n. 1, pp. 87–99, Aug 1975.

[49] XIA, Y., WANG, J., "A recurrent neural network for solving nonlinear convex programs subject to linear constraints". *IEEE Transactions on Neural Networks* v. 16, n. 2, pp. 379–385, Mar 2005.

[50] KASZKUREWICZ, E., BHAYA, A., *Matrix Diagonal Stability in Systems and Computation*. Boston, Birkhäuser, 2000.

[51] HSU, L., KASZKUREWICZ, E., BHAYA, A., "Matrix-theoretic conditions for the realizability of sliding manifolds". *Systems & Control Letters* v. 40, pp. 145–152, 2000.

[52] BHAYA, A., KASZKUREWICZ, E., *Control Perspectives on Numerical Algorithms And Matrix Problems*, v. 10, *Advances in Design and Control*. Philadelphia, SIAM, 2006.

[53] SPIVAK, M., *Calculus on Manifolds*. Reading, Addison-Wesley, 1965.

[54] ROYDEN, H. L., *Real Analysis*. 2nd ed., New York, Macmillan Publishing Company, 1968.

[55] BAZARAA, M. S., DAVIS, J. J., SHERALI, H. D., *Linear Programming and Network Flows*. 2nd ed., New York, John Wiley and Sons, 1990.

[56] MIGDALAS, A., TORALDO, G., KUMAR, V., "Nonlinear optimization and parallel computing". *Parallel Computing* v. 29, pp. 375–391, 2003.

[57] CICHOCKI, A., AMARI, S.-I., *Adaptive blind signal and image processing*. New York, John Wiley and Sons, 2002.

[58] BUTCHER, J. C., *Numerical Methods for Ordinary Differential Equations*. New York, John Wiley and Sons, 2003.

[59] BARZILAI, J., BORWEIN, J. M., "Two-point step size gradient methods". *IMA Journal of Numerical Analysis* v. 8, pp. 141–148, 1988.

[60] JACOBS, R. A., "Increased rates of convergence through learning rate adaptation". *Neural Networks* v. 1, n. 3, pp. 295–307, 1988.

[61] VRAHATIS, M. N., ANDROULAKIS, G. S., LAMBRINOS, J. N., MAGOULAS, G. D., "A class of gradient unconstrained minimization algorithms with adaptive stepsize". *Journal of Computation and Applied Mathematics* v. 114, pp. 367–386, 2000.

[62] DAI, Y.-H., ZHANG, H., "Adaptive two-point stepsize gradient algorithm". *Numerical Algorithms* v. 27, pp. 377–385, 2001.

[63] DAI, Y.-H., YUAN, Y.-X., "Alternate minimization gradient method". *IMA Journal of Numerical Analysis* v. 23, pp. 377–393, 2003.

[64] SHI, Z.-J., SHEN, J., "Step-size estimation for unconstrained optimization methods". *Computational & Applied Mathematics* v. 24, n. 3, pp. 399–416, 2005.

[65] CASH, J. R., KARP, A. H., "A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides". *ACM Transactions on Mathematical Software* v. 16, n. 3, pp. 201–222, Sep 1990.

[66] VALLI, A. M. P., CAREY, G. F., COUTINHO, A. L. G. A., "Control strategies for timestep selection in finite element simulation of incompressible flows of coupled reaction-convection-diffusion processes". *International Journal for Numerical Methods in Fluids* v. 47, pp. 201–231, Nov 2004.

[67] HAYKIN, S., *Neural Networks: A Comprehensive Foundation*. New York, Macmillan Publishing Company, 1994.

[68] KASKI, S., KOHONEN, T., "Winner-take-all networks for physiological models of competitive learning". *Neural Networks* v. 7, n. 6, pp. 973–984, 1994.

[69] LEMMON, M., VIJAYAKUMAR, B. V. K., "Competitive learning with generalized winner-take-all activation". *IEEE Trans. Neural Networks* v. 3, n. 2, pp. 167–175, Mar 1992.

[70] URAHAMA, K., NAGAO, T., "K-winners-take-all circuit with $O(N)$ complexity". *IEEE Transactions on Neural Networks* v. 6, n. 3, pp. 776–778, May 1995.

[71] MAJANI, E., ERLANSON, R., ABU-MOSTAFA, Y. *On the k-winners-take-all network*. In: *Advances in Neural Information Processing Systems*, v. 1, pp. 634–642. Morgan Kauffmann, 1989.

[72] CALVERT, B. D., MARINOV, C. A., "Another k-winners-take-all analog neural network". *IEEE Transactions on Neural Networks* v. 11, n. 4, pp. 829–838, Jul 2000.

[73] YEN, J., GUO, J., CHEN, H.-C., "A new k-winners-take all neural network and its array architecture". *IEEE Trans. Neural Networks* v. 9, pp. 901–912, Sep 1998.

[74] ŻAK, S. H., UPATISING, V., HUI, S., "Solving linear programming problems with neural networks: a comparative study". *IEEE Transactions on Neural Networks* v. 6, n. 1, pp. 94–104, Jan 1995.

[75] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., *Introduction to Algorithms*. Cambridge, The Mit Press, 1990.

[76] KENNEDY, M. P., CHUA, L. O., "Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of chua and lin". *IEEE Transactions on Circuits and Systems* v. CAS-34, n. 2, pp. 210–214, Feb 1987.

[77] CORTES, C., VAPNIK, V., "Support-vector networks". *Machine Learning* v. 20, n. 3, pp. 273–297, 1995.

[78] SUYKENS, J. A., GESTEL, T. V., BRABANTER, J. D., MOOR, B. D., VANDEWALLE, J., *Least Squares Support Vector Machines*. Singapore, World Scientific, 2002.

[79] SCHÖLKOPF, B., SMOLA, A. J., WILLIAMSON, R. C., BARTLETT, P. L., "New support vector algorithms". *Neural Computation* v. 12, pp. 1207–1245, 2000.

[80] ANGUITA, D., BONI, A., "Improved neural network for SVM learning". *IEEE Transactions on Neural Networks* v. 13, n. 5, pp. 1243–1244, Sep 2002.

[81] ANGUITA, D., RIDELLA, S., ROVETTA, S., "Circuital implementation of support vector machines". *Electronic Letters* v. 34, n. 16, pp. 1596–1597, Aug 1998.

[82] TAN, Y., XIA, Y., WANG, J., "Neural network realization of support vector methods for pattern classification". In: *Proceedings of the International Joint Conference on Neural Networks*, v. 6, pp. 411–415, Como, Italy, 24-27 July 2000.

[83] ANGUITA, D., BONI, A., RIDELLA, S., "A digital architecture for support vector machines: theory, algorithm and FPGA implementation". *IEEE Transactions on Neural Networks* v. 14, n. 5, pp. 993–1009, Sep 2003.

[84] CRISTIANINI, N., SHAWE-TAYLOR, J., *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, Cambridge University Press, 2000.

[85] SCHÖLKOPF, B., SMOLA, A., *Learning with Kernels*. Cambridge, The MIT Press, 2002.

[86] CHANG, C.-C., LIN, C.-J., "Training ν-support vector classifiers: Theory and algorithms". *Neural Computation* v. 13, n. 9, pp. 2119–2147, 2001.

[87] STEINWART, I., "On the optimal parameter choice for ν-support vector machines". *IEEE Transactions on Pattern Analysis and Machine Intelligence* v. 25, n. 10, pp. 1274–1284, Oct 2003.

[88] LEUNG, Y., CHEN, K.-Z., JIAO, Y.-C., GAO, X.-B., LEUNG, K. S., "A new gradient-based neural network for solving linear and quadratic programming problems". *IEEE Transactions on Neural Networks* v. 12, n. 5, pp. 1074–1083, Sep 2001.

[89] XIA, Y., WANG, J., "A one-layer recurrent neural network for support vector machine learning". *IEEE Transactions on Systems, Man and Cybernetics – Part B* v. 34, n. 2, pp. 1261–1269, Apr 2004.

[90] PETCU, D., *Parallelism in Solving Ordinary Differential Equations*. Mathematical Monographs 64. Tipografia Universitatii, 1998. URL `http://web.info.uvt.ro/~petcu/publicat.html`.

[91] BLAKE, C. L., MERZ, C. J. UCI Repository of Machine Learning Databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[92] BISHOP, C. M., *Neural Networks for Pattern Recognition*. Oxford, Oxford University Press, 1995.

[93] JOACHIMS, T., "Making large-scale SVM learning practical". In: Schölkopf, B., Burges, C., Smola, A. (eds), *Advances in Kernel Methods – Support Vector Learning*, chapter 11. Cambridge, MIT-Press, 1999.

[94] OSUNA, E., FREUND, R., GIROSI, F., "An improved algorithm for support vector machines". In: *Proceedings of the 1997 IEEE Workshop on Neural Networks for Signal Processing*, pp. 276–285, 24-26 Sept. 1997.

[95] PLATT, J. C., *Sequential minimal optimization: A fast algorithm for training support vector machines*. Microsoft Research, MSR-TR-98-14, 1998. URL `http://www.research.microsoft.com/~jplatt`.

[96] CHANG, C.-C., LIN, C.-J. LIBSVM : a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, 2001.

[97] FAN, R.-E., CHEN, P.-H., LIN, C.-J., "Working set selection using second order information for training support vector machines". *Journal of Machine Learning Research* v. 6, pp. 1889–1918, Dec 2005.

[98] GRAMA, A., GUPTA, A., KARYPIS, G., KUMAR, V., *Introduction to Parallel Computing*. 2nd ed., Reading, Addison-Wesley, 2003.

[99] ZANGHIRATI, G., ZANNI, L., "A parallel solver for large quadratic programs in training support vector machines". *Parallel Computing* v. 29, pp. 535–551, 2003.

[100] DE LEONE, R., "Parallel algorithm for support vector machines training and quadratic optimization problems". *Optimization Methods and Software* v. 20, n. 2-3, pp. 379–388, Apr-Jun 2005.

[101] DONG, J.-X., KRZYŻAK, A., SUEN, C. Y., "Fast SVM training algorithm with decomposition on very large data sets". *IEEE Transactions on Pattern Analysis and Machine Intelligence* v. 27, n. 4, pp. 603–618, Apr 2005.

[102] KATSAGGELOS, A. K., BIEMOND, J., SCHAFER, R. W., MERSEREAU, R. M., "A regularized iterative image restoration algorithm". *IEEE Transactions on Signal Processing* v. 39, n. 4, pp. 914–929, Apr 1991.

[103] BANHAM, M. B., KATSAGGELOS, A. K., "Digital image restoration". *IEEE Signal Processing Magazine* v. 14, n. 2, pp. 24–41, Mar 1997.

[104] ZHOU, Y.-T., CHELLAPPA, R., VAID, A., JENKINS, B. K., "Image restoration using a neural network". *IEEE Transactions on Acoustic, Speech and Signal Processing* v. 36, n. 7, pp. 1141–1151, Jul 1988.

[105] PAIK, J. K., KATSAGGELOS, A. K., "Image restoration using a modified Hopfield network". *IEEE Transactions on Image Processing* v. 1, n. 1, pp. 49–63, Jan 1992.

[106] FIGUEIREDO, M. A. T., LEITÃO, J. M. N., "Sequential and parallel image restoration: neural network implementations". *IEEE Transactions on Image Processing* v. 3, n. 6, pp. 789–801, Nov 1994.

[107] PERRY, S. W., GUAN, L., "A partitioned modified Hopfield neural network algorithm for real-time image restoration". *Real-Time Imaging* v. 2, pp. 215–224, 1996.

[108] SUN, Y., "Hopfield neural network based algorithms for image restoration and reconstruction – part I: algorithms and simulations". *IEEE Transactions on Signal Processing* v. 48, n. 7, pp. 2105–2118, Jul 2000.

[109] SUN, Y., "Hopfield neural network based algorithms for image restoration and reconstruction – part II: performance analysis". *IEEE Transactions on Signal Processing* v. 48, n. 7, pp. 2119–2131, Jul 2000.

[110] SUN, Y., YU, S.-Y., "An eliminating highest error criterion in Hopfield neural network for bilevel image restoration". In: *International Symposium in Information Theory and Applications*, v. 3, pp. 1409–1411, Singapore, 1992.

[111] ZENARI, N., ACHOUR, K., "Restoration method using a neural network". In: *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, pp. 122–124, Beirut, 25-29 June 2001.

[112] WU, Y.-D., CHEN, Y.-H., ZHANG, H. Y., "An improved algorithm form image restoration based on modified Hopfield network". In: *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, v. 8, pp. 4720–4723, Guangzhou, 18-21 August 2005.

[113] PERRY, S. W., *Adaptive Image Restoration: Perception Based Neural Network Models and Algorithms*. Ph.D. dissertation, School of Electrical and Information Engineering, University of Sidney, Sidney, Australia, 1999.

[114] YUBING, H., LENAN, W., "Image processing using a modified Hopfield network of continuous state change". *Signal Processing* v. 20, n. 5, pp. 431–435, Mar 2004. In Chinese.

[115] XIA, Y., LEUNG, H., BOSSÉ, E., "Neural data fusion algorithms based on a linearly constrained least square method". *IEEE Transactions on Neural Networks* v. 13, n. 2, pp. 320–329, Mar 2002.

[116] WANG, Y., WAHL, F. M., "Vector-entropy optimization-based neural-network approach to image reconstrution from projections". *IEEE Transactions on Neural Networks* v. 8, n. 5, pp. 1008–1014, Sep 1997.

[117] CICHOCKI, A., UNBEHAUEN, R., LENDL, M., WENZIERL, K., "Neural networks for linear inverse problems with incomplete data especially in applications to signal and image reconstruction". *Neurocomputing* v. 8, pp. 7–41, 1995.

[118] GOPAL, S. S., HEBERT, T. J., "Pre-reconstruction of SPECT projection images by a neural network". *IEEE Transactions on Nuclear Science* v. 41, n. 4, pp. 1620–1625, Aug 1994.

[119] PERRY, S. W., WONG, H.-S., GUAN, L., *Adaptive Image Processing: A Computational Intelligence Perspective*. Bellingham, Spie Press, 2002.

[120] GONZALEZ, R. C., WOODS, R. E., *Digital Image Processing*. 2nd ed., Upper Saddle River, Prentice-Hall,, 2002.

[121] ANDREWS, H. C., HUNT, B. R., *Digital Image Restoration*. Englewood Cliff, Prentice-Hall, 1977.

[122] MILLER, K., "Least squares methods for ill-posed problems with a prescribed bound". *SIAM Journal on Mathematical Analysis* v. 1, n. 1, pp. 52–74, Feb 1970.

[123] KATSAGGELOS, A. K., BIEMOND, J., SCHAFER, R. W., MERSEREAU, R. M., "A general formulation of constrained iterative restoration algorithms". In: *Proceedings of the 1985 IEEE International Conference on Acoustics, Speech and Signal Processing*, v. 10, pp. 700–703, Apr 1985.

[124] TIKHONOV, A. N., ARSENIN, V. Y., *Solutions of Ill-Posed Problems*. Washington, V. H. Winston & Sons, 1977.

[125] CICHOCKI, A., UNBEHAUEN, R., "Neural networks for solving systems of linear equations – Part II: Minimax and least absolute value problems". *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing* v. 39, n. 9, pp. 619–633, Sep 1992.

[126] XIA, Y., WANG, J., "A general projection neural network for solving monotone variational inequalities and related optimization problems". *IEEE Transactions on Neural Networks* v. 15, n. 2, pp. 318–328, Mar 2004.

[127] HESTENES, M. R., STIEFEL, E., "Methods of conjugate gradients for solving linear systems". *Journal of Research of the National Bureau of Standards* v. 49, n. 6, pp. 409–436, 1952.

[128] GOLUB, G. H., VAN LOAN, C. F., *Matrix Computations*. 2nd ed., Baltimore, The Johns Hopkins University Press, 1989.

[129] NAGY, J. G., PLEMMONS, R. J., "Iterative image restoration using approximate inverse preconditioning". *IEEE Transactions on Image Processing* v. 5, n. 7, pp. 1151–1162, Jul 1996.

[130] BIEMOND, J., LAGENDIJK, R. L., MERSEREAU, R. M., "Iterative methods for image deblurring". *Proceedings of the IEEE* v. 5, pp. 856–883, May 1990.

[131] HANKE, M. *Iterative regularization techniques in image restoration*. In: Colton, D., Engl, H. W., Louis, A. K., McLaughlin, J. R., Rundell, W. (eds), *Surveys on Solution Methods for Inverse Problems*, pp. 35–52. Springer-Verlag, Wien, 2000.

[132] NAGY, J. G., PALMER, K. M., "Steepest descent, CG, and iterative regularization for ill-posed problems". *BIT Numerical Mathematics* v. 43, pp. 1003–1017, Dec 2003.

[133] HANKE, M., HANSEN, P. C., "Regularization methods for large-scale problems". *Surveys on Mathematics for Industry* v. 3, n. 1, pp. 253–315, 1993.

[134] CHAN, R. H., NG, M. K., "Conjugate gradient methods for Toeplitz systems". *SIAM Review* v. 38, n. 3, pp. 427–482, Sep 1996.

[135] STRANG, G., "A proposal for Toeplitz matrix calculations". *Studies in Applied Mathematics* v. 74, n. 2, pp. 171–176, Apr 1986.

[136] CHAN, R. H., NG, M. K., PLEMMONS, R. J., "Generalization of Strang's preconditioner with applications to Toeplitz least squares problems". *Numerical Linear Algebra with Applications* v. 31, n. 1, pp. 45–64, Jan 1996.

[137] CHAN, R. H., NAGY, J. G., PLEMMONS, R. J., "FFT-based preconditioners for Toeplitz-block least squares problems". *SIAM Journal on Numerical Analysis* v. 30, n. 6, pp. 1740–1768, Dec 1993.

[138] CHAN, T. F., "An optimal circulant preconditioner for Toeplitz matrices". *SIAM Journal on Scientific and Statistical Computing* v. 9, n. 4, pp. 766–771, Jul 1988.

[139] CHAN, R., JIN, X. Q., "A family of block preconditioners for block systems". *SIAM Journal on Scientific and Statistical Computing* v. 13, n. 5, pp. 1218–1235, Sep 1992.

[140] LUCY, L. B., "An iterative technique for the rectification of observed distributions". *The Astronomical Journal* v. 79, n. 6, pp. 745–754, Jun 1974.

[141] WALSH, D. O., MARCELLIN, M. W., "Another stopping rule for iterative signal restoration". *IEEE Transactions on Signal Processing* v. 47, n. 11, pp. 3156–3159, Nov 1999.

[142] TRUSSELL, H. J., "Convergence criteria for iterative restoration methods". *IEEE Transactions on Acoustic, Speech and Signal Processing* v. ASSP-31, n. 1, pp. 129–136, Feb 1983.

[143] GALATSANOS, N. P., KATSAGGELOS, A. K., "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation". *IEEE Transactions on Image Processing* v. 1, n. 3, pp. 322–336, Jul 1992.

[144] KANG, M. G., KATSAGGELOS, A. K., "General choice of the regularization functional in regularized image restoration". *IEEE Transactions on Image Processing* v. 4, n. 5, pp. 594–602, May 1995.

[145] SUN, Y., LI, J.-G., YU, S.-Y., "Improvement on performance of modified Hopfield neural network for image restoration". *IEEE Transactions on Image Processing* v. 4, n. 5, pp. 688–692–41, May 1995.

[146] BLOOMFIELD, P., STEIGER, W. L., *Least Absolute Deviations: Theory, Applications and Algorithms*. Boston, Birkhäuser, 1983.

[147] KOROVIN, S. K., UTKIN, V. I., "Using sliding modes in static optimization and nonlinear programming". *Automatica* v. 10, n. 5, pp. 525–532, 1974.

[148] ROOSTA, S. H., *Parallel Processing and Parallel Algorithms*. Berlin, Springer-Verlag, 2000.

[149] FREEMAN, T. L., PHILLIPS, H. B., *Parallel Numerical Algorithms*. New York, Prentice-Hall, 1992.

[150] FLYNN, M. J., "Some computer organizations and their effectiveness". *IEEE Transactions on Computers* v. C-21, n. 9, pp. 948–960, 1972.