PROJETO DE ALGORITMOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO UTILIZANDO FUNÇÕES DE LIAPUNOV COM CONTROLE

Fernando Agustín Pazos

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 CIÊNCIA EM ENGENHARIA ELÉTRICA

Aprovada por:

Prof. Amit Bhaya, Ph.D.

Prof. Eugenius Kaszkurewicz, D.Sc.

Prof. Liu Hsu, Docteur d'Etat.

Prof. Gilberto Oliveira Corrêa, Ph.D.

Profa. Claudia Alejandra Sagastizábal, D. Habil

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2007

PAZOS, FERNANDO AGUSTÍN

Projeto de algoritmos para resolução de problemas de otimização utilizando funções de Liapunov com controle [Rio de Janeiro] 2007

XI, 285 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2007)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Funções de Liapunov de controle, Otimização, Sistemas dinâmicos de controle

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Em primeiro lugar, a minha mais sincera gratidão ao professor Amit Bhaya, por tudo o que me ensinou, pela sua paciência, seu incentivo permanente, e, principalmente pela amizade e afeto que me demonstrou ao longo destes anos de trabalho.

Queria agradecer também especialmente a meu amigo Christian Schaerer, porque me incentivou a realizar este doutorado. Sem ele provavelmente nunca teria nem começado.

Finalmente, a todos os professores e colegas da COPPE, pelo seu estímulo sempre alentador.

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

PROJETO DE ALGORITMOS PARA RESOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO UTILIZANDO FUNÇÕES DE LIAPUNOV COM CONTROLE

Fernando Agustín Pazos

Setembro de 2007

Orientador: Amit Bhaya

Programa: Engenharia Elétrica

Diversos problemas da ciência e da engenharia podem ser formulados como problemas de otimização. Quando a solução analítica destes problemas não é possível, existe o recurso de empregar métodos numéricos, ou algoritmos, para achar a solução. Muitos destes algoritmos podem ser interpretados como sistemas de controle em malha fechada. A teoria de controle oferece poderosas ferramentas de dedução e análise de algoritmos tanto discretos como contínuos, o que possibilita a dedução de novos algoritmos assim como uma nova perspectiva sobre aqueles já conhecidos na bibliografia.

A contribuição desta tese consiste na dedução de algoritmos como sistemas dinâmicos de controle para a resolução de diversos problemas de otimização. Essa dedução esta baseada em funções de Liapunov de controle e controle ótimo de Liapunov. Os problemas abordados são os de achar zeros de funções vetoriais não lineares, achar mínimos de funções escalares, o problema geral de otimização convexa e desigualdades variacionais.

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

DESIGN OF ALGORITHMS TO SOLVE OPTIMIZATION PROBLEMS USING CONTROL LIAPUNOV FUNCTIONS

Fernando Agustín Pazos

September 2007

Advisor: Amit Bhaya

Department: Electrical Engineering

Several problems of science and engineering can be formulated as optimization problems. When analytical solutions are not possible, numerical methods, or algorithms, are used to find the solution. Many algorithms can be interpreted as closed loop control systems. Control theory offers powerful tools for deduction and analysis of algorithms, either continuous or discrete ones, which make possible the derivation of new algorithms as well as a new perspective on existing ones.

The contribution of this thesis consists of the derivation of algorithms as dynamical control systems to solve several optimization problems. This approach is based on control Liapunov functions and Liapunov optimizing control. The problems solved are zero finding for nonlinear vector functions, minimization of scalar functions, the general problem of convex optimization and variational inequalities.

Sumário

Capítulo 1

Introdução

Diversos problemas comuns em muitas áreas da ciência e da engenharia podem ser formulados como problemas de otimização. Devido a esta abrangência, estes problemas foram amplamente estudados nas últimas décadas, e diversas ferramentas matemáticas foram desenvolvidas e utilizadas com o intuito de resolvé-los.

A diversidade dos problemas de otimização é muito ampla. Dentre os principais, podemos mencionar o de achar zeros de uma função vetorial não linear, o de achar mínimos de funções escalares, convexas ou não, o de achar mínimos de funções escalares dentro de uma determinada região do seu domínio, problemas de desigualdades variacionais, entre outros. Todos estes problemas foram abundantemente tratados na bibliografia referenciada. Entretanto, em geral, soluções são apresentadas e analisadas caso a caso, sem uma metodologia sistemática de dedução e análise de métodos e ferramentas matemáticas capazes de resolver estes problemas.

Por outro lado, no contexto de achar zeros de funções vetoriais, Bhaya e Kaszkurewicz (ver, principalmente, [13, 14]), foram os primeiros a apresentar uma metodologia baseada na teoria de controle, através da qual diversos algoritmos podem ser interpretados como sistemas dinâmicos de controle em malha fechada. Os autores aplicam a teoria de controle também para a resolução de outros problemas de otimização, como por exemplo programação linear.

Esta mesma metodologia pode ser aplicada no contexto de outros problemas de otimização. Desta forma, novos algoritmos destinados a solucionar tais problemas podem ser desenvolvidos, assim como esta teoria também apresenta poderosas ferramentas

de análise e síntese (ou projeto) dos algoritmos.

Pretende-se neste trabalho, portanto, utilizar as ferramentas de controle no desenvolvimento e análise de algoritmos destinados a resolver problemas de achar zeros de funções vetoriais e outros problemas de otimização a serem descritos a seguir.

Em alguns casos, os problemas mencionados podem ser resolvidos analiticamente; porém em muitos outros a resolução analítica não é possível. Para estes casos, empregam-se métodos numéricos, também chamados de algoritmos, que desenvolvem uma trajetória de uma variável $\mathbf{x}$, chamada de variável de estado no contexto da teoria de controle, a partir de um valor inicial até chegar a um ponto arbitrariamente próximo da solução. Muitas classificações dos algoritmos existentes podem ser realizadas, mas sem dúvida a principal classificação os divide em algoritmos contínuos e discretos.

Os algoritmos contínuos apresentam uma lei de atualização das variáveis de estado caracterizada por equações diferenciais ordinárias (EDO), que podem ser representadas, na sua forma mais simples, da seguinte maneira:

$$\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), t) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.1}$$

onde $\mathbf{x} : \mathbb{R} \to \mathbb{R}^n$ é uma variável vetorial dependente do tempo, e $\mathbf{u} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ é um campo vetorial chamado de lei de atualização das variáves de estado dependente, de modo geral, da própria variável de estado e do tempo. A equação (1.1) representa os chamados sistemas não autônomos, quando o mapeamento depende explicitamente da variável temporal. A teoria que trata sobre sistemas não autônomos é mais complexa que aquela que trata com sistemas autônomos, onde a lei de atualização das variáveis não depende explicitamente da variável temporal, mas apenas da variável de estado. Os sistemas autônomos podem ser representados pela seguinte lei de atualização das variáveis:

$$\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t)) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.2}$$

No presente trabalho, trataremos exclusivamente de sistemas autônomos.

Os algoritmos contínuos eram implementados antigamente com computadores analógicos. Hoje em dia, eles podem ser implementados como redes neurais. Cabe mencionar que uma motivação importante para o estudo e desenvolvimento de algoritmos contínuos consiste em que a partir destes podem ser derivados algoritmos discretos, e muitas

características destes podem ser estudadas com eficiência a partir dos equivalentes contínuos.

Os algoritmos discretos apresentam uma lei de atualização das variáveis que pode ser caracterizada como uma equação a diferenças, onde a cada passo ou iteração calcula-se um novo ponto, ou valor da variável de estado, a partir do anterior (ou anteriores). Assim, o algoritmo discreto gera uma seqüência discreta de pontos a partir de um valor inicial da variável de estado. Na sua forma mais simples, a lei de atualização das variáveis pode ser representada como a seguinte equação a diferenças:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}(\mathbf{x}_k) \quad \text{a partir de algum ponto inicial } \mathbf{x}_0 \tag{1.3}$$

onde o índice k representa o número de iteração, a função $\mathbf{u}(\mathbf{x}_k)$ é a lei de atualização das variáveis (que no caso autônomo, não depende explicitamente do número de iteração k, esta lei de atualização é referida na literatura de otimização como passo de iteração), e α_k é o comprimento do passo.

Os algoritmos discretos podem ser desenvolvidos a partir de equivalentes contínuos discretizados por algum método específico. Dentre os métodos de discretização, cabe mencionar o método de Euler (ver [4]). Aplicando este método no sistema contínuo (1.2), obtem-se o algoritmo discreto representado pela equação em diferenças (1.3), para algum comprimento do passo de iteração.

A associação de métodos numéricos com a teoria de controle não é nova. Por exemplo, Soderlind ([70]) aplica a teoria de controle mas com a finalidade de calcular comprimentos dos passos de algoritmos discretos de maneira adaptativa. Goh ([33]) desenvolve dois algoritmos discretos escolhendo o passo de iteração como uma seqüência de problemas de controle ótimo, aplicando função inversa de Liapunov, porém não calcula o comprimento do passo de maneira ótima. Chao e de Figueiredo ([21]) propõem um método contínuo (e sua discretização por Euler) $\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}(\mathbf{x})(\mathbf{f}(\mathbf{x}) - \mathbf{u}(\mathbf{x}))$, onde a função de controle $\mathbf{u}(\mathbf{x})$ é escolhida de maneira ótima com o objetivo de minimizar uma função de custo ao longo de um intervalo predeterminado. Chehab e Laminie ([22]) fazem para um algoritmo discreto proposto, um estudo de estabilidade baseado em uma metodologia empregada em controle; tal análise está baseada nos trabalhos

de Bhaya e Kaszkurewicz. Porém, todas estas abordagens são mais restritas que a proposta por estes autores.

Especificamente, [33], propõe uma formulação do problema de minimização irrestrita como um problema de controle ótimo.

Dado o problema de minimizar uma função $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, é estabelecida uma determinada lei de atualização das variáveis $\dot{\mathbf{x}} = \mathbf{u}(t)$. Portanto:

$$\frac{\partial f(\mathbf{x})}{\partial t} = \nabla^T f(\mathbf{x})\dot{\mathbf{x}} = \nabla^T f(\mathbf{x})\mathbf{u}(t)$$

Integrando ambos membros da equação:

$$f(\mathbf{x}(t_f)) = f(\mathbf{x}(0)) + \int_0^{t_f} \nabla^T f(\mathbf{x}(t))\mathbf{u}(t)\partial t \tag{1.4}$$

o que conduz ao seguinte problema de controle ótimo:

$$\begin{gathered}\min f(\mathbf{x}(t_f)) = \min \left[f(\mathbf{x}(0)) + \int_0^{t_f} \nabla f(\mathbf{x}(t))\mathbf{u}(t)\partial t \right] \\ \text{s.t } \dot{\mathbf{x}} = \mathbf{u}, \quad \mathbf{x}(0) = \mathbf{x}_0\end{gathered}$$

Chao e de Figueiredo ([21]), propõem a formulação para achar um zero de uma função vetorial $\mathbf{f}(\mathbf{x}(t)) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ como um problema de controle ótimo, através da seguinte equação:

$$\dot{\mathbf{f}}(\mathbf{x}(t)) = -\mathbf{f}(\mathbf{x}) + \mathbf{u} \tag{1.5}$$

sendo $\mathbf{u}$ a lei de atualização das variáveis de estado.

O problema consiste em selecionar uma lei de controle ótima $\mathbf{u}$ que minimize a seguinte função de custo:

$$J = \frac{1}{2}\int_0^{t_f} \|\mathbf{u}\|^2 \partial t$$

sujeita às condições de contorno:

$$\mathbf{f}(\mathbf{x}(0)) = \mathbf{f}_0, \quad \mathbf{f}(\mathbf{x}(t_f)) = \mathbf{0}$$

Os autores demonstram que a lei de controle $\mathbf{u}$ que minimiza esta função de custo

está dada por:

$$\mathbf{u} = -\frac{e^{-(t_f - t)}}{\sinh t_f}\mathbf{f}_0$$

o que conduz à seguinte variação temporal da função objetivo:

$$\mathbf{f}(t) = \mathbf{f}_0 \frac{\sinh(t_f - t)}{\sinh t_f}$$

Pelo fato destas propostas serem formuladas em termos de controle ótimo, elas levam a projetos de algoritmos razoavelmente complexos, requerendo, em geral, soluções de problemas de contorno que podem ser, inclusive, mais difíceis de resolver do que o problema original de otimização ou o de achar zeros. Observa-se que, nesse contexto, o autor Goh se refere aos algoritmos propostos por ele ([33]) como apenas conceituais. No caso do método proposto por Chao e de Figueiredo, observa-se que a solução em forma fechada do problema de contorno (1.5) é crucial para a simplificação que define um novo algoritmo iterativo (ver [21]), e esta simplificação poderá não ocorrer com problemas mais complexos, como por exemplo os de otimização restrita.

Por outro lado, a metodologia apresentada por Bhaya e Kaszkurewicz em [13], [14], [11] e [10] consiste em interpretar os algoritmos, tanto contínuos quanto discretos, como sistemas dinâmicos de controle com realimentação, isto é, sistemas de controle em malha fechada. Utiliza-se a idéia básica de controle realimentado para o problema de regulação, i.e. o problema de tornar o valor assumido por uma função igual a um valor de referência, através de uma escolha adequada dos vários objetos envolvidos na definição do problema de regulação. Esses objetos, chamados planta e controlador, serão explicitados no próximo capítulo no contexto do problema de achar zeros, mas cabe enfatizar aqui que a metodologia é mais geral e será utilizada ao longo deste trabalho em diversos contextos.

Os diferentes algoritmos surgem de diferentes escolhas da lei de controle do sistema. Estas escolhas são realizadas seguindo o método de funções de Liapunov de controle (CLF) e controle ótimo de Liapunov (LOC). A metodologia de funções de Liapunov de controle consiste no seguinte: uma função de Liapunov candidata positiva definida e dependente da variável de estado é escolhida, e a função de controle do sistema pode ser *qualquer* uma que faça com que a derivada temporal da função de Liapunov ao longo das trajetórias do sistema em malha fechada seja negativa definida, garantindo assim

a estabilidade do sistema. Já a metodologia de controle ótimo de Liapunov consiste na escolha da função de controle que minimize a derivada temporal da função de Liapunov candidata ao longo das trajetórias do sistema em malha fechada, garantindo assim a maior taxa de convergência ao ponto de equilíbrio do sistema (da EDO) dada a função de Liapunov escolhida. Assim, a metodologia CLF/LOC é utilizada para a derivação de diversos algoritmos, assim como para a análise de algumas das suas características tais como taxa de convergência. Alguns destes algoritmos já são amplamente conhecidos na literatura, mas nunca antes foram abordados sob esta perspectiva.

Os algoritmos contínuos assim achados podem ser discretizados pelo método de Euler. O comprimento do passo é escolhido de maneira ótima utilizando a metodologia de controle ótimo de Liapunov (LOC). No contexto discreto, esta metodologia consiste em escolher uma função de Liapunov candidata discreta, e o cálculo do comprimento do passo é realizado de maneira de minimizar a variação desta função a cada iteração. Desta forma, o comprimento do passo assim escolhido será aquele que proporciona maior diminuição da função de Liapunov discreta de uma iteração para a seguinte.

No contexto de achar zeros de funções, dentre os muitos algoritmos contínuos existentes, a classe de algoritmos de continuação ou homotopia [1, 2] também utiliza uma EDO do tipo (1.2) como ponto de partida dos métodos chamados preditor-corretor, entretanto, não há a utilização dos conceitos de controle no projeto e na análise destes algoritmos.

Neste mesmo contexto, podemos mencionar ainda os algoritmos contínuos propostos, desenvolvidos, estudados ou aplicados em [11], [13, c. 2], [14], [79, c. 5], [29, c. 2], [18], [21], [8], [41], [40],[59], [68], [5], [6], [49], [39], entre muitos outros. Uma bibliografia on-line pode ser encontrada em McNamee, 1993 [58]. Mais comentários contextualizando essas contribuições serão apresentadas nos capítulos 2 e 3.

Ainda no contexto de achar zeros de funções, dentre os algoritmos discretos propostos, desenvolvidos ou utilizados na bibliografia, podemos mencionar [13, c. 2], [12], [10], [14], [33], [70], [35], [22], [51], [84], [52], [66], [43], [25] e suas referências, também entre muitos outros, os que serão contextualizados também nos capítulos 2 e 3.

Alguns destes algoritmos, tal como o algoritmo de Newton, nas suas versões contínua e discreta, foram amplamente estudados e abordados na bibliografia (ver, por

exemplo, [18], [87], [34], [24], [57], [43] e [41]); outros são certamente menos conhecidos e analisados. Porém, os diversos algoritmos existentes são usualmente projetados caso a caso, não existindo uma metodologia genérica de derivação e análise de cada um deles. Também, nos casos dos algoritmos discretos, a escolha do comprimento do passo de iteração, freqüentemente, é realizada com a única finalidade de garantir a convergência da seqüência de pontos calculados, sem uma metodologia genérica de escolha de um comprimento do passo que seja ótimo para um determinado critério.

No capítulo 2, será apresentada a abordagem CLF/LOC ([13, 14]) no desenvolvimento e análise de diferentes algoritmos de primeira ordem para achar zeros de funções. Algoritmos de primeira ordem podem ser caracterizados por uma EDO de primeira ordem[1], no caso contínuo (equação 1.2), ou uma equação em diferenças onde a lei de atualização da variável depende apenas do variável de estado na iteração corrente, no caso discreto (equação 1.3). Tanto no caso contínuo como discreto, serão estudadas diferentes características dos algoritmos assim desenvolvidos, tais como pontos singulares, conseqüências da presença de singularidades estranhas, regiões de convergência de cada zero, taxa de convergência e as vantagens da utilização de um conjunto de algoritmos ao invês de um único algoritmo.

Os algoritmos assim desenvolvidos são testados com uma ampla quantidade de funções clássicas na bibliografia, com diversos números de variáveis.

Os algoritmos de primeira ordem apresentados no capítulo 2 podem ser utilizados também para achar o mínimo de uma função escalar contínua convexa $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ irrestrita, pois este ponto mínimo corresponde ao zero do seu gradiente. Porém, existem outros algoritmos específicos para resolver este problema. Dentre eles, o mais mencionado na bibliografia é o *steepest descent* (ver, por exemplo, [79, c. 5]). Este consiste em estabelecer as trajetórias da variável de estado na direção contrária do gradiente da função, garantindo desta forma o decrescimento desta. Este algoritmo, assim como outros mencionados na bibliografia ([6], [5]), podem ser vistos como um sistema dinâmico de controle em malha fechada, onde o algoritmo específico é determinado pelo ganho do controlador. Utilizando a metodologia CLF, outros controladores, e assim

[1]Deve ser enfatizado que o termo "primeira ordem", não se refere à taxa de convergência do algoritmo.

também outros algoritmos, podem ser desenvolvidos e suas características analisadas. Os algoritmos assim derivados podem ser discretizados por Euler com seu comprimento do passo calculado de maneira ótima por meio da metodologia LOC.

Este projeto e análise de algoritmos de primeira ordem para achar mínimos de funções escalares convexas também será apresentado no capítulo 2.

No capítulo 3 serão apresentados algoritmos de segunda ordem para achar zeros de funções. Os algoritmos de segunda ordem estão caracterizados por um modelo matemático que pode ser representado como uma equação diferencial ordinária de segunda ordem[2]:

$$\ddot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), \dot{\mathbf{x}}(t)) \quad \mathbf{x}(0) = \mathbf{x}_0,\ \dot{\mathbf{x}}(0) = \dot{\mathbf{x}}_0 \tag{1.6}$$

onde $\mathbf{u}(\mathbf{x}(t), \dot{\mathbf{x}}(t))$ é lei de controle dependente das variáveis dinâmicas do sistema.

Este sistema também pode ser interpretado como um sistema dinâmico de controle em malha fechada, onde agora o dispositivo controlador também possui uma dinâmica própria, sendo que o sistema pode ser representado na forma:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \varphi(\mathbf{u}(t), \mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \\ \dot{\mathbf{u}}(t) &= \psi(\mathbf{u}(t), \mathbf{x}(t)), \quad \mathbf{u}(0) = \mathbf{u}_0 \end{aligned} \tag{1.7}$$

onde a função $\varphi(\mathbf{u}, \mathbf{x})$ depende da escolha da planta e a função $\psi(\mathbf{u}, \mathbf{x})$ depende da escolha do controlador, sendo que este último agora possui uma dinâmica determinada.

A escolha de planta e controlador é realizada pela metodologia de funções de Liapunov de controle e controle ótimo de Liapunov.

A presença de uma dinâmica no controlador visa dar mais facilidade aos algoritmos para achar zeros ali onde os algoritmos de primeira ordem se mostram falhos.

Os algoritmos de segunda ordem desenvolvidos com a metodologia CLF são discretizados por Euler, e seus comprimentos de passo, que podem ser vistos como ganhos dos controladores discretos, calculados de maneira ótima pela metodologia LOC. De maneira geral, a discretização de (1.7) pode ser representada como:

[2]Deve ser enfatizado que o termo "segunda ordem", não se refere à taxa de convergência do algoritmo.

$$\begin{aligned} \mathbf{x}_{k+1} &= \varphi(\mathbf{x}_k, \mathbf{u}_k, \alpha_k) \\ \mathbf{u}_{k+1} &= \psi(\mathbf{x}_k, \mathbf{u}_k, \beta_k) \end{aligned} \tag{1.8}$$

sendo α_k e β_k os comprimentos dos passos.

Algoritmos específicos de segunda ordem para achar mínimos de funções escalares também são desenvolvidos pela metodologia CLF, sempre interpretando estes como sistemas dinâmicos de controle. Pretende-se que, ao contrário dos algoritmos de primeira ordem para minimização de funções, quando estas não apresentam convexidade, os de segunda ordem sejam capazes de ultrapassar eventuais mínimos locais para achar mínimos globais.

Algoritmos de segunda ordem não são muito abundantes na bibliografia. Alguns destes são apresentados em [19], [3] e [29, s. 2.5]. Entretanto, estas referências não interpretam os algoritmos como sistemas dinâmicos de controle, assim como não são propostas metodologias genéricas de projeto e análise dos diferentes algoritmos, o que será aqui realizado utilizando a teoria de controle.

As leis de atualização das variáveis de estado e de controle dos algoritmos de segunda ordem específicos para minimização de funções escalares, também são discretizadas por Euler e os comprimentos dos passos calculados de maneira ótima segundo a metodologia LOC.

No capítulo 4, é abordado o problema geral de otimização convexa, ou problema de achar o mínimo de uma função escalar convexa diferenciável restrita a um conjunto convexo.

Uma das possíveis abordagens para o problema de otimização restrita tem suas raízes no método de penalização exata, o qual transforma o problema de otimização restrita em um problema de otimização irrestrita não suave, mesmo que o problema restrito seja suave. A interpretação de penalização exata em termos de controle foi iniciada nos trabalhos [78, 32], a idéia principal sendo a identificação da descontinuidade associada à penalização exata com uma função de chaveamento, gerando um sistema dinâmico conhecido como sistema a estrutura variável na literatura de controle. A utilização de sistemas de controle a estrutura variável também se mostra adequada na resolução de problemas de otimização convexa, isto é, de minimização de funções

escalares convexas sujeitas a restrições de desigualdade convexas e de igualdade afins (ver [78], [86, s. 6.9], [32] e [27], para o caso de programação linear e quadrática).

No capítulo 4 será apresentado o desenvolvimento de um algoritmo a estrutura variável utilizando uma função de Liapunov de controle para resolver problemas de otimização convexa, o qual representa uma extensão daquele apresentado em [32] e que apresenta diversas vantagens com respeito a outros algoritmos tratados na bibliografia. Dentre estas vantagens podemos adiantar algumas tais como a não utilização do operador de projeção ortogonal, a ausência de limitações com respeito às restrições que conformam o conjunto viável, a manutenção da dimensão do problema como o número de variáveis, entre outras que serão especificadas oportunamente. O algoritmo também é discretizado por Euler e seu comprimento de passo otimizado pela metodologia LOC. Uma versão preliminar do algoritmo aqui desenvolvido, nas suas versões contínua e discreta, foi apresentado em [61].

No capítulo 5, é abordado o problema de desigualdades variacionais.

Algoritmos específicos para a resolução de problemas de desigualdades variacionais não são muito abundantes na bibliografia, e os apresentados, tanto contínuos quanto discretos, apresentam diversas desvantagens, como utilização de projeção ortogonal, limitações com respeito à função objetivo ou às restrições, entre outras.

Seguindo idêntica metodologia utilizada no capítulo 4, no desenvolvimento do algoritmo para a resolução do problema geral de otimização convexa, será implementado um novo algoritmo para resolver problemas de desigualdades variacionais baseado em um sistema de controle em malha fechada a estrutura variável, o qual não possui as desvantagens mencionadas. Este algoritmo é discretizado por Euler e seu comprimento de passo calculado pela metodologia LOC, com algumas mudanças para garantir a convergência das trajetórias ao conjunto solução.

Uma versão preliminar do algoritmo discreto foi apresentada em [60].

Finalmente, no capítulo 6 são apresentadas as conclusões gerais, as propostas de trabalho futuro e são explicitadas as contribuições desta tese.

Capítulo 2

Algoritmos de primeira ordem destinados a achar zeros de funções vetoriais ou mínimos de funções escalares como sistemas dinâmicos de controle

2.1 Introdução

O problema de achar os zeros de uma função não condicionada de n variáveis tem muitas aplicações concretas em diversas áreas da ciência e da engenharia. O problema abrange aquele de encontrar os pontos extremos e/ou selas de uma função escalar $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, pois é equivalente a achar os zeros do seu gradiente, quando este está definido. Muitos problemas matemáticos podem ser reduzidos ao problema de achar zeros de uma função ou de minimização de uma função.

Em muitos casos o problema pode ser resolvido analiticamente, por exemplo quando se trata de resolver o sistema de equações lineares algébricas expressado como $A\mathbf{x} = \mathbf{b}$ (equivalente a achar os zeros da função $\mathbf{f}(\mathbf{x}) = \mathbf{b} - A\mathbf{x}$), sendo a matriz A inversível. Mas em outros casos a resolução analítica não é possível.

Para esses casos, existe a possibilidade de empregar algoritmos que desenvolvem

uma trajetória da variável $\mathbf{x}$ a partir de um valor inicial até chegar a um ponto arbitrariamente próximo de um zero. Existe uma grande quantidade destes métodos que tem sido desenvolvidos e aplicados há muito tempo, e são muito abundantes na bibliografia.

Esta seção apresentará a abordagem de Bhaya e Kaszkurewicz, apresentada em [13] e [14], na dedução e análise de diferentes algoritmos, tanto contínuos quanto discretos.

Um dos problemas tratados na teoria de controle é o problema da regulação, também chamado de controle ponto a ponto. Este consiste em, dado um determinado sistema, geralmente referido como *planta*, cuja resposta é caracterizada por determinadas variáveis a serem controladas, encontrar um mecanismo que mantenha ou regule estas variáveis a valores constantes, mesmo que esta planta tenha sua resposta afetada por diferentes distúrbios. Assim, estas variáveis que caraterizam a resposta, são permanentemente comparadas com o valor desejado ou referência, através do chamado laço de realimentação ou *feedback*; a diferença entre a referência e a resposta da planta gera uma nova variável chamada de resíduo ou erro, o qual representa o desvio entre a resposta real da planta e seu valor desejado. Este erro é utilizado para alimentar um controlador dependente de determinados parâmetros, este mecanismo controlador será o responsável por gerar uma variável chamada de excitação, com que será alimentada a planta, de maneira tal de alterar a resposta até conseguir que o resíduo ou erro tenda para zero. O mecanismo em malha fechada, portanto, resulta em um erro e uma excitação igual a zero, e assim permanecerá o sistema sempre que um distúrbio não desvie a resposta do valor desejado. Na figura 2.1 é representado um mecanismo de controle em malha fechada.

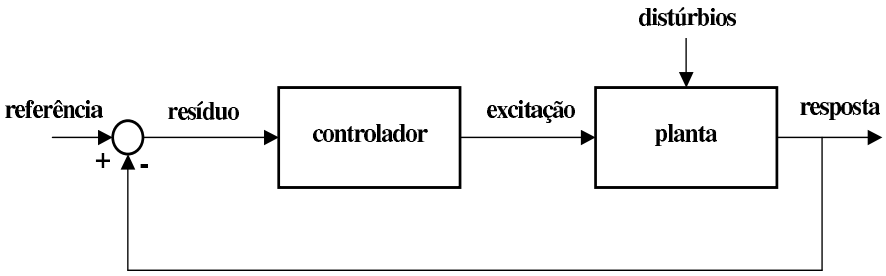

Figura 2.1: Sistema de controle em malha fechada

A proposta em [13], [11] e [10] consiste em utilizar a teoria de controle ponto a ponto, não mais para regular as variáveis de saída, ou resposta, de um sistema físico, mas para

conseguir que uma determinada função matemática atinja um valor desejado constante. No caso do problema de achar as raízes de uma função não linear $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$, esta função pode representar a planta do sistema, em cujo caso o valor desejado é, obviamente, zero. Quando a função atinge este valor, a variável $\mathbf{x}$, também chamada de variável de estado, tomará o valor de uma raíz da função, referida como $\mathbf{x}^*$.

Bhaya e Kaszkurewicz ([13, c. 2] e [11]) propõem como método para o desenvolvimento do mecanismo controlador, a utilização de funções de controle de Liapunov (CLF). Através deste método, diversos mecanismos, equivalentes a diversos algoritmos, podem ser derivados e suas convergências analisadas.

Os algoritmos discretos assim derivados são discretizados por Euler, com os comprimentos de passo calculados de maneira ótima pelo método de controle ótimo de Liapunov (LOC).

O presente capítulo está organizado da seguinte maneira. Na seção 2 é apresentada a teoria de controle como ferramenta para a derivação de algoritmos para achar zeros de funções vetoriais não lineares.

Na seção 3 são detalhadas as já mencionadas metodologias CLF e LOC, neste mesmo contexto.

Na seção 4 são derivados os algoritmos de primeira ordem contínuos apresentados em [13] e [14]. São realizadas algumas considerações sobre o algoritmo de Newton, e são simulados com diversas funções de teste.

Na seção 5 é estudado o tema de singularidades estranhas para o algoritmo de Newton e de Branin ([18]), e são propostos métodos de linearização para o algoritmo de Newton.

Na seção 6 são discretizados os algoritmos apresentados na seção 4, são realizadas simulações com diversas funções de teste, e é apresentada a metodologia de utilização de um conjunto, ou "time" de algoritmos.

Na seção 7 são estudadas as bacias de atração dos zeros de diversas funções de teste para os algoritmos apresentados.

Na seção 8 são desenvolvidos algoritmos específicos para minimização de funções escalares convexas como sistemas dinâmicos de controle. Estes são simulados com diversas funções de teste, discretizados com comprimentos de passo calculados por LOC,

e os algoritmos discretos assim desenvolvidos também são aplicados em simulações com as mesmas funções de teste.

Finalmente, na seção 9 são apresentadas as conclusões deste capítulo.

2.2 A teoria de controle utilizada para achar zeros de uma função

O objetivo desta seção é mostrar como um sistema dinâmico contínuo em malha fechada pode ser utilizado para achar zeros de uma função contínua não linear $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ através de uma perspectiva de controle por realimentação. Pretende-se, portanto, encontrar um vetor $\mathbf{x}^*$ tal que

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{0} \tag{2.1}$$

Em geral, para uma função não linear existirá mais de uma solução.

A função $\mathbf{f}(\mathbf{x})$ será interpretada como a planta cujo valor se deseja controlar. O valor desejado, ou referência, é obviamente zero. O resíduo ou erro será a diferença entre a referência e a resposta da planta, ou valor da função. Isto é:

$$\mathbf{r} = \mathbf{0} - \mathbf{f}(\mathbf{x}) = -\mathbf{f}(\mathbf{x}) \tag{2.2}$$

A norma deste vetor de erro ou resíduo pode ser interpretada como a distância entre o valor real da variável $\mathbf{x}$, também chamada de variável de estado, e seu valor desejado.

Observe-se que, se o objetivo do sistema de controle é fazer o resíduo tender a zero, isto equivale a fazer tender a variável $\mathbf{x}(t)$ ao valor desejado $\mathbf{x}^*$.

O resíduo alimentará um controlador (função de parâmetros dependentes) que por sua vez gerará o sinal de excitação $\mathbf{u}(t)$. Esta variável de excitação atuará diretamente sobre a variável de estado $\mathbf{x}(t)$, provocando sua variação temporal, isto é, $\dot{\mathbf{x}} = \mathbf{u}$. Assim, quando a planta atingir o valor desejado, $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, o resíduo será igual a zero, assim como o valor da excitação $\mathbf{u}$, o qual implica que a variável de estado $\mathbf{x}$ permanecerá constante e igual ao valor de uma raíz da função $\mathbf{x}^*$, que será um ponto de equilíbrio do sistema em malha fechada.

Em termos de controle, a equação que relaciona a excitação com a variável de estado é chamada equação de estado, e a equação que relaciona a variável de estado com a resposta do sistema é chamada equação de saída do sistema. Chamando a variável de saída, ou resposta, de $\mathbf{y}(t)$, podemos escrever:

$$\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{x}(t)}{\partial t} = \mathbf{u}(t) \quad \text{equação de estado} \tag{2.3}$$

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t)) \quad \text{equação de saída} \tag{2.4}$$

Observe-se que, de (2.2) e (2.4):

$$\mathbf{y}(t) = -\mathbf{r}(t) \tag{2.5}$$

O problema de achar os zeros de $\mathbf{f}(\mathbf{x})$ pode ser estabelecido em termos da teoria de controle como segue: encontrar uma variável de excitação $\mathbf{u}(t)$ tal que o resíduo tenda a zero e consequentemente a variável de controle $\mathbf{x}(t)$ ao valor desejado $\mathbf{x}^*$. Este é o problema de regulação onde a saída deve ser controlada ao valor da referência, neste caso zero.

A figura 2.2 mostra um diagrama de blocos do sistema em malha fechada, ou realimentado, que representa a dinâmica das variáveis que interatuam neste sistema de controle.

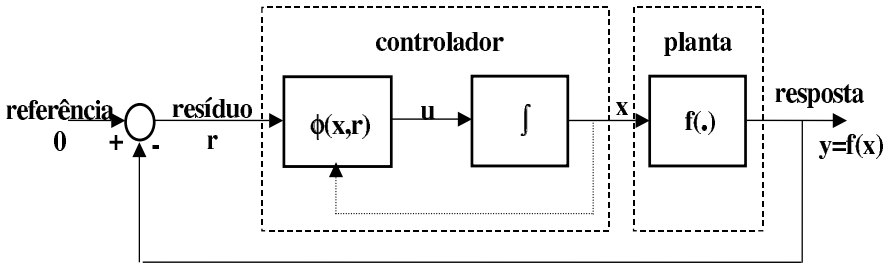

Figura 2.2: Diagrama de blocos representativo do sistema contínuo

Observe-se que podemos interpretar a variável de excitação como dependente da variável de estado e do resíduo:

$$\mathbf{u} = \phi(\mathbf{x}, \mathbf{r}) \tag{2.6}$$

levando ao chamado sistema em malha fechada, da forma

$$\dot{\mathbf{x}} = \phi(\mathbf{x}, \mathbf{r}) \tag{2.7}$$

O objetivo da metodologia CLF será, portanto, achar uma função $\phi(\mathbf{x}, \mathbf{r})$ que atinja o objetivo desejado.

2.3 Metodologia CLF/LOC para projetar algoritmos para achar zeros de funções

A metodologia de funções de Liapunov de controle (CLF) e de controle ótimo de Liapunov (LOC), pode ser utilizada para achar diversas funções de controladores (2.6).

O esquema geral é o seguinte: uma função de Liapunov candidata positiva definida é escolhida. A derivada no tempo $\dot{V}$ ao longo das trajetórias do sistema em malha fechada é calculada, gerando uma função da variável de estado $\mathbf{x}$, da saída $\mathbf{y}$ e da variável de entrada ou excitação $\mathbf{u}$. A metodologia CLF consiste na escolha de *qualquer* função $\mathbf{u}(\mathbf{x}, \mathbf{r})$ que provoque que $\dot{V}$ seja negativa definida, e portanto o sistema estável (para mais informação sobre o método direto de Liapunov, ver [69]). Mediante outros lemas ou teoremas tais como Barbalat ou La Salle, pode se demonstrar a estabilidade assintótica do sistema.

Assim, diversas leis de controle, e portanto diversos algoritmos, podem ser derivados através destes métodos.

A metodologia de LOC vai além, esta consiste em utilizar os graus de liberdade disponíveis escolhendo a entrada $\mathbf{u}$ que faz a derivada temporal da função de Liapunov $\dot{V}$ tão negativa quanto possível. Esta tentativa se relaciona, por um lado, com o chamado método *speed gradient*, quando não há restrições sobre a variável de controle, por outro lado, quando uma variável de controle limitada é aplicada, frequentemente o controle ótimo emprega uma função *signum*, o que está relacionado com o chamado controle a estrutura variável. De fato, existem relações próximas entre controle ótimo de Liapunov, controle *speed gradient*, controle a estrutura variável e controle ótimo, as quais são examinadas em [13, c. 3].

Neste ponto, cabe apresentar a seguinte definição: uma trajetória define-se como **convergente** a um determinado ponto se, quando o tempo tende a infinito (número de iterações no caso discreto), a trajetória tende a esse ponto. A convergência de uma trajetória de todas as variáveis de estado de um sistema a um ponto de equilíbrio, implica na estabilidade assintótica desse sistema; portanto aqui nos referiremos indistintamente a estabilidade assintótica como a convergência.

Uma noção amplamente utilizada para quantificar a velocidade de convergência de algoritmos discretos é a seguinte ([40]): o método (1.3) converge ao resultado $\mathbf{x}^*$ a uma taxa $q > 1$ uniformemente em uma bola $B(\mathbf{x}^*, \rho)$, onde ρ é uma constante positiva, se existe $\beta > 0$ tal que

$$\|\mathbf{x} + \mathbf{v}(\mathbf{x}) - \mathbf{x}^*\| \leq \beta \|\mathbf{x} - \mathbf{x}^*\|^q \tag{2.8}$$

para todo $\mathbf{x} \in B(\mathbf{x}^*, \rho)$. Isto é, a seqüência de pontos $\mathbf{x}_k$, uma vez que entra na bola $B(\mathbf{x}^*, \rho)$, convergirá a $\mathbf{x}^*$ melhorando a cada iteração a precisão de $\mathbf{x}_k$ em relação a $\mathbf{x}^*$, q vezes.

2.4 Algoritmos de primeira ordem contínuos para achar zeros de funções

Nesta seção serão derivados diferentes algoritmos de primeira ordem para achar zeros de uma função $\mathbf{f}(\mathbf{x})$. Estes algoritmos são apresentados em [13, c. 2.1]. Os algoritmos serão expressados em função do resíduo ou erro $\mathbf{r}$, de maneira tal que o ponto de equilíbrio desejado será $\mathbf{r} = \mathbf{0}$. É assumido que uma mudança de coordenadas da variável $\mathbf{x}$ à variável $\mathbf{r}$ é possível. Desde que $\mathbf{r} = -\mathbf{f}(\mathbf{x})$, pelo teorema da função inversa, é assumido que o jacobiano é inversível, então $\mathbf{f}$ é localmente inversível, isto é, a desejada mudança de coordenadas existe.

Derivando (2.2) em função do tempo:

$$\dot{\mathbf{r}} = \frac{d\mathbf{r}}{dt} = -\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} = -D_{\mathbf{f}}(\mathbf{x})\dot{\mathbf{x}} \tag{2.9}$$

onde $D_{\mathbf{f}}(\mathbf{x})$ indica a matriz jacobiana de $\mathbf{f}$ no ponto $\mathbf{x}$.

De (2.9) e (2.3), pode-se escrever:

$$\dot{\mathbf{r}} = -D_{\mathbf{f}}(\mathbf{x})\mathbf{u} \tag{2.10}$$

Uma escolha simples da função de Liapunov candidata consiste na norma quadrado do resíduo:

$$V(\mathbf{r}) := \frac{1}{2}\|\mathbf{r}\|_2^2 = \frac{1}{2}\mathbf{r}^T\mathbf{r} \tag{2.11}$$

a qual é evidentemente positiva definida[1] e assume o valor zero apenas no ponto $\mathbf{r}(\mathbf{x}) = \mathbf{0}$. A derivada de (2.11) ao longo das trajetórias de (2.10), é dada por

$$\dot{V}(\mathbf{r}) = \mathbf{r}^T\dot{\mathbf{r}} \tag{2.12}$$

Substituindo (2.10) em (2.12):

$$\dot{V}(\mathbf{r}) = -\mathbf{r}^T D_{\mathbf{f}}(\mathbf{x})\mathbf{u}(\mathbf{r}) \tag{2.13}$$

Esta equação é o ponto de partida para o projeto de diferentes algoritmos através da metodologia CLF, consistente na escolha de diferentes leis de controle $\mathbf{u}$ que façam (2.13) negativa definida, o que garante que o resíduo ou erro tenderá para zero e o sistema em malha fechada será assintoticamente estável; portanto a variável de estado $\mathbf{x}(t)$ tenderá para o valor desejado $\mathbf{x}^*$.

Teorema 2.4.1 *([13, teorema 2.1]) Dada uma função* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$*, suponha-se* $\mathbf{x}^*$ *ser um zero de* $\mathbf{f}$ *tal que o jacobiano* $D_{\mathbf{f}}(\mathbf{x})$ *seja inversível nalguma vizinhança* $\mathcal{N}(\mathbf{x}^*)$ *de* $\mathbf{x}^*$*. Então, para toda condição inicial* $\mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$*, para as trajetórias correpondentes do sistema dinâmico*

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{r}), \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*) \tag{2.14}$$

onde $\mathbf{u}(\mathbf{r})$ *é a lei de controle escolhida pela metodologia CLF/LOC da maneira especificada, a variável de estado converge ao zero* $\mathbf{x}^*$ *de* $\mathbf{f}(.)$*.*

Note-se que este teorema resulta em estabilidade local, desde que sua prova depende

[1] Uma função escalar contínua $V(\mathbf{x})$ é globalmente positiva definida se $V(\mathbf{0}) = 0$, e se, para todo $\mathbf{x} \neq \mathbf{0}$, $V(\mathbf{x}) > 0$ ([69, p. 59])

das propriedades de estabilidade do sistema (2.10) e (2.12), o qual é obtido após uma mudança local de coordenadas. O tipo de estabilidade, exponencial, assintótica ou em tempo finito depende da lei de controle particular escolhida.

A seguir, serão apresentados cinco algoritmos de primeira ordem derivados com esta metodologia e inicialmente apresentados em [13, c. 2].

1) Primeira escolha

$$\mathbf{u} := D_{\mathbf{f}}^{-1} P\mathbf{r} \tag{2.15}$$

onde P é uma matriz positiva definida. Observe-se que com esta escolha, substituindo em (2.13):

$$\dot{V} = -\mathbf{r}^T P\mathbf{r} \tag{2.16}$$

a qual é negativa definida e só toma o valor 0 em $\mathbf{r} = \mathbf{0}$, sendo portanto o sistema assintóticamente estável, estacionando-se apenas no ponto $\mathbf{x}$ tal que $\mathbf{r}(\mathbf{x}) = \mathbf{0}$. A dinâmica do sistema em malha fechada está dada por:

$$\dot{\mathbf{x}} = D_{\mathbf{f}}^{-1} P\mathbf{r} = -D_{\mathbf{f}}^{-1} P\mathbf{f}(\mathbf{x}) \tag{2.17}$$

Observe-se que, para $P = I$, este método coincide com o método de Newton contínuo (CN), amplamente estudado na bibliografia, razão pela qual aqui também receberá esse nome, mesmo com outro valor de ganho P.

Para a escolha particular $P = \alpha I$, onde $\alpha > 0$, podemos escrever (2.17) como

$$D_{\mathbf{f}}\dot{\mathbf{x}} = -\alpha\mathbf{f}(\mathbf{x}) \quad \Rightarrow \quad \dot{\mathbf{f}}(\mathbf{x}) = -\alpha\mathbf{f}(\mathbf{x}) \tag{2.18}$$

e para um determinado valor inicial da variável de estado $\mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$,

$$\mathbf{f}(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}_0)e^{-\alpha t} \tag{2.19}$$

sendo portanto o sistema exponencialmente estável. Observe-se que, apesar de (2.16) ser negativa definida, a estabilidade pode ser apenas local, pois, pelo teorema 2.4.1, estamos assumindo o jacobiano inversível apenas numa vizinhança $\mathcal{N}(\mathbf{x}^*)$, isto é, o algoritmo nunca poderá atravesar as hipersuperfícies determinadas pelo locus da equação

$\det(D_{\mathbf{f}}(\mathbf{x})) = 0$.

2) Segunda escolha

$$\mathbf{u} := D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{r}) \tag{2.20}$$

Substituindo em (2.13):

$$\dot{V} = -\mathbf{r}^T\text{sgn}(\mathbf{r}) = -\|\mathbf{r}\|_1 \tag{2.21}$$

a qual é negativa definida e só toma o valor 0 em $\mathbf{r} = \mathbf{0}$, sendo assim o algoritmo assintoticamente estável. A dinâmica do sistema em malha fechada está dada por:

$$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{f}(\mathbf{x})) \tag{2.22}$$

Este sistema dinâmico é chamado de Newton a estrutura variável (NV), e possui como principal característica um lado direito descontínuo. Tais funções originam sistemas a estrutura variável, e o comportamento típico das trajetórias na dinâmica destes sistemas em malha fechada é o de modos deslizantes. Em particular, se sobre uma superfície $h_i(\mathbf{x}) = 0$, as condições

$$\lim_{h_i \to 0^+} h_i(\mathbf{x}) < 0 \quad \text{e} \quad \lim_{h_i \to 0^-} h_i(\mathbf{x}) > 0$$

se mantêm, então um deslocamento em modo deslizante ocorre sobre esta superfície (ver [13, s. 4.3]).

Os sistemas dinâmicos com lado direito descontínuo, assim como a estabilidade deles, foram inicialmente estudados em [28] e abordados em uma abundante bibliografia. Por exemplo, em [68] e [8] condições de estabilidade são estudadas, em [59] estas condições são aplicadas no controle de um robô manipulador. [32], [78] e [86] utilizam sistemas a estrutura variável na resolução de problemas de otimização convexa. [23] faz uma profunda análise das condições de existência e unicidade das trajetórias geradas por este tipo de sistemas em diversas situações.

Em particular, para o sistema $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$, com lado direito descontínuo, onde $\mathbf{f} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ é mensurável e essencialmente localmente limitada, define-se (ver [68,

def. 2.1] e [23, p. 18])

Definição 2.4.2 *Solução de Filippov*

$\mathbf{x}(t)$ *é solução da equação* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ *sobre* $[t_0, t_1]$ *se* $\mathbf{x}(.)$ *for absolutamente contínuo sobre* $[t_0, t_1]$ *e se para quase todo* $t \in [t_0, t_1]$, $\dot{\mathbf{x}} \in \mathcal{F}[\mathbf{f}](\mathbf{x}, t)$, *onde*

$$\mathcal{F}[\mathbf{f}](\mathbf{x}, t) \equiv \cap_{\delta>0} \cap_{\mu N=0} co\,\{\mathbf{f}(B(\mathbf{x}, \delta)\backslash N, t)\}$$

onde $\cap_{\mu N=0}$ *denota a interseção de todos os conjuntos* N *de medida zero, e co é o fecho convexo entre vetores.*

Se o sistema for autônomo, isto é $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, *e a função* $\mathbf{f}(\mathbf{x})$ *for contínua por partes, a seguinte definição é equivalente ([23, p. 22])*

$$\mathcal{F}[\mathbf{f}](\mathbf{x}) \equiv co\{\lim_{i\to\infty} \mathbf{f}(\mathbf{x}_i) | \mathbf{x}_i \to \mathbf{x},\ \mathbf{x}_i \notin N\}$$

O problema nos sistemas descontínuos é que sobre a descontinuidade não está definido o gradiente da função, e portanto não se aplicam os conceitos de estabilidade fornecidos pelo método direto de Liapunov. Em [8] e [23] são apresentados os teoremas de Liapunov para o caso descontínuo, onde o gradiente é substituído pelo gradiente generalizado de Clarke para uma função não diferenciável $f(\mathbf{x}, t)$:

Definição 2.4.3 *Gradiente generalizado de Clarke*

Seja $f(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ *uma função localmente Lipschitz,*

$$\partial f(\mathbf{x}, t) = co\{lim_{i\to\infty} \nabla f(\mathbf{x}, t) | (\mathbf{x}_i, t_i) \to (\mathbf{x}, t),\ (\mathbf{x}_i, t_i) \notin \Omega_v\} \tag{2.23}$$

onde Ω_v *é um conjunto de medida zero sobre o qual o gradiente de* $f(\mathbf{x}, t)$ *não está definido.*

Observe-se que podemos escrever (2.22) como

$$D_{\mathbf{f}}\dot{\mathbf{x}} = -\text{sgn}(\mathbf{f}(\mathbf{x})) \quad \Rightarrow \quad \dot{\mathbf{f}}(\mathbf{x}) = -\text{sgn}(\mathbf{f}(\mathbf{x})) \tag{2.24}$$

e para um determinado valor inicial da variável de estado $\mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$,

$$\mathbf{f}(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}_0) - \text{sgn}(\mathbf{f}(\mathbf{x}))t \tag{2.25}$$

existindo portanto convergência em tempo finito. Aqui também deve-se observar que essa convergência pode não ser global, pois também estamos assumindo a inversibilidade do jacobiano apenas em $\mathcal{N}(\mathbf{x}^*)$, sendo também este algoritmo incapaz de atravessar as hipersuperfícies determinadas pelo locus de $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$.

3) Terceira escolha

$$\mathbf{u} := PD_{\mathbf{f}}^T\mathbf{r} \tag{2.26}$$

onde P é uma matriz positiva definida. Substituindo em (2.13),

$$\dot{V} = -\mathbf{r}^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathbf{r} \tag{2.27}$$

a qual é obviamente negativa semi-definida. A razão de não poder garantir aqui convergência assintótica, ao menos globalmente, é que este sistema pode se estacionar em um valor tal que $\mathbf{r} \neq \mathbf{0}$ se $\mathbf{r} = -\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_{\mathbf{f}}^T)$, dependendo portanto da função $\mathbf{f}(\mathbf{x}(t))$ e do valor inicial da variável de estado $\mathbf{x}(0) = \mathbf{x}_0$ a possibilidade do sistema convergir.

Exemplo 2.4.4 *Seja a função:*

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \sin x_1 \\ x_2 \end{bmatrix} \tag{2.28}$$

a qual apresentará zeros em $\mathbf{x}^* = [n\pi\ 0]^T \quad$ *para todo* $n \in \mathbb{N}$. *O jacobiano é*

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} \cos x_1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.29}$$

e portanto

$$D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \cos(x_1)\sin(x_1) \\ x_2 \end{bmatrix} \tag{2.30}$$

e para valores da variável $\mathbf{x} = [(2n+1)\frac{\pi}{2}\ \ 0]^T \quad$ *para todo* $n \in \mathbb{N}$, *o sistema se estacionará apesar do valor da função* $\mathbf{f}(\mathbf{x}) = [\pm 1\ 0]^T \neq \mathbf{0}$.

A dinâmica do sistema em malha fechada está dada por:

$$\dot{\mathbf{x}} = -PD_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) \tag{2.31}$$

Este algoritmo será chamado de jacobiano trasposto (CJT).

Observe-se que (2.31) pode ser escrito como

$$\dot{\mathbf{f}}(\mathbf{x}(t)) = -D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}) \tag{2.32}$$

Porém, apenas para determinados valores do jacobiano $D_{\mathbf{f}}$ o sistema convergirá exponencialmente a $\mathbf{r} = \mathbf{0}$.

4) Quarta escolha

$$\mathbf{u} := \text{sgn}(D_{\mathbf{f}}^T \mathbf{r}) \tag{2.33}$$

Substituindo em (2.13),

$$\dot{V} = -\mathbf{r}^T D_{\mathbf{f}} \text{sgn}(D_{\mathbf{f}}^T \mathbf{r}) = -\|D_{\mathbf{f}}^T \mathbf{r}\|_1 \tag{2.34}$$

a qual é negativa semi-definida. Aqui também não pode ser garantida a convergência global pois existe a possibilidade do sistema se estacionar em um valor $\mathbf{r} \neq \mathbf{0}$ se $\mathbf{r} = -\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_{\mathbf{f}}^T)$, dependendo portanto da função $\mathbf{f}(\mathbf{x}(t))$ e do valor inicial da variável de estado $\mathbf{x}(0) = \mathbf{x}_0$ a possibilidade do sistema convergir assintoticamente. Este algoritmo receberá o nome de jacobiano trasposto a estrutura variável (VJT). Este sistema é a estrutura variável também por possuir um lado direito descontínuo.

A dinâmica do sistema em malha fechada está dada por:

$$\dot{\mathbf{x}} = -\text{sgn}(D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})) \tag{2.35}$$

o qual implica

$$\dot{\mathbf{f}}(\mathbf{x}) = -D_{\mathbf{f}} \text{sgn}(D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})) \quad \Rightarrow \quad \mathbf{f}(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}_0) - \int_0^t D_{\mathbf{f}} \text{sgn}(D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x})) \partial t \tag{2.36}$$

Porém, a convergência em tempo finito depende do valor particular de $D_{\mathbf{f}}$. Observe-se que cada componente do vetor $\dot{\mathbf{x}}$ só pode adotar os valores $\{-1; 0; 1\}$, considerando $\text{sgn}(0) = 0$.

5) Quinta escolha

Existe também a possibilidade de mudar a função de Liapunov candidata. Por exemplo, escolhendo

$$W(\mathbf{r}) = \|\mathbf{r}\|_1 = \mathbf{r}^T \mathrm{sgn}(\mathbf{r}) \tag{2.37}$$

a qual é obviamente positiva definida, tomando o valor 0 apenas no ponto de equilíbrio $\mathbf{r}(\mathbf{x}) = \mathbf{0}$. A função *signum* é constante exceto na origem; portanto, formalmente, sua derivada é zero exceto na origem (ver [78]),

$$\dot{W}(\mathbf{r}) = \dot{\mathbf{r}}^T \mathrm{sgn}(\mathbf{r}) + \mathbf{r}^T \frac{\partial \mathrm{sgn}(\mathbf{r})}{\partial t} = \dot{\mathbf{r}}^T \mathrm{sgn}(\mathbf{r}) = \mathrm{sgn}^T(\mathbf{r})(-D_{\mathbf{f}}\mathbf{u}) \tag{2.38}$$

Pode-se escolher como variável de controle

$$\mathbf{u} := P D_{\mathbf{f}}^T \mathrm{sgn}(\mathbf{r}) \tag{2.39}$$

sendo P uma matriz positiva definida. Substituindo em (2.38),

$$\dot{W}(\mathbf{r}) = -\mathrm{sgn}^T(\mathbf{r}) D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathrm{sgn}(\mathbf{r}) \tag{2.40}$$

a qual é negativa semi-definida. Aqui também, a razão de não poder garantir convergência assintótica, ao menos globalmente, é que o sistema poderia se estacionar em pontos tais que $\mathrm{sgn}(\mathbf{f}(\mathbf{x})) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}))$. Este sistema é a estrutura variável por possuir também um lado direito descontínuo, e por tal razão recebe o nome de jacobiano trasposto a estrutura variável (JTV). Cabe mencionar que a diferença deste sistema com o proposto em [13, c. 2], é que aqui admitimos um ganho P positivo definido, não mencionado nessa bibliografia.

A dinâmica do sistema em malha fechada está dada por:

$$\dot{\mathbf{x}} = -P D_{\mathbf{f}}^T \mathrm{sgn}(\mathbf{f}(\mathbf{x})) \tag{2.41}$$

o qual implica

$$\dot{\mathbf{f}}(\mathbf{x}) = -D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathrm{sgn}(\mathbf{f}(\mathbf{x})) \quad \Rightarrow \quad \mathbf{f}(\mathbf{x}(t)) = \mathbf{f}(\mathbf{x}_0) - \int_0^t D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathrm{sgn}(\mathbf{f}(\mathbf{x})) \partial t \tag{2.42}$$

Porém, a convergência em tempo finito depende do valor particular de $D_{\mathbf{f}}$.

Este método apresenta similaridades com o método de gradiente descendente (ou *steepest descent*, ver [79, c. 5]) pois, para o caso particular $P = I$:

$$\dot{\mathbf{x}} = -\sum_{i=1}^{n} \nabla f_i(\mathbf{x}) \text{sgn}(f_i(\mathbf{x}))$$

e portanto a trajetória $\mathbf{x}(t)$ se desenvolverá ao longo de linhas de fluxo determinada pela soma dos gradientes, corregidos pelos sinais de cada componente da função.

Neste ponto cabe destacar a relação entre os algoritmos apresentados, escolhidos segundo a metodologia de CLF, com a metodologia LOC já mencionada na seção anterior. Seja $\mathcal{U} = \{\mathbf{u} | \|\mathbf{u}\|_\infty \leq 1\}$, esta consiste na escolha de uma lei de controle $\mathbf{u}(\mathbf{r}) \in \mathcal{U}$ que faça a derivada temporal da função de Liapunov candidata $\dot{V}$ tão negativa quanto possível. Por exemplo, dada uma função escalar $g = -\mathbf{a}^T\mathbf{b}$ tal que $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, é evidente que o valor do vetor $\mathbf{b}$, de norma 1, que minimiza g para qualquer valor do vetor $\mathbf{a}$ é dado por $\mathbf{b} = \text{sgn}(\mathbf{a})$, que implica em $g = -\mathbf{a}^T\text{sgn}(\mathbf{a}) = -\|\mathbf{a}\|_1$. Assim, dada a função (2.12), $\dot{V} = -\mathbf{r}^T D_\mathbf{f} \mathbf{u}(\mathbf{r})$, é evidente que a escolha de $\mathbf{u} \in \mathcal{U}$ que minimiza $\dot{V}$ para qualquer valor de $\mathbf{r}$ e $D_\mathbf{f}$ é dado por $\mathbf{u} = \text{sgn}(D_\mathbf{f}^T \mathbf{r})$, que corresponde à escolha VJT de lei de controle.

Porém, a escolha da planta é arbitrária, o qual nos permite escolher um outro sistema como planta, por exemplo

$$\dot{\mathbf{x}} = D_\mathbf{f}^{-1} \mathbf{u} \tag{2.43}$$

a qual nos leva ao sistema $\dot{\mathbf{r}} = -D_\mathbf{f}\dot{\mathbf{x}} = -\mathbf{u}$. Mantendo a mesma função de Liapunov candidata (2.11),

$$\dot{V} = -\mathbf{r}^T D_\mathbf{f} \dot{\mathbf{x}} = -\mathbf{r}^T D_\mathbf{f} D_\mathbf{f}^{-1} \mathbf{u} = -\mathbf{r}^T \mathbf{u} \tag{2.44}$$

a qual nos leva à conclusão que a escolha ótima de $\mathbf{u} \in \mathcal{U}$ que minimiza $\dot{V}$ para qualquer valor de $\mathbf{r}$ é dado por $\mathbf{u} = \text{sgn}(\mathbf{r})$, e portanto $\dot{\mathbf{x}} = D_\mathbf{f}^{-1}\text{sgn}(\mathbf{r})$, que corresponde ao algoritmo NV. Portanto, esta lei de controle pode ser vista como a função que otimiza a dinâmica do sistema em malha fechada (LOC), dependendo apenas da escolha particular da planta.

A escolha $\mathbf{u} = \mathbf{r}$ em (2.44), nos leva a $\dot{V} = -\mathbf{r}^T\mathbf{r} = -\|\mathbf{r}\|_2^2$, e ao sistema em malha

fechada $\dot{\mathbf{x}} = D_{\mathbf{f}}^{-1}\mathbf{r}$ que corresponde ao método de Newton CN, com ganho $P = I$. Esta escolha é ótima no sentido de ser a escolha mais simples que conduz a um sistema exponencialmente estável em coordenadas $\mathbf{r}$:

$$\dot{\mathbf{r}} = -\mathbf{r} \tag{2.45}$$

Similarmente, pode se considerar a planta

$$\dot{\mathbf{x}} = D_{\mathbf{f}}^{T}\mathbf{u} \tag{2.46}$$

e mantendo a equação de saída (2.4) e usando a função de Liapunov candidata de norma 1 (2.37), obtem-se $\dot{\mathbf{r}} = -D_{\mathbf{f}}\dot{\mathbf{x}} = -D_{\mathbf{f}}D_{\mathbf{f}}^{T}\mathbf{u}$ e consequentemente

$$\dot{W} = \mathrm{sgn}^{T}(\mathbf{r})\dot{\mathbf{r}} = -\mathrm{sgn}^{T}(\mathbf{r})D_{\mathbf{f}}D_{\mathbf{f}}^{T}\mathbf{u} \tag{2.47}$$

A escolha $\mathbf{u} = \mathrm{sgn}(\mathbf{r})$ é claramente uma possível escolha LOC. O sistema em malha fechada resulta em:

$$\dot{\mathbf{x}} = D_{\mathbf{f}}^{T}\mathrm{sgn}(\mathbf{r})$$

que corresponde à escolha do algoritmo JTV com ganho $P = I$.

Se for utilizada a norma quadrática (2.11) como função de Liapunov candidata, então $\dot{V} = \mathbf{r}^{T}\dot{\mathbf{r}} = -\mathbf{r}^{T}D_{\mathbf{f}}D_{\mathbf{f}}^{T}\mathbf{u}$. A escolha $\mathbf{u} = \mathbf{r}$, com esta mesma planta, nos conduziria a $\dot{V} = -\|D_{\mathbf{f}}^{T}\mathbf{r}\|_2^2$ e ao sistema em malha fechada $\dot{\mathbf{x}} = D_{\mathbf{f}}^{T}\mathbf{r}$. Este sistema corresponde à escolha do algoritmo CJT com ganho $P = I$.

Resumindo, a metodologia CLF/LOC se mostrou capaz de derivar 5 algoritmos de primeira ordem destinados a achar zeros de funções vetoriais. Estes algoritmos são apresentados resumidamente na seguinte tabela.

$\mathbf{u}(\mathbf{r})$	$\dot{V}$ (linhas 1-4) $\dot{W}$ (linha 5)	**algoritmos contínuos**	nome	número de equação
$D_{\mathbf{f}}^{-1}P\mathbf{r}$	$-\mathbf{r}^T P\mathbf{r}$	$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}P\mathbf{f}(\mathbf{x})$	CN	(2.15)
$D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{r})$	$-\|\mathbf{r}\|_1$	$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{f}(\mathbf{x}))$	NV	(2.20)
$PD_{\mathbf{f}}^T\mathbf{r}$	$-\mathbf{r}^T D_{\mathbf{f}} PD_{\mathbf{f}}^T\mathbf{r}$	$\dot{\mathbf{x}} = -PD_{\mathbf{f}}^T\mathbf{f}(\mathbf{x})$	CJT	(2.26)
$\text{sgn}(D_{\mathbf{f}}^T\mathbf{r})$	$-\|D_{\mathbf{f}}^T\mathbf{r}\|_1$	$\dot{\mathbf{x}} = -\text{sgn}(D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}))$	VJT	(2.33)
$PD_{\mathbf{f}}^T\text{sgn}(\mathbf{r})$	$-\text{sgn}^T(\mathbf{r})D_{\mathbf{f}}PD_{\mathbf{f}}^T\text{sgn}(\mathbf{r})$	$\dot{\mathbf{x}} = -PD_{\mathbf{f}}^T\text{sgn}(\mathbf{f}(\mathbf{x}))$	JTV	(2.39)

Tabela 2.1 Algoritmos contínuos de primeira ordem derivados com a metodologia CLF/LOC

2.4.1 Considerações sobre o algoritmo de Newton

A equação (2.19), demonstra a convergência exponencial das linhas de fluxo determinadas pelas trajetórias do algoritmo contínuo.

Outra forma de analisar a convergência exponencial é através do método direto de Liapunov. Escolhendo como função de Liapunov candidata

$$V(\mathbf{x}) = \tfrac{1}{2}\mathbf{f}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) \quad \text{positiva definida}$$
$$\dot{V}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})\dot{\mathbf{f}}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x})D_{\mathbf{f}}(\mathbf{x})\dot{\mathbf{x}} = -\mathbf{f}^T D_{\mathbf{f}} D_{\mathbf{f}}^{-1}\mathbf{f} = -\|\mathbf{f}(\mathbf{x})\|^2 \quad \text{negativa definida}$$
$$\Rightarrow\ \dot{V}(\mathbf{x}) = -2V(\mathbf{x})\ \Rightarrow\ V(\mathbf{x}(t)) = V(0)e^{-2t}$$

a partir do qual também se conclui a convergência exponencial.

Observe-se também que ([41, lema 2.1]) este resultado indica

$$\frac{\mathbf{f}(\mathbf{x}(t))}{\|\mathbf{f}(\mathbf{x}(t))\|} = \frac{\mathbf{f}(\mathbf{x}(0))}{\|\mathbf{f}(\mathbf{x}(0))\|} = \text{ constante} \tag{2.48}$$

e portanto a trajetória $\mathbf{x}(t)$ estará determinada ao longo de linhas de fluxo onde a direção do vetor $\mathbf{f}(\mathbf{x})$ é constante e sua norma decresce exponencialmente no tempo.

Gomulka ([34]) utiliza este resultado com uma função de duas componentes $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})\ f_2(\mathbf{x})]^T \in \mathbb{R}^2$, e define

$$\psi(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \quad \in \mathbb{R} \tag{2.49}$$

Lema 2.4.5 *A função (2.49) permanece constante ao longo do tempo.*

Prova:

$$\mathbf{f}(t) = \mathbf{f}(0)e^{-t} \ \Rightarrow\ \frac{f_1(\mathbf{x}(t))}{f_2(\mathbf{x}(t))} = \frac{f_1(\mathbf{x}(0))}{f_2(\mathbf{x}(0))} = cte. \ \forall t > 0$$

Supondo $\mathbf{f}(\mathbf{x})$ duas vezes diferenciável, também são constantes seu gradiente e seu hessiano ao longo das trajetórias determinadas pelo algoritmo de Newton. Evidentemente, a existência desta função está limitada às regiões onde $f_2(\mathbf{x}) \neq 0$.

Hauser, em [41], observa, contradizendo a intuição, que se a função for perturbada apenas em partes da trajetória, gerando a função $\hat{\mathbf{f}}(\mathbf{x})$, a trajetória perturbada $\hat{\mathbf{x}}(t)$ se afastará da original apenas durante a perturbação, e coincidirão em aquelas regiões onde a função não é perturbada, mesmo que isto aconteça temporalmente depois da perturbação atuar.

Outra conclusão interessante ([41, teorema 3.1]), é que a direção limite das linhas de fluxo da trajetória, depende apenas da função nas posições iniciais $\mathbf{x}_0$ e final $\mathbf{x}^*$, independentemente do percurso da trajetória, isto é:

$$\lim_{t\to\infty} e^t \dot{\mathbf{x}}(t) = -D_{\mathbf{f}}^{-1}(\mathbf{x}^*)\mathbf{f}(\mathbf{x}_0) \tag{2.50}$$

Dedieu ([24]) estuda a convergência do método nos casos sobredeterminado, quando a dimensão do domínio da função $\mathbf{f}(\mathbf{x})$ é maior que a dimensão do codomínio, e indeterminado (caso contrário); outros teoremas de convergência foram apresentados em

[57].

Finalmente, cabe mencionar a existência dos métodos conhecidos como quase-Newton. O método de Newton inverte a matriz jacobiana, procedimento que pode se encarecer do ponto de vista computacional quando aumenta o número de variáveis ou quando esta matriz é mal condicionada. Por outro lado, os métodos quase-Newton apenas estimam uma matriz inversa do jacobiano, sem afetar a convergência das trajetórias ([17]). Tais métodos não foram implementados no presente trabalho.

2.4.2 Simulações dos algoritmos de primeira ordem contínuos

Em seguida, serão apresentadas, a modo de ilustração, simulações realizadas com diversas funções de teste. As funções foram escolhidas de duas variáveis a fim de poderem ser graficadas. As simulações foram realizadas no aplicativo Simulink do Matlab 6, discretizando os algoritmos pelo método de Euler; o comprimento do passo é de 0.01s. constante e foi simulado um tempo de 10 segundos.

Em todos os algoritmos que admitem ganhos, estes foram escolhidos $P = I$.

2.4.2.1 Simulações dos algoritmos contínuos com a função de Camelback

A primeira função escolhida como teste será a função de Camelback, apresentada em [18, p. 521], descrita pela primitiva:

$$\phi(x_1, x_2) = ax_1^2 + bx_1^4 + cx_1^6 - x_1x_2 + dx_2^2 + ex_2^4 \tag{2.51}$$

Foram escolhidas como constantes $a = -2,\ b = 1.05,\ c = -\frac{1}{6},\ d = -1$ e $e = 0$. Com estas constantes a função apresenta um máximo global, dois máximos locais, e duas selas. Estes pontos correspondem a zeros do gradiente da função primitiva, cuja expressão e valor para as constantes escolhidas estão dadas por:

$$\mathbf{f}(\mathbf{x}) = \nabla\phi(\mathbf{x}) = \begin{bmatrix} 2ax_1 + 4bx_1^3 + 6cx_1^5 - x_2 \\ -x_1 + 2dx_2 + 4ex_2^3 \end{bmatrix} = \begin{bmatrix} -4x_1 + 4.2x_1^3 - x_1^5 - x_2 \\ -x_1 - 2x_2 \end{bmatrix} \tag{2.52}$$

Os zeros desta função correspondem aos pontos $\mathbf{x} = [-1.7475\ 0.8737]^T$,
$\mathbf{x} = [-1.0705\ 0.5352]^T,\ \mathbf{x} = [0\ 0]^T,\ \mathbf{x} = [1.0705\ -0.5352]^T,\ \mathbf{x} = [1.7475\ -0.8737]^T$

O hessiano da função primitiva, e seu cálculo para as constantes escolhidas, são dados por:

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 2a + 12bx_1^2 + 30cx_1^4 & -1 \\ -1 & 2d + 12ex_2^2 \end{bmatrix} = \begin{bmatrix} -4 + 12.6x_1^2 - 5x_1^4 & -1 \\ -1 & -2 \end{bmatrix} \quad (2.53)$$

cujo determinante é dado pela função $\det(D_{\mathbf{f}}(\mathbf{x})) = 7 - 25.2x_1^2 + 10x_1^4$, o qual possui zeros ao longo das retas verticais definidas por $x_1 = 1.7737$, $x_1 = 0.6739$, $x_1 = -0.6739$, $x_1 = -1.7737$. Observe-se que, para as constantes escolhidas, $D_{\mathbf{f}}(\mathbf{x}) = D_{\mathbf{f}}(x_1)$.

Os pontos iniciais a partir dos quais foram testados os cinco algoritmos contínuos são $\mathbf{x}_0 = [-1 \ -1.5]^T$, $\mathbf{x}_0 = [0 \ 1.5]^T$, $\mathbf{x}_0 = [2 \ 1]^T$.

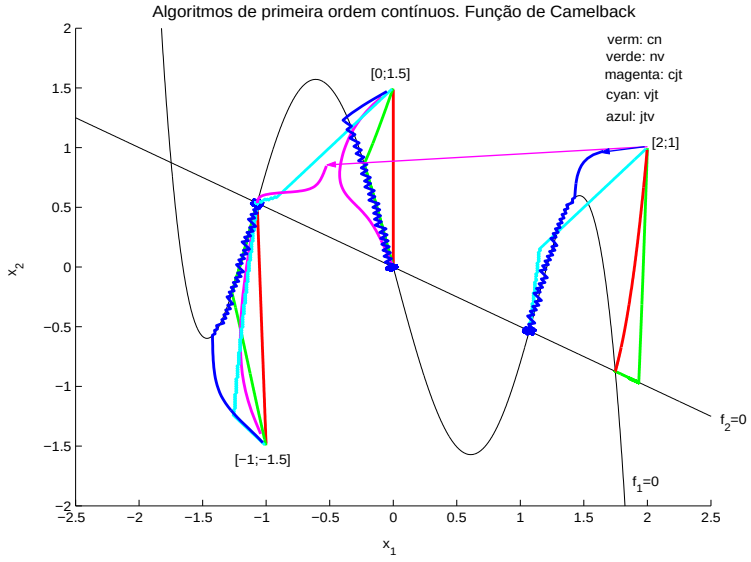

Figura 2.3: Simulações dos algoritmos contínuos para a função de Camelback mostrando a convergência das trajetórias

Observa-se aqui que todos os algoritmos foram capazes de achar algum zero a partir dos pontos iniciais testados, embora partindo do mesmo ponto inicial nem todos convergiram ao mesmo zero. Observa-se também que os algoritmos CJT e JTV apresentam um pulo a partir do ponto inicial $\mathbf{x}_0 = [2 \ 1]^T$ até $\mathbf{x}_1$, representados pelas setas das cores correspondentes. É visível, também, o comportamento em modos deslizantes dos algoritmos a estrutura variável, notadamente o JTV.

2.4.2.2 Simulações dos algoritmos contínuos com a função de Rosenbrock

A função de Rosenbrock ([18, p. 512]) está definida pela primitiva:

$$\phi(x_1, x_2) = a(x_1^2 - x_2)^2 + (x_1 - b)^2 \tag{2.54}$$

As constantes escolhidas são $a = 0.5$ e $b = 1$. Com estas constantes, a função apresenta um mínimo global em $x_1 = x_2 = 1$, que corresponde a um único zero do gradiente. A expressão deste e seu valor para as constantes escolhidas estão dadas por:

$$\mathbf{f}(\mathbf{x}) = \nabla\phi(\mathbf{x}) = \begin{bmatrix} 4ax_1(x_1^2 - x_2) + 2(x_1 - b) \\ -2a(x_1^2 - x_2) \end{bmatrix} = \begin{bmatrix} 2x_1(x_1^2 - x_2) + 2(x_1 - 1) \\ -(x_1^2 - x_2) \end{bmatrix} \tag{2.55}$$

O hessiano da função primitiva, e seu cálculo para as constantes escolhidas, são dados por:

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 12ax_1^2 - 4ax_2 + 2 & -4ax_1 \\ -4ax_1 & 2a \end{bmatrix} = \begin{bmatrix} 6x_1^2 - 2x_2 + 2 & -2x_1 \\ -2x_1 & 1 \end{bmatrix} \tag{2.56}$$

cujo determinante é $\det(D_{\mathbf{f}}(\mathbf{x})) = 8a^2x_1^2 - 8a^2x_2 + 4a = 2x_1^2 - 2x_2 + 2$, o qual possui raízes determinadas pelo locus $x_2 = x_1^2 + 1$.

Os pontos iniciais a partir dos quais foram testados os cinco algoritmos contínuos foram $\mathbf{x}_0 = [-0.5\ 2.5]^T$, $\mathbf{x}_0 = [0.5\ 2]^T$, $\mathbf{x}_0 = [1.5\ -0.5]^T$.

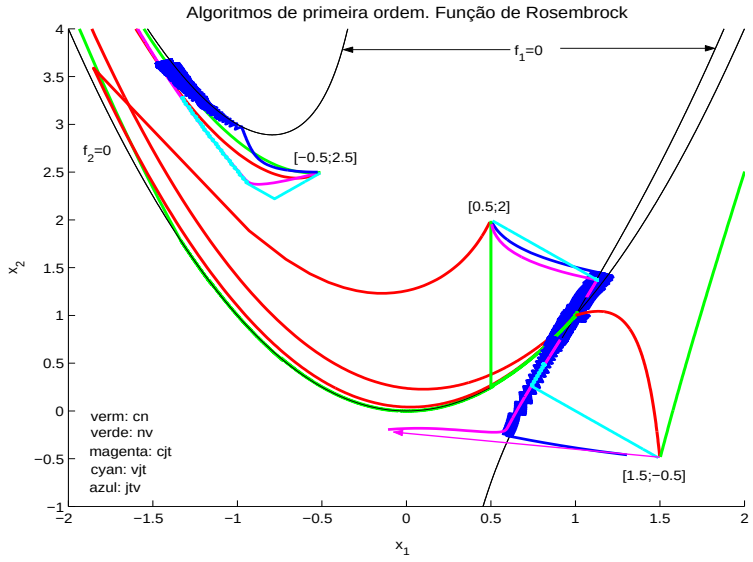

Figura 2.4: Simulações dos algoritmos contínuos para a função de Rosenbrock

A partir deste gráfico podemos chegar a algumas conclusões de ordem qualitativas. Por exemplo, observa-se claramente que a partir do ponto inicial $\mathbf{x}_0 = [-0.5\ 2.5]^T$ apenas os algoritmos CN e NV conseguiram atingir o zero da função. O algoritmo NV não conseguiu convergir a partir do ponto inicial $\mathbf{x}_0 = [1.5\ -0.5]^T$. O algoritmo CJT aqui também apresenta um pulo desse mesmo ponto inicial até o ponto achado na primeira iteração $\mathbf{x}_1$, representado por uma seta da cor correspondente. Observa-se claramente o comportamento em modos deslizantes das trajetórias dos algoritmos de estrutura variável, principalmente do JTV. A razão pela qual os algoritmos CN e NV conseguiram atravessar o locus da função $\det(D_{\mathbf{f}}(\mathbf{x})) = x_1^2 + 1 - x_2 = 0$, deve-se à discretização do algoritmo pelo método de Euler. Se estes forem perfeitamente contínuos, não estariam definidos sobre esta parábola. Uma outra observação interessante é o comportamento dos algoritmos a partir do ponto inicial $\mathbf{x}_0 = [0.5\ 2]^T$: por enquanto o algoritmo de Newton acompanha a forma espacial da função, a qual está sugerida pelos locus $f_i = 0$, os outros não são afetados por esta, inclusive, a trajetória gerada pelo NV percorre uma linha reta até o locus $f_2 = 0$, para seguir uma trajetória em modos deslizantes por esta curva até o zero. Este comportamento deve-se à presença de uma das chamadas singularidades estranhas, tema que será abordado numa seção posterior.

2.4.2.3 Simulações dos algoritmos contínuos com a função de Branin

A função apresentada por Branin em [18, p. 510], está definida como uma função vetorial de duas variáveis, e portanto não corresponde com o gradiente de uma função primitiva escalar. A função define-se:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} a - bx_2 + c\sin(dx_2) - x_1 \\ x_2 - e\sin(fx_1) \end{bmatrix} \tag{2.57}$$

As constantes utilizadas são $a = 1,\ b = 2,\ d = 4\pi,\ e = 0.5,\ f = 2\pi$. Inicialmente, a constante c é escolhida com valor 0, o qual faz de f_1 uma função linear. Com estas constantes a função apresenta 5 zeros. A derivada da função e seu valor para as constantes escolhidas são:

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} -1 & -b + cd\cos(dx_2) \\ -ef\cos(fx_1) & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 \\ -\pi\cos(2\pi x_1) & 1 \end{bmatrix} \tag{2.58}$$

Observe-se que, para as constantes escolhidas, $D_{\mathbf{f}}(\mathbf{x}) = D_{\mathbf{f}}(x_1)$.

Analisando a função de Branin, $\det(D_{\mathbf{f}}(\mathbf{x})) = -1 - 2\pi\cos(2\pi x_1)$, o qual é igual a zero ao longo das retas verticais definidas por $x_1 = \pm 0.2754 \pm k$, para todo $k \in \mathbb{N}$.

Os algoritmos serão testados a partir dos pontos iniciais
$\mathbf{x}_0 = [0.25 \ -0.4]^T, \quad \mathbf{x}_0 = [0.75 \ 0.6]^T, \quad \mathbf{x}_0 = [1.25 \ -0.6]^T$.

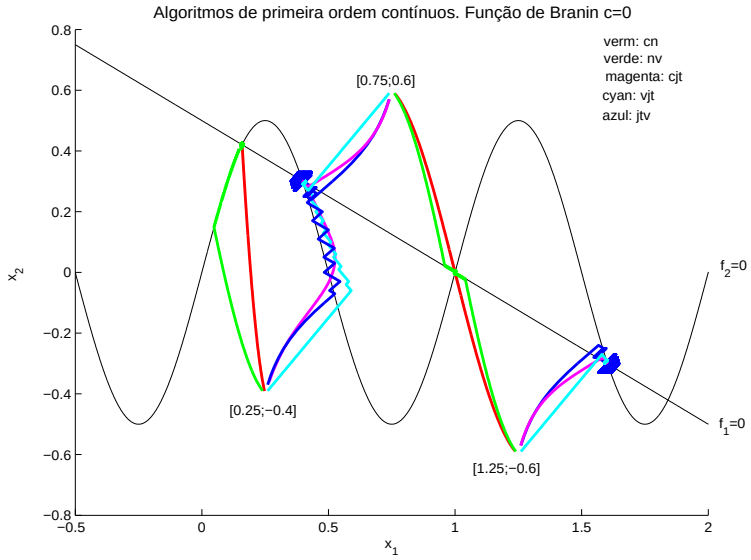

Figura 2.5: Simulações dos algoritmos contínuos para a função de Branin com $c = 0$

Observa-se que todos os algoritmos conseguiram convergir a algum zero a partir dos três pontos iniciais testados, embora em alguns casos achem zeros diferentes mesmo partindo de um ponto inicial comum. Pode se observar também a trajetória em modos deslizantes descrita pelos algoritmos a estrutura variável, notadamente o JTV, que é o que apresenta maiores oscilações uma vez atingida uma curva descrita por um locus $f_i = 0$.

A continuação, será aumentado o valor da constante c para 0.13, e os algoritmos serão testados a partir dos mesmos pontos iniciais. Com estas constantes, a função de Branin continua apresentando 5 zeros. O lugar das raízes da equação $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$ continuam sendo as retas verticais localizadas em $x_1 = \pm 0.2754 \pm k$, para todo $k \in \mathbb{N}$, só que estas retas verticais agora se vêem moduladas horizontalmente. A amplitude desta modulação parece proporcional ao valor da constante c.

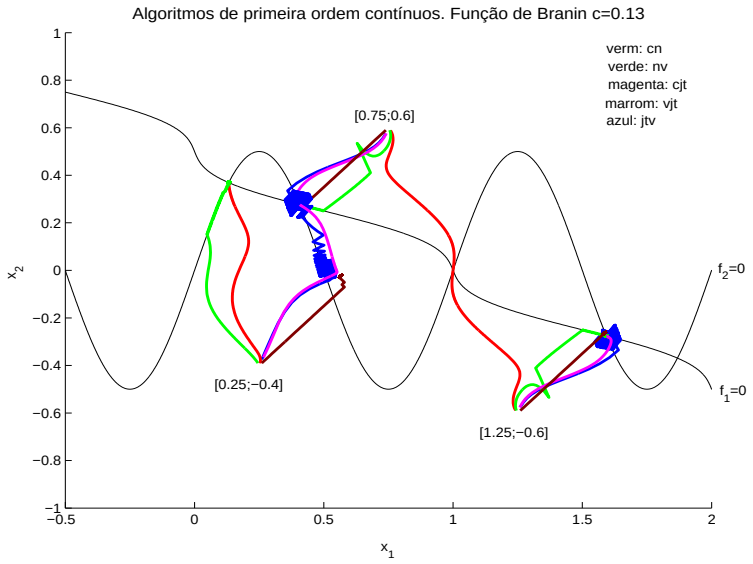

Figura 2.6: Simulações dos algoritmos contínuos para a função de Branin com $c = 0.13$

Aqui também nota-se que todos os algoritmos conseguiram achar um zero, embora em alguns casos convergiram para zeros diferentes mesmo partindo de um ponto inicial comum, como pode ser observado no gráfico.

2.5 Singularidades estranhas no algoritmo de Newton e de Branin

Nesta seção serão analisadas, de maneira qualitaiva, os chamados pontos singulares, que serão definidos adiante, e como estes afetam o algoritmo de Newton e a modificação a este realizada por Branin ([18]).

Como foi analisado (equação 2.18, considerando $\alpha = 1$) o algoritmo de Newton pode ser expressado como:

$$\dot{\mathbf{f}}(\mathbf{x}(t)) = -\mathbf{f}(\mathbf{x}(t))$$

o qual é exponencialmente convergente, isto é, $\mathbf{f}(t) = \mathbf{f}(0)e^{-t}$, o que implica que a trajetória tende exponencialmente a um zero da função $\mathbf{f}(\mathbf{x})$ quando t tende a infinito.

Aplicando a regra da cadeia nesta equação:

$$\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}}\dot{\mathbf{x}} = -\mathbf{f}(t) \quad \Rightarrow \quad \dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(t)$$

O algoritmo de Newton encontra um ponto de equilíbrio estável em um zero da função, isto é, se existe um instante t_1 tal que: $\mathbf{f}(\mathbf{x}(t_1)) = \mathbf{0} \Rightarrow \dot{\mathbf{x}} \equiv \mathbf{0}\ \forall t > t_1$.

Branin ([18]), propõe a seguinte modificação ao algoritmo de Newton:

$$\dot{\mathbf{x}} = \pm D_{\mathbf{f}}^{-1}\mathbf{f}(\mathbf{x}) \tag{2.59}$$

onde o sinal muda após achar um zero ou após a trajetória atravessar a hipersuperfície determinada pelo locus da equação $\det(D_{\mathbf{f}}) = 0$.

O objetivo desta modificação seria, diante da presença de diversos zeros de $\mathbf{f}(\mathbf{x})$, achar todos eles sucessivamente.

Quando o algoritmo de Branin adota o sinal negativo, este coincide com o algoritmo de Newton, e a trajetória a partir de um ponto inicial determinado converge exponencialmente a um zero da função. Após a convergência, o algoritmo adota o sinal positivo, o que instabiliza o algoritmo ($\dot{V}$ é positiva definida). A trajetória, portanto, se afasta do zero achado. O algoritmo retorna ao sinal negativo depois de atravessar a hipersuperfície determinada pelo locus de $\det(D_{\mathbf{f}}) = 0$, ponto a partir do qual a trajetória converge para um outro zero.

Porém, Zufiria e Guttalu ([87]), apresentam um contraexemplo onde tal conjectura não se verifica. Treccani ([77]), apresenta outro contraexemplo onde um único zero não pode ser achado pelo algoritmo. Zufiria e Guttalu apontam, também, que para que este raciocínio seja bem sucedido devem existir erros numéricos, devido a que se o algoritmo se estacionar exatamente em um zero em um instante t_1, mesmo que a mudança no sinal torne este ponto de equilíbrio instável, $\dot{\mathbf{x}} \equiv \mathbf{0}$ para todo $t > t_1$. Hardy ([39]), também contesta esta conjectura de Branin e propõe algumas modificações a este algoritmo para atingir o objetivo de alcançar todos os zeros da função sucessivamente.

2.5.1 Pontos singulares

A equação (2.17) (com ganho $P = I$) pode ser expressada como:

$$\dot{\mathbf{x}} = -\frac{\mathrm{adj}(D_{\mathbf{f}})}{\det(D_{\mathbf{f}})}\mathbf{f}(\mathbf{x}) \tag{2.60}$$

Obviamente, esta equação não está determinada nos pontos sobre a hipersuperfície

$\det(D_{\mathbf{f}}) = 0$, porém, a seguinte equação continua se verificando:

$$\det(D_{\mathbf{f}})\dot{\mathbf{x}} = -\text{adj}(D_{\mathbf{f}})\mathbf{f}(\mathbf{x}) \tag{2.61}$$

Gomulka ([34]) analisa as trajetórias do algoritmo

$$\dot{\mathbf{x}} = -\text{adj}(D_{\mathbf{f}})\mathbf{f}(\mathbf{x})$$

afirmando que o determinante é apenas um fator escalar que não muda a forma das linhas de fluxo das trajetórias. Evidentemente, o sinal deste fator altera o sentido das trajetórias.

Branin ([18]), classifica os pontos singulares do algoritmo como:

- Singularidades essenciais

  São os ponto onde $\text{adj}(D_{\mathbf{f}})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Correspondem aos zeros da função.

- Singularidades estranhas

  São os pontos onde $\text{adj}(D_{\mathbf{f}})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. Esses pontos implicam que $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(\text{adj}(D_{\mathbf{f}}))$, sendo portanto $\text{adj}(D_{\mathbf{f}})$ singular, o que implica que esses pontos devem estar localizados sobre a hipersuperfície determinada pelo locus da equação $\det(D_{\mathbf{f}}) = 0$.

Branin ([18]) e Gomulka ([34]) analisam as conseqüências da existência destas singularidades estranhas; porém, eles não esgotam todas as possíveis implicações. Por exemplo, Branin estabelece dois conjecturas:

1. As singularidades estranhas seriam selas ou nós instáveis.

2. Na ausência de singularidades estranhas, o algoritmo (2.59) seria convergente a todos os zeros da função $\mathbf{f}(\mathbf{x})$.

Zufiria e Guttalu ([87]) apresentam contraexemplos para ambas conjecturas. A primeira conjectura está baseada na linearização do algoritmo, tema que será tratado na próxima subseção. A segunda conjectura está baseada na suposição que cada zero estará encerrado dentro de uma hipersuperfície determinada pela equação $\det(D_{\mathbf{f}}) = 0$,

e portanto separado dos outros zeros por esta hipersuperfície. Tal conjectura pode não se verificar em todas as funções, por exemplo, em funções trigonométricas.

Exemplo 2.5.1 *([87])*

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} e^{x_1} cos x_2 - 1 \\ e^{x_1} sen x_2 \end{bmatrix} : \mathbb{R}^2 \to \mathbb{R}^2$$

o qual apresenta zeros em $[0\ 2n\pi]^T$ *e onde o determinante da matriz jacobiana* $\det(D_{\mathbf{f}}) = e^{2x_1} > 0 \ \ \forall \mathbf{x} \in \mathbb{R}^2$.

2.5.2 Linearização do algoritmo de Newton

Para analisar o comportamento das linhas de fluxo determinadas pelas trajetórias nas proximidades de uma singularidade estranha, o algoritmo pode ser linearizado ao redor deste ponto singular. Os autovalores da matriz linearizada determinarão o tipo de comportamento das trajetórias.

Gomulka ([34]), apresenta a seguinte linearização para o algoritmo de Newton.

Seja $\mathbf{x}_0$ uma singularidade estranha, isto é, $\text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))\mathbf{f}(\mathbf{x}_0) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}_0) \neq \mathbf{0}$, linearizando ao redor deste ponto:

$$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x}) \ \Rightarrow \ \det(D_{\mathbf{f}}(\mathbf{x}))\dot{\mathbf{x}} = -\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) := \mathbf{g}(\mathbf{x}) \tag{2.62}$$

$$\mathbf{g}(\mathbf{x}) \simeq \mathbf{g}(\mathbf{x}_0) + \frac{\partial \mathbf{g}}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) \quad \text{por ser } \mathbf{g}(\mathbf{x}_0) = \mathbf{0}$$

$$\begin{gathered} \tfrac{\partial \mathbf{g}}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0} = \tfrac{\partial}{\partial \mathbf{x}}(-\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}))_{\lfloor \mathbf{x}=\mathbf{x}_0} = \\ -\tfrac{\partial\ \text{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0}\mathbf{f}(\mathbf{x}_0) - \text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))D_{\mathbf{f}}(\mathbf{x}_0) = \\ -\tfrac{\partial}{\partial \mathbf{x}}\text{adj}(D_{\mathbf{f}}(\mathbf{x}))_{\lfloor \mathbf{x}=\mathbf{x}_0}\mathbf{f}(\mathbf{x}_0) - \det(D_{\mathbf{f}}(\mathbf{x}_0))D_{\mathbf{f}}^{-1}(\mathbf{x}_0)D_{\mathbf{f}}(\mathbf{x}_0) \end{gathered}$$

$$\Rightarrow \quad \mathbf{g}(\mathbf{x}) \simeq \left(-\frac{\partial\ \text{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0}\mathbf{f}(\mathbf{x}_0) - I\det(D_{\mathbf{f}}(\mathbf{x}_0)) \right)(\mathbf{x} - \mathbf{x}_0)$$

mas a singularidade estranha deve estar localizada sobre a hipersuperfície determinada por $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$, portanto

$$\Rightarrow \quad \mathbf{g}(\mathbf{x}) \simeq -\frac{\partial\ \text{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}}_{\lfloor \mathbf{x}=\mathbf{x}_0}\mathbf{f}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

Cabe destacar que Gomulka ([34]) não esclarece como operacionalizar a derivada da matriz.

Bhaya e Kaszkurewicz ([13]) apresentam a seguinte linearização:

$$\mathbf{g}(\mathbf{x}) := -\text{ adj }(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \det(D_{\mathbf{f}}(\mathbf{x}))\dot{\mathbf{x}}$$

$$\begin{aligned}
\Rightarrow\ \mathbf{g}(\mathbf{x}) &\simeq \mathbf{g}(\mathbf{x}_0) - \frac{\partial\ \text{adj}_{(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x})}}{\partial \mathbf{x}}{\rfloor}_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x}-\mathbf{x}_0) \\
&= -\frac{\partial\ \text{adj}_{(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x})}}{\partial \mathbf{x}}{\rfloor}_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x}-\mathbf{x}_0) \\
&= \left[-\frac{\partial\ \text{adj}_{(D_f(\mathbf{x}))}}{\partial \mathbf{x}}{\rfloor}_{\mathbf{x}=\mathbf{x}_0} F(\mathbf{x}_0) - \text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))D_{\mathbf{f}}(\mathbf{x}_0)\right](\mathbf{x}-\mathbf{x}_0) \\
&= -\frac{\partial\ \text{adj}_{(D_{\mathbf{f}}(\mathbf{x}))}}{\partial \mathbf{x}}{\rfloor}_{\mathbf{x}=\mathbf{x}_0} F(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0)
\end{aligned} \tag{2.63}$$

por ser $\text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))D_{\mathbf{f}}(\mathbf{x}_0) = \det(D_{\mathbf{f}}(\mathbf{x}_0))I = 0$ e onde, dada uma matriz M, chamamos:

$$\frac{\partial M}{\partial \mathbf{x}}F := \sum_{i=1}^{n}\frac{\partial M}{\partial x_i}F_i \quad \text{e} \quad F_i := [0... \underbrace{\mathbf{f}}_{\text{coluna i}}...0] \in \mathbb{R}^{n\times n}$$

e no caso da equação (2.63), $M = \text{adj}(D_{\mathbf{f}}(\mathbf{x}))$.

A seguinte equação também se verifica ([13]):

$$\dot{\mathbf{x}} = -\left(\frac{\partial D_{\mathbf{f}}^{-1}(\mathbf{x})}{\partial \mathbf{x}}F{\rfloor}_{\mathbf{x}=\mathbf{x}_0} + I\right)(\mathbf{x}-\mathbf{x}_0) \tag{2.64}$$

2.5.2.1 Algoritmo linearizado para $n = 2$

Por simplicidade, partindo de

$$\mathbf{g}(\mathbf{x}) := \det(D_{\mathbf{f}}(\mathbf{x}))\dot{\mathbf{x}} \simeq -\frac{\partial\ \text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}(\mathbf{x}-\mathbf{x}_0) = -\frac{\partial\ \text{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}}{\rfloor}_{\mathbf{x}=\mathbf{x}_0} F(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0)$$

e para $n = 2$:

$$\text{adj}(D_{\mathbf{f}}(\mathbf{x})) = \begin{bmatrix} \frac{\partial f_2}{\partial x_2} & -\frac{\partial f_1}{\partial x_2} \\ -\frac{\partial f_2}{\partial x_1} & \frac{\partial f_1}{\partial x_1} \end{bmatrix}$$

$$\Rightarrow \quad \frac{\partial\ \mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}} F(\mathbf{x}) = \sum_{i=1}^{2} \frac{\partial\ \mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))}{\partial \mathbf{x}} F_i(\mathbf{x}) =$$

$$\begin{bmatrix} \frac{\partial^2 f_2}{\partial x_1 \partial x_2} & -\frac{\partial^2 f_1}{\partial x_2 \partial x_1} \\ -\frac{\partial^2 f_2}{\partial x_1^2} & \frac{\partial^2 f_1}{\partial x_1^2} \end{bmatrix} \begin{bmatrix} f_1 & 0 \\ f_2 & 0 \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 f_2}{\partial x_2^2} & -\frac{\partial^2 f_1}{\partial x_2^2} \\ -\frac{\partial^2 f_2}{\partial x_1 \partial x_2} & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} \end{bmatrix} \begin{bmatrix} 0 & f_1 \\ 0 & f_2 \end{bmatrix} =$$

$$\begin{bmatrix} f_1 \frac{\partial^2 f_2}{\partial x_1 \partial x_2} - f_2 \frac{\partial^2 f_1}{\partial x_1 \partial x_2} & f_1 \frac{\partial^2 f_2}{\partial x_2^2} - f_2 \frac{\partial^2 f_1}{\partial x_2^2} \\ -f_1 \frac{\partial^2 f_2}{\partial x_1^2} + f_2 \frac{\partial^2 f_1}{\partial x_1^2} & -f_1 \frac{\partial^2 f_2}{\partial x_1 \partial x_2} + f_2 \frac{\partial^2 f_1}{\partial x_1 \partial x_2} \end{bmatrix} = M(\mathbf{x}) \tag{2.65}$$

$$\Rightarrow \quad \mathbf{g}(\mathbf{x}) \simeq -M(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

Cabe destacar que ao linearizar diretamente $\dot{\mathbf{x}} = -D_f^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})$ (como em [13]) se produz uma indeterminação pois a matriz $D_{\mathbf{f}}^{-1}(\mathbf{x})$ não está definida sobre as singularidades estranhas, nem sobre nenhum ponto sobre o locus de $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$.

Por exemplo, para $n = 2$, o algoritmo de Newton resulta:

$$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x}) = \begin{bmatrix} -\left(f_1 \frac{\partial f_2}{\partial x_2} - f_2 \frac{\partial f_1}{\partial x_2}\right)\left(\frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_2} - \frac{\partial f_1}{\partial x_2}\frac{\partial f_2}{\partial x_1}\right)^{-1} \\ -\left(-f_1 \frac{\partial f_2}{\partial x_1} + f_2 \frac{\partial f_1}{\partial x_1}\right)\left(\frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_2} - \frac{\partial f_1}{\partial x_2}\frac{\partial f_2}{\partial x_1}\right)^{-1} \end{bmatrix}$$

ao diferenciar (por qualquer método) o denominador se elevará ao quadrado nas componentes da matriz linearizada, o qual é igual a zero sobre a singularidade estranha.

2.5.2.2 Comparação entre duas formas de linearização

Bhaya e Kaszkurewicz ([13]), fornecem a expressão (2.63) para linearizar o algoritmo ao redor de um ponto. Mas também é possível linearizar diretamente o vetor $\mathbf{g}(\mathbf{x}) = -\mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x})$ ao redor da singularidade estranha $\mathbf{x}_0$, isto é:

$$\mathbf{g}(\mathbf{x}) \simeq \mathbf{g}(\mathbf{x}_0) + D_{\mathbf{g}}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) = D_{\mathbf{g}}(\mathbf{x})(\mathbf{x} - \mathbf{x}_0) \tag{2.66}$$

sendo $D_{\mathbf{g}}(\mathbf{x})$ o jacobiano do vetor $\mathbf{g}(\mathbf{x})$, chegando ao mesmo resultado.

Utilizando a expressão (2.66), genericamente para $n = 2$:

$$\mathbf{g}(\mathbf{x}) = -\mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} -f_1 \frac{\partial f_2}{\partial x_2} + f_2 \frac{\partial f_1}{\partial x_2} \\ f_1 \frac{\partial f_2}{\partial x_1} - f_2 \frac{\partial f_1}{\partial x_1} \end{bmatrix} \simeq D_{\mathbf{g}}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) =$$

$$\left[\begin{array}{cc} -\frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_2} - f_1\frac{\partial^2 f_2}{\partial x_1 \partial x_2} + \frac{\partial f_2}{\partial x_1}\frac{\partial f_1}{\partial x_2} + f_2\frac{\partial^2 f_1}{\partial x_1 \partial x_2} & -\frac{\partial f_1}{\partial x_2}\frac{\partial f_2}{\partial x_2} - f_1\frac{\partial^2 f_2}{\partial x_2^2} + \frac{\partial f_2}{\partial x_2}\frac{\partial f_1}{\partial x_2} + f_2\frac{\partial^2 f_1}{\partial x_2^2} \\ \frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_1} + f_1\frac{\partial^2 f_2}{\partial x_1^2} - \frac{\partial f_2}{\partial x_1}\frac{\partial f_1}{\partial x_1} - f_2\frac{\partial^2 f_1}{\partial x_1^2} & \frac{\partial f_1}{\partial x_2}\frac{\partial f_2}{\partial x_1} + f_1\frac{\partial^2 f_2}{\partial x_1 \partial x_2} - \frac{\partial f_2}{\partial x_2}\frac{\partial f_1}{\partial x_1} - f_2\frac{\partial^2 f_1}{\partial x_1 \partial x_2} \end{array}\right]_{\mathbf{x}_0} (\mathbf{x}-\mathbf{x}_0)$$

$$= \left[\begin{array}{cc} -f_1\frac{\partial^2 f_2}{\partial x_1 \partial x_2} + f_2\frac{\partial^2 f_1}{\partial x_1 \partial x_2} & -f_1\frac{\partial^2 f_2}{\partial x_2^2} + f_2\frac{\partial^2 f_1}{\partial x_2^2} \\ f_1\frac{\partial^2 f_2}{\partial x_1^2} - f_2\frac{\partial^2 f_1}{\partial x_1^2} & f_1\frac{\partial^2 f_2}{\partial x_1 \partial x_2} - f_2\frac{\partial^2 f_1}{\partial x_1 \partial x_2} \end{array}\right]_{\mathbf{x}=\mathbf{x}_0} (\mathbf{x} - \mathbf{x}_0) \qquad (2.67)$$

onde utilizamos o fato que, sobre a singularidade estranha $\det(D_{\mathbf{f}}(\mathbf{x}_0)) = \frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_2} - \frac{\partial f_2}{\partial x_1}\frac{\partial f_1}{\partial x_2} = 0$.

Observe-se que o resultado coincide com (2.65), sendo portanto as duas formas de linearização equivalentes.

2.5.3 Exemplos de singularidades estranhas

A seguir, serão apresentados dois exemplos ilustrativos do comportamento das singularidades estranhas para os algoritmos de Newton e Branin.

Exemplo 2.5.2 *([87])*

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1^2 + 4x_1x_2 + x_2^2 \\ x_2 + 1 \end{bmatrix} \quad \textit{a qual apresenta zeros em } \mathbf{x}^* = \begin{bmatrix} 2 \pm \sqrt{3} \\ -1 \end{bmatrix}$$

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 2x_1 + 4x_2 & 4x_1 + 2x_2 \\ 0 & 1 \end{bmatrix}$$

$$\Rightarrow \ \det(D_{\mathbf{f}}(\mathbf{x})) = 2x_1 + 4x_2 \ \Rightarrow \ \det(D_{\mathbf{f}}) = 0 \Leftrightarrow x_2 = -\frac{x_1}{2}$$

$$\mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1^2 + 4x_1x_2 + x_2^2 - (x_2+1)(4x_1 + 2x_2) \\ (x_2+1)(2x_1 + 4x_2) \end{bmatrix}$$

$$\Rightarrow \ \mathbf{x}_0 = \mathbf{0} \ \textit{é singularidade estranha}$$

$$M(\mathbf{x}) = \begin{bmatrix} -(x_2+1)4 & -(x_2+1)2 \\ (x_2+1)2 & (x_2+1)4 \end{bmatrix}$$

$$\Rightarrow \quad \mathbf{g}(\mathbf{x}) \simeq -M(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) = -\begin{bmatrix} -4 & -2 \\ 2 & 4 \end{bmatrix}\mathbf{x}$$

com autovalores $\lambda = \pm 2\sqrt{3}$. *Portanto a singularidade estranha na origem é uma sela com autovetores normalizados* $\mathbf{v}_1 = \left[\frac{1}{2\sqrt{2-\sqrt{3}}} \quad -\frac{\sqrt{2-\sqrt{3}}}{2}\right]^T$ *e* $\mathbf{v}_2 = \left[\frac{1}{2\sqrt{2+\sqrt{3}}} \quad -\frac{\sqrt{2+\sqrt{3}}}{2}\right]^T$.

Na seguinte figura mostra-se um diagrama da localização dos zeros e da singularidade.

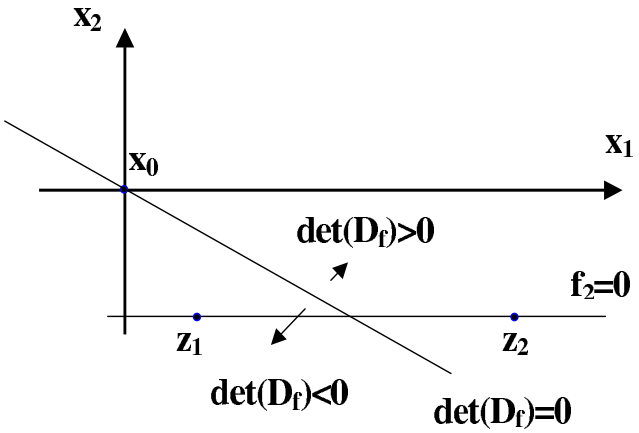

Figura 2.7: Localização dos pontos singulares para o exemplo 2.5.2

Exemplo 2.5.3

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1 x_2 \\ x_2 - 1 \end{bmatrix} : \mathbb{R}^2 \to \mathbb{R}^2 \quad \Rightarrow \ \mathbf{x}^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} x_2 & x_1 \\ 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \det(D_{\mathbf{f}}(\mathbf{x})) = x_2$$

$$\mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & -x_1 \\ 0 & x_2 \end{bmatrix}\begin{bmatrix} x_1 x_2 \\ x_2 - 1 \end{bmatrix} = \begin{bmatrix} x_1 x_2 - x_1(x_2-1) \\ x_2(x_2-1) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2(x_2-1) \end{bmatrix}$$

$$\Rightarrow \ \textit{o algoritmo apresenta pontos singulares em } \mathbf{x}_s = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix} ; \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

onde $\mathbf{x}^* = [0\ 1]^T$ *é um zero da função e* $\mathbf{x}_0 = [0\ 0]^T$ *é uma singularidade estranha.*

$$\mathbf{g}(\mathbf{x}_0) \simeq -\frac{\partial M}{\partial \mathbf{x}}_{\rfloor \mathbf{x}=\mathbf{0}} F(\mathbf{0})\mathbf{x} = -\begin{bmatrix} -(x_2-1) & 0 \\ 0 & (x_2-1) \end{bmatrix}_{\rfloor \mathbf{x}=\mathbf{0}} \mathbf{x} = -\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{x}$$

Os autovalores são $\lambda = \pm 1$*, portanto a singularidade estranha na origem é uma sela, de autovetores* $\mathbf{v}_1 = [1\ 0]^T$ *e* $\mathbf{v}_2 = [0\ 1]^T$*, respectivamente.*

O eixo de atração da sela está sobre o locus $\det(D_{\mathbf{f}}(\mathbf{x})) = x_2 = 0$ *e o algoritmo de Newton resulta neste caso:*

$$\dot{\mathbf{x}} = \begin{bmatrix} -\frac{x_1}{x_2} \\ -(x_2-1) \end{bmatrix} \tag{2.68}$$

Na figura seguinte é mostrada a localização das singularidades, assim como uma possível percurso das trajetórias para o algoritmo de Newton.

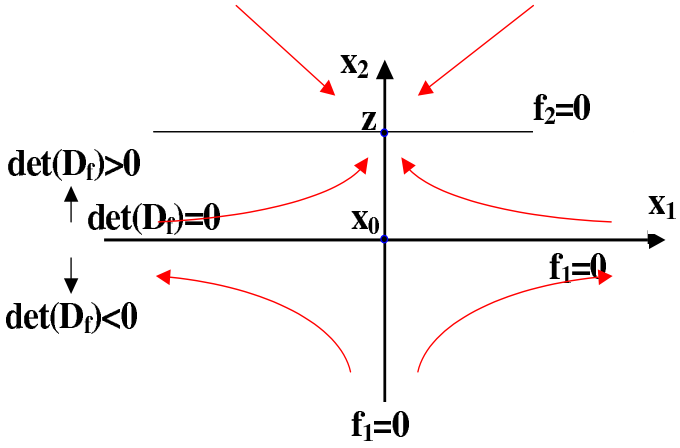

Figura 2.8: Localização das singularidades singulares para o exemplo 2.5.3 mostrando em vermelho as linhas de fluxo das trajetórias

Observe-se que estas trajetórias tem um comportamento similar àquele apresentado pelo algoritmo de Newton utilizado na função de Rosenbrock, cuja implementação foi apresentada na seção anterior. Este comportamento está caracterizado da seguinte maneira: diante da presença de uma singularidade estranha que se comporta como uma sela, as trajetórias se fazem assíntotas à curva determinada pelo locus $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$, para depois retornar pelo semiespaço onde se encontra o zero. Este retorno só é possível devido à discretização do algoritmo, pois teoricamente as trajetórias contínuas não podem atravessar essa curva sobre a qual a função não está definida. Essas linhas de fluxo assíntotas ao eixo horizontal estão determinadas por um locus tal que a di-

reção do vetor $\mathbf{f}(\mathbf{x})$ permanece constante ([41, lema 2.1]), por enquanto sua norma vai diminuindo exponencialmente com o tempo.

Analisando as linhas de fluxo das trajetórias de (2.68), se observa que, quanto mais próximas do locus determinado pela equação $D_{\mathbf{f}}(\mathbf{x}) = x_2 = 0$, o vetor da lei de atualização das variáveis deveria aumentar em norma. Efetivamente, isto acontece no caso da primeira componente do vetor, $\dot{x}_1$. Mas quando a trajetória se aproxima da singularidade estranha, esta norma vai se aproximando da unidade. Isto significa que pode se esperar grandes pulos de trajetória perto da hipersuperfície $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$, exceto nas proximidades das singularidades estranhas, onde o comportamento é mais imprevisível. Quanto mais complexa for a função, evidentemente mais difícil é predeterminar o percurso das linhas de fluxo. Simulações realizadas com a função de Branin partindo de pontos próximos às singularidades estranhas mostraram que podem se produzir grandes pulos (o que é esperável também nas proximidades de $\det(D_{\mathbf{f}}) = 0$) mas também oscilações temporais.

Analisando a equação (2.68) para o algoritmo de Branin, $\dot{\mathbf{x}} = \pm D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})$, observa-se o seguinte comportamento: se a trajetória começar em um ponto inicial tal que $x_2 < 0$ (por exemplo em $\mathbf{x} = [0 \ -1]^T$), e com sinal negativo, a trajetória tenderá ao zero da função. Porém, quando o algoritmo discretizado atravessar o locus $\det(D_{\mathbf{f}}) = 0$ ($x_2 = 0$), muda o sinal, virando positivo; se $x_2 < 1$ (por exemplo em $\mathbf{x} = [0 \ 1/2]^T$), então $\dot{x}_2 < 0$ e portanto a trajetória retornará à singularidade estranha, apresentando um comportamento oscilatório ao redor desta (por exemplo, esta oscilação pode se apresentar sobre o eixo $x_1 = 0$).

Conclusão: pela primeira conjectura de Branin, por ser as singularidades estranhas selas ou nós instáveis, seriam pontos de repulsão das trajetórias. Porém, neste exemplo fica claro que a singularidade estranha pode ser um ponto de atração para o algoritmo de Branin, dependendo do comprimento do passo de iteração.

Hardy ([39]), propõe uma modificação ao algoritmo de Branin (2.59) destinada a evitar este problema:

$$\dot{\mathbf{x}} = -(-1)^k \text{sgn}(\det(D_{\mathbf{f}}(\mathbf{x})))D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{2.69}$$

onde k é o número de zero a ser localizado. Neste caso, a mudança no sinal de $\det(D_{\mathbf{f}}(\mathbf{x})) = x_2$ ao atravessar o eixo horizontal mantém a direção ascendente das linhas de fluxo, atingindo o zero procurado.

Branin menciona uma função apresentada por Brent, para a qual o algoritmo tem uma zona de não convergência (exatamente para trajetórias começando dentro do locus $\det(D_{\mathbf{f}}) = 0$). Porém, as singularidades estranhas não são pontos de atração nesse caso.

2.5.4 Outras considerações sobre singularidades estranhas

Cabe destacar que o algoritmo de Branin ([18]), assim como o estudo realizado por Gomulka ([34]), é normalizado, isto é, é analisado $\frac{\partial \mathbf{x}}{\partial s}$.

Considerando que

$$\partial s = \sqrt{\partial x_1^2 + \partial x_2^2 + ... + \partial x_n^2} \quad \Rightarrow \quad \frac{\partial s}{\partial t} = \sqrt{\left(\frac{\partial x_1}{\partial t}\right)^2 + ... + \left(\frac{\partial x_n}{\partial t}\right)^2} = \|\dot{\mathbf{x}}\|$$

$$\Rightarrow \quad \frac{\partial \mathbf{x}}{\partial s} = \frac{\partial \mathbf{x}}{\partial t}\frac{\partial t}{\partial s} = \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} = \frac{-D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})}{\|D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x})\|}$$

o que não altera a análise das trajetórias do algoritmo.

É sabido que se o posto de $D_{\mathbf{f}}(\mathbf{x})$ é $n-2$ ou menor, a matriz adjunta de $D_{\mathbf{f}}$ é uma matriz nula. Para uma singularidade estranha $\mathbf{x}_0$:

$$\text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))\mathbf{f}(\mathbf{x}_0) = \mathbf{0} \quad \text{para} \quad \mathbf{f}(\mathbf{x}_0) \neq \mathbf{0}$$

$$\text{portanto, posto adj } (D_{\mathbf{f}}(\mathbf{x}_0)) < n \;\Leftrightarrow\; \text{posto } (D_{\mathbf{f}}(\mathbf{x}_0)) < n$$

sendo portanto a matriz jacobiana e sua adjunta singulares sobre a singularidade estranha.

Se posto $(D_{\mathbf{f}}(\mathbf{x}_0)) = n-1$, então posto adj$(D_{\mathbf{f}}(\mathbf{x}_0)) = 1$, portanto o espaço nulo, ao qual pertence $\mathbf{f}(\mathbf{x}_0)$, está determinado por um vetor constante no espaço. Portanto, existe um vetor $\mathbf{l} \neq \mathbf{0}$ tal que $\mathbf{l}^T D_{\mathbf{f}}(\mathbf{x}_0) = \mathbf{0}$, e um vetor $\mathbf{r} \neq \mathbf{0}$ tal que $D_{\mathbf{f}}^T(\mathbf{x}_0)\mathbf{r} = \mathbf{0}$; $\mathbf{l},\ \mathbf{r} \in \mathbb{R}^n$. Como esses vetores não são únicos (poderiam ser multiplicados por um escalar que continuariam estando nos espaços nulos de $D_{\mathbf{f}}(\mathbf{x}_0)$ e

$D_{\mathbf{f}}^T(\mathbf{x}_0)$, respectivamente), existe uma adequada escolha desses vetores tal que

$$\text{adj}(D_{\mathbf{f}}(\mathbf{x}_0)) = \mathbf{r}\mathbf{l}^T$$

$$\Rightarrow \quad \text{adj}(D_{\mathbf{f}}(\mathbf{x}_0))\mathbf{f}(\mathbf{x}_0) = \mathbf{r}\mathbf{l}^T\mathbf{f}(\mathbf{x}_0) = \mathbf{0} \;\Leftrightarrow\; \mathbf{l}^T\mathbf{r} = 0 \tag{2.70}$$

A equação (2.70), junto com $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$ determinam a presença de singularidades estranhas.

Gomulka ([34]) afirma que, para $n > 2$, as singularidades estranhas usualmente não são pontos isolados. Isto também pode ocorrer com dimensão 2, como será mostrado no seguinte exemplo.

Exemplo 2.5.4

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_1^2 x_2 \\ x_2 - 1 \end{bmatrix} : \mathbb{R}^2 \to \mathbb{R}^2 \quad \textit{zero em } \mathbf{x}^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 2x_1x_2 & x_1^2 \\ 0 & 1 \end{bmatrix}$$

$$\det(D_{\mathbf{f}}(\mathbf{x})) = 2x_1x_2 \;\Rightarrow\; \det(D_{\mathbf{f}}(\mathbf{x})) = 0 \;\Leftrightarrow\; x_1 = 0 \textit{ ou } x_2 = 0$$

$$\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 1 & -x_1^2 \\ 0 & 2x_1x_2 \end{bmatrix} \begin{bmatrix} x_1^2 x_2 \\ x_2 - 1 \end{bmatrix} = \begin{bmatrix} x_1^2 \\ 2x_1x_2(x_2 - 1) \end{bmatrix}$$

Portanto existem singularidades estranhas ao longo de todo o eixo vertical excluindo o próprio zero: $\mathbf{x}_0 = \left\{\mathbf{x} = [0\ k]^T\ \forall k \in \mathbb{R}\right\} - \left\{[0\ 1]^T\right\}$.

Gomulka ([34]), propõe a utilização da função (2.49) para analisar as singularidades estranhas em uma função de duas componentes ($\mathbf{f}(\mathbf{x}) \in \mathbb{R}^2$), a qual também é utilizada em ([87]).

O gradiente da função é dado por:

$$\nabla\psi(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}\frac{1}{f_2} - \frac{f_1}{f_2^2}\frac{\partial f_2}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2}\frac{1}{f_2} - \frac{f_2}{f_2^2}\frac{\partial f_2}{\partial x_2} \end{bmatrix} = \frac{1}{f_2^2}\begin{bmatrix} f_2\frac{\partial f_1}{\partial x_1} - f_1\frac{\partial f_2}{\partial x_1} \\ f_2\frac{\partial f_1}{\partial x_2} - f_1\frac{\partial f_2}{\partial x_2} \end{bmatrix} \tag{2.71}$$

Observe-se que, assumindo $f_2(\mathbf{x}) \neq 0$, este gradiente é zero se e somente se $\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Portanto as selas e pontos extremos desta função $\psi(\mathbf{x})$ correspondem a singularidades do algoritmo de Newton.

O hessiano da função $\psi(\mathbf{x})$, sobre uma singularidade estranha, é dado por:

$$\nabla^2\psi(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1^2} f_2 - \frac{\partial^2 f_2}{\partial x_1^2} f_1 & \frac{\partial^2 f_1}{\partial x_1 \partial x_2} f_2 - \frac{\partial^2 f_2}{\partial x_1 \partial x_2} f_1 \\ \frac{\partial^2 f_1}{\partial x_1 \partial x_2} f_2 - \frac{\partial^2 f_2}{\partial x_1 \partial x_2} f_1 & \frac{\partial^2 f_1}{\partial x_2^2} f_2 - \frac{\partial^2 f_2}{\partial x_2^2} f_1 \end{bmatrix} \frac{1}{f_2^2} \tag{2.72}$$

Comparando esta matriz com (2.67), observa-se que tem as mesmas componentes. Portanto, os autovalores desta matriz também determinarão o tipo de singularidade estranha.

Gomulka ([34, prop. 2]) utiliza esta matriz para concluir que toda singularidade estranha, sempre que $f_2 \neq 0$, só pode ser sela ou nó instável para $n = 2$, provando a primeira conjectura de Branin. Porém, Zufiria e Guttalu ([87]) apresentam um contraexemplo:

Exemplo 2.5.5 *([87])*

$$f_1 = ax_1^2 + 2bx_1x_2 + cx_2^2 + \mathcal{O}\|\cdot\|^m$$

onde f_2 é tal que $f_2(\mathbf{0}) \neq 0$ e $\frac{\partial f_2}{\partial x_1}\rfloor_{\mathbf{x}=\mathbf{0}} = 0$ e onde $\mathcal{O}\|\cdot\|^m$ são termos de ordem maior que 2 que inicialmente não serão considerados.

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 2ax_1 + 2bx_2 & 2bx_1 + 2cx_2 \\ 0 & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \Rightarrow D_{\mathbf{f}}(\mathbf{0}) = \begin{bmatrix} 0 & 0 \\ 0 & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_2}{\partial x_2} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_2}{\partial x_2}(ax_1^2 + 2bx_1x_2 + cx_2^2) \\ 0 \end{bmatrix}$$

Portanto $\mathbf{x} = \mathbf{0}$ é singularidade estranha.

$$\psi(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \quad \nabla\psi(\mathbf{x}) = \frac{1}{f_2^2} \begin{bmatrix} f_2(2ax_1 + 2bx_2) \\ f_2(2bx_1 + 2cx_2) - (ax_1^2 + 2bx_1x_2 + cx_2^2)\frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

$$\Rightarrow \quad \nabla\psi(\mathbf{0}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; para \; f_2 \neq 0$$

$$\nabla^2\psi(\mathbf{0}) = \frac{2}{f_2(\mathbf{0})} \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Os autovalores desta matriz são:

$$\lambda_{12} = \frac{2}{f_2}\left(\frac{a+c}{2} \pm \sqrt{\left(\frac{a+c}{2}\right)^2 - (ac - b^2)}\right)$$

onde se conclui que

i) se $b^2 > ac$, então $\lambda_1 > 0$ e $\lambda_2 < 0$, a singularidad estranha na origem é uma sela.

ii) se $b^2 < ac$, então $\lambda_1 > 0$ e $\lambda_2 > 0$, a origem é um nó instável.

iii) se $b^2 = ac$, então $\lambda_1 = 0$ e $\lambda_2 = \frac{2(a+c)}{f_2}$, onde nada pode se concluir e passam a influenciar os termos $\mathcal{O}\|\cdot\|^m$.

No caso iii), Zufiria e Guttalu exemplificam:

a) se $b \neq 0$ e $\mathcal{O}\|\cdot\|^m = 0$ não há bifurcação na origem e trajetórias simples atravessam a singularidade estranha.

b) se $\mathcal{O}\|\cdot\|^m = x_2^3$ a origem é um nó sela.

c) se $a = b = c = 0$ e $\mathcal{O}\|\cdot\|^m = (x_1^2 - 2x_2^2)(x_1^2 - \frac{1}{2}x_2^2)$ e $f_2 = x_2 + 1$, a origem é uma "sela" de oito braços, 4 de atração e 4 de repulsão das trajetórias.

2.5.5 Análise da singularidade estranha na função de Rosenbrock

A função de Rosenbrock (2.54), como foi antecipado, possui uma singularidade estranha para o algoritmo de Newton. Observe-se que

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 4ax_1(x_1^2 - x_2) + 2(x_1 - b) \\ -2a(x_1^2 - x_2) \end{bmatrix}$$

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 12ax_1^2 - 4ax_2 + 2 & -4ax_1 \\ -4ax_1 & 2a \end{bmatrix}$$

$$\Rightarrow \det(D_{\mathbf{f}}(\mathbf{x})) = 8a^2x_1^2 - 8a^2x_2 + 4a$$

Portanto, o locus de $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$ está dado pela equação

$$x_2 = x_1^2 + \frac{1}{2a}$$

que caracteriza a curva à qual se fazem assíntotas algumas trajetórias do algoritmo de Newton.

$$\begin{aligned}\mathbf{g}(\mathbf{x}) = \text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) &= \begin{bmatrix} 2a & 4ax_1 \\ 4ax_1 & 12ax_1^2 - 4ax_2 + 2 \end{bmatrix} \begin{bmatrix} 4ax_1(x_1^2 - x_2) + 2(x_1 - b) \\ -2a(x_1^2 - x_2) \end{bmatrix} \\ &= \begin{bmatrix} 4a(x_1 - b) \\ -4a(x_1^2 - x_2)(2ax_1^2 - 2ax_2 + 1) + 8ax_1(x_1 - b) \end{bmatrix}\end{aligned}$$

Este vetor apresentará zeros em:

$$\mathbf{x} \in \left\{ \begin{bmatrix} b \\ b^2 \end{bmatrix}; \begin{bmatrix} b \\ b^2 + \frac{1}{2a} \end{bmatrix} \right\}$$

sendo que o zero da função (correspondente a um mínimo da função primitiva (2.54)) estará em $\mathbf{x}^* = [b\ b^2]^T$, e a singularidade estranha em $\mathbf{x}_0 = [b\ \ b^2 + \frac{1}{2a}]^T$.

Para as constantes escolhidas nas simulações $a = \frac{1}{2}$ e $b = 1$, o zero está em $\mathbf{x}^* = [1\ 1]^T$ e a singularidade estranha em $\mathbf{x}_0 = [1\ 2]^T$.

Linearizando $\mathbf{g}(\mathbf{x})$ ao redor da singularidade estranha:

$$D_{\mathbf{g}}(\mathbf{x}) = \begin{bmatrix} 4a & 0 \\ -32ax_1^3 + 32a^2x_1x_2 + 8ax_1 - 8ab & 16a^2x_1^2 - 16a^2x_2 + 4a \end{bmatrix}$$

matriz que na singularidade estranha toma o valor

$$D_{\mathbf{g}}(\mathbf{x}_0) = \begin{bmatrix} 4a & 0 \\ 16ab & -4a \end{bmatrix}$$

a qual apresenta autovalores em $\lambda = \pm 4a$, sendo portanto uma sela de autovetores $\mathbf{v}_1 = [k\ \ 2bk]^T$ para todo $k \in \mathbb{R}$ e $\mathbf{v}_2 = [0\ k]^T$ para todo $k \in \mathbb{R}$, respectivamente.

Na figura seguinte ilustra-se o comportamento ds trajetórias para os algoritmos CN

e NV perto da singularidade estranha. A simulação foi implementada com o aplicativo Simulink do programa Matlab 6, discretizando por Euler com comprimento do passo fixo de 0.01s, e simulando durante 10 segundos.

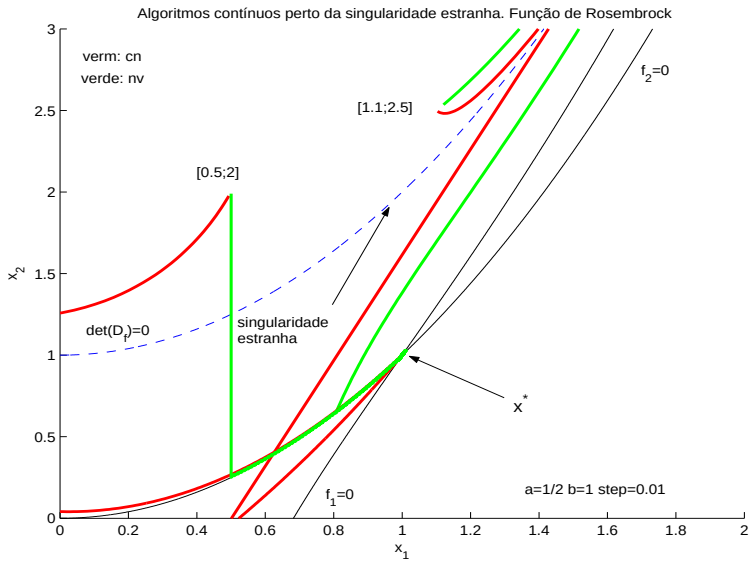

Figura 2.9: Simulações dos algoritmos contínuos CN e NV
para a função de Rosenbrock partindo próximo da singularidade estranha

2.5.6 Análise das singularidades estranhas na função de Branin

Na função de Branin (2.57) foram escolhidas as constantes $a = 1,\ b = 2,\ d = 4\pi,\ e = 0.5,\ f = 2\pi$. A constante c inicialmente foi escolhida com o valor 0.

O jacobiano da função foi calculado na equação (2.58), e foi observado que, para as constantes escolhidas, $\det(D_{\mathbf{f}}(\mathbf{x})) = \det(D_{\mathbf{f}}(x_1)) = -1 - 2\pi\cos(2\pi x_1)$, o qual é igual a zero ao longo das retas verticais definidas por $x_1 = \pm 0.2754 \pm k$, para todo $k \in \mathbb{N}$, e

$$\mathbf{g}(\mathbf{x}) = \mathrm{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} -1 & -2 \\ -\pi\cos(2\pi x_1) & 1 \end{bmatrix} \begin{bmatrix} 1 - 2x_2 - x_1 \\ x_2 - \frac{1}{2}\sin(2\pi x_1) \end{bmatrix} = \begin{bmatrix} 1 - x_1 - \sin(2\pi x_1) \\ \pi\cos(2\pi x_2)(1 - 2x_2 - x_1) - x_2 + \frac{1}{2}\sin(2\pi x_1) \end{bmatrix}$$

cujos zeros correspondem com os zeros da função, não existindo, portanto, singularidades estranhas.

Para a constante $c = 0.13$, o determinante do jacobiano é dado por $\det(D_{\mathbf{f}}) = -1 + \pi\cos(2\pi x_1)(-2 + 0.52\pi\cos(4\pi x_2))$. O locus $\det(D_{\mathbf{f}}) = 0$ serão as mesmas retas verticais que no caso da constante $c = 0$, mas levemente moduladas; a amplitude desta modulação é proporcional à constante c.

As singularidades estranhas, dentro da janela definida em $x_1 \in [-0.5; 2.5]$, $x_2 \in [-1; 1]$ se localizam em:

$\mathbf{x} = [0.2879\ -0.0923]^T$ $\mathbf{x} = [0.2767\ -0.8797]^T$ $\mathbf{x} = [1.3216\ -0.4466]^T$

$\mathbf{x} = [1.2933\ -0.9168]^T$ $\mathbf{x} = [1.7336\ -0.8154]^T$ $\mathbf{x} = [0.2810\ -0.6076]^T$

$\mathbf{x} = [0.6748\ 0.5525]^T$ $\mathbf{x} = [1.3254\ -0.5523]^T$ $\mathbf{x} = [1.7330\ -0.6721]^T$

$\mathbf{x} = [0.2777\ -0.3827]^T$ $\mathbf{x} = [1.7125\ 0.0920]^T$ $\mathbf{x} = [0.2809\ 0.1076]^T$

$\mathbf{x} = [1.7190\ -0.1076]^T$ $\mathbf{x} = [0.6779\ 0.4459]$ $\mathbf{x} = [1.7223\ 0.3827]^T$

$\mathbf{x} = [0.2670\ 0.6721]^T$ $\mathbf{x} = [1.7190\ 0.6076]^T$ $\mathbf{x} = [0.2664\ 0.8154]^T$

$\mathbf{x} = [0.7067\ 0.9168]^T$ $\mathbf{x} = [1.7233\ 0.8797]^T$

Na seguinte figura, pode se observar um locus das curvas descritas, onde os pontos marcados com x correspondem a singularidades estranhas do algoritmo de Newton.

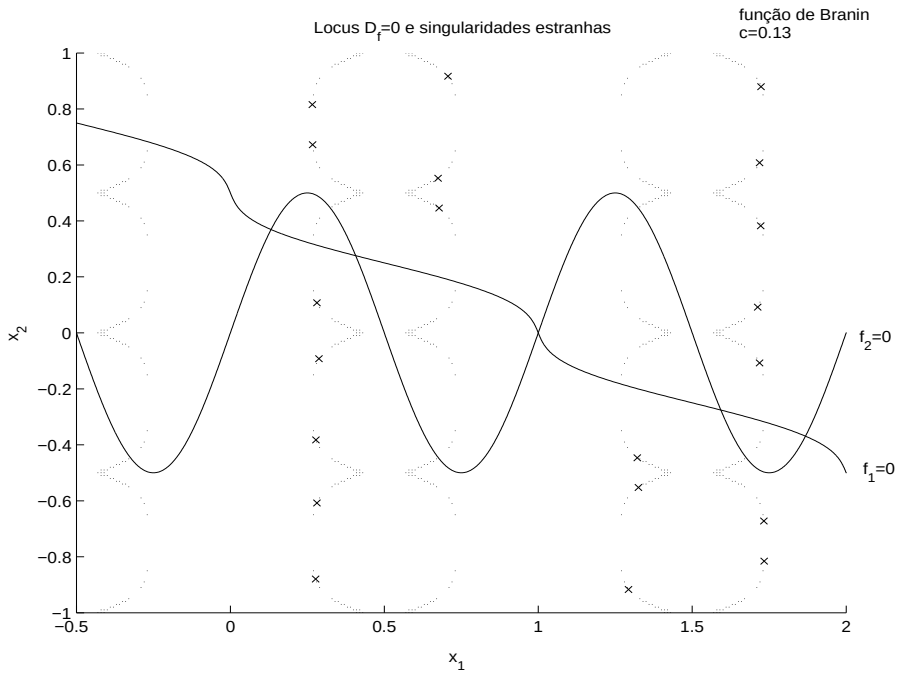

Figura 2.10: Lugar de raízes da equação $\det(D_{\mathbf{f}}) = 0$
e singularidades estranhas para o algoritmo de Newton $c = 0.13$

2.5.7 Análise das singularidades estranhas na função de Camelback

A função de Camelback é descrita pela primitiva (2.51), cujo gradiente está dado por (2.52) e seu hessiano por (2.53). Foram escolhidas como constantes $a = -2$, $b = 1.05$, $c = -\frac{1}{6}$, $d = -1$ e $e = 0$, com cujos valores a função apresenta um máximo global, dois máximos locais, e duas selas.

Como foi observado, o determinante do gradiente é $\det(D_{\mathbf{f}}(\mathbf{x})) = (2a + 12bx_1^2 + 30cx_1^4)(2d + 12ex_2^2) - 1$, que para as constantes escolhidas é igual a $\det(D_{\mathbf{f}}(\mathbf{x})) = 7 - 25.2x_1^2 + 10x_1^4$, o qual é igual a zero ao longo das retas verticais determinadas por $x_1 = \pm 1.7737$ e $x_1 = \pm 0.6739$.

$$\mathbf{g}(\mathbf{x}) = \text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x}) = \begin{bmatrix} -2 & 1 \\ 1 & -4 + 12.6x_1^2 - 5x_1^4 \end{bmatrix} \begin{bmatrix} -4x_1 + 4.2x_1^3 - x_1^5 - x_2 \\ -x_1 - 2x_2 \end{bmatrix} = \begin{bmatrix} 7x_1 - 8.4x_1^3 + 2x_1^5 \\ -8.4x_1^3 + 4x_1^5 + 7x_2 - 25.2x_1^2x_2 + 10x_1^4x_2 \end{bmatrix}$$

Este vetor só apresenta zeros nos pontos $\mathbf{x} = [-1.7475\ 0.8737]^T$,
$\mathbf{x} = [-1.0705\ 0.5352]^T$, $\mathbf{x} = [0\ 0]^T$, $\mathbf{x} = [1.0705\ -0.5352]^T$,
$\mathbf{x} = [1.7475\ -0.8737]^T$. Todos estes pontos correspondem a zeros da função, não existindo, portanto, singularidades estranhas com as constantes escolhidas.

2.5.8 Singularidades estranhas para o algoritmo de Newton a estrutura variável

O algoritmo de Newton a estrutura variável (NV) apresentado na equação (2.20), também pode apresentar singularidades estranhas, definindo-se estas de maneira diferente que no caso do algoritmo de Newton. Define-se o algoritmo

$$\dot{\mathbf{x}} = -D_{\mathbf{f}}^{-1}(\mathbf{x})\text{sgn}(\mathbf{f}(\mathbf{x}))$$

Naqueles pontos onde $\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\text{sgn}(\mathbf{f}(\mathbf{x})) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$, verifica-se que $\text{sgn}(\mathbf{f}(\mathbf{x})) \in \mathcal{N}(\text{adj}(D_{\mathbf{f}}(\mathbf{x})))$ e portanto esta matriz é não singular. Por não estar na presença de um zero da função, $\text{sgn}(\mathbf{f}(\mathbf{x})) \neq \mathbf{0}$ e portanto esses pontos comportam-se como singulari-

dades estranhas.

O algoritmo NV aplicado às funções de Rosenbrock, de Camelback, e de Branin com as constantes $c = 0$ e $c = 0.13$ não apresentou nenhuma singularidade estranha.

2.6 Algoritmos discretos

Nesta seção serão apresentados os algoritmos de primeira ordem discretizados. A discretização será realizada pelo método de Euler, interpretanto sempre os algoritmos discretos como sistemas de controle em malha fechada.

De (2.2), dada a variável de estado $\mathbf{x}_k$ definida como a variável na iteração k, podemos escrever:

$$\mathbf{r}_k := \mathbf{r}(\mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k) \tag{2.73}$$

Discretizando (2.3) por Euler (para mais informação sobre a discretização por Euler, ver [4] e [36]):

$$\Delta\mathbf{x} = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k \mathbf{u}_k \;\Rightarrow\; \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k \tag{2.74}$$

sendo α_k o comprimento do passo e $\mathbf{u}_k$ a variável de controle a ser escolhida na iteração k.

Assim:

$$\mathbf{f}(\mathbf{x}_{k+1}) = \mathbf{f}(\mathbf{x}_k + \alpha_k \mathbf{u}_k) \tag{2.75}$$

O resíduo, aproximando por Taylor até o termo de primeira ordem, resulta:

$$\mathbf{r}(\mathbf{x}_{k+1}) = \mathbf{r}(\mathbf{x}_k + \Delta\mathbf{x}) = \mathbf{r}(\mathbf{x}_k) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}\Delta\mathbf{x} + h.o.t \simeq \mathbf{r}_k + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}\Delta\mathbf{x} = \mathbf{r}_k - D_{\mathbf{f}}(\mathbf{x}_k)\Delta\mathbf{x} \tag{2.76}$$

Portanto

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k D_{\mathbf{f}} \mathbf{u}_k \tag{2.77}$$

chamando, por comodidade, $D_{\mathbf{f}} := D_{\mathbf{f}}(\mathbf{x}_k)$.

Assumindo a variável de controle como função do resíduo:

$$\mathbf{u}_k := \phi(\mathbf{r}_k) \tag{2.78}$$

$$\Rightarrow \quad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k D_{\mathbf{f}} \phi(\mathbf{r}_k) \tag{2.79}$$

Na seguinte figura, observamos o diagramas de blocos representativo do algoritmo discreto.

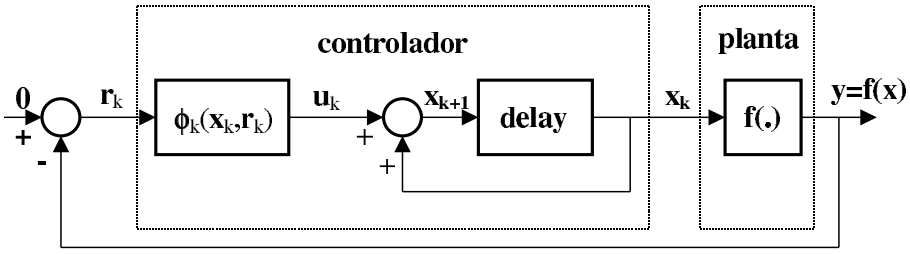

Figura 2.11: Diagrama de blocos representativo do sistema discreto

Neste ponto, estamos em condições de escolher uma função de Liapunov candidata discreta:

$$V_k := \mathbf{r}_k^T \mathbf{r}_k \tag{2.80}$$

$$\Rightarrow \quad \Delta V = V_{k+1} - V_k = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} - \mathbf{r}_k^T \mathbf{r}_k \tag{2.81}$$

e sustituindo com (2.77):

$$\begin{aligned} \Delta V &= (\mathbf{r}_k - \alpha_k D_\mathbf{f} \mathbf{u}_k)^T (\mathbf{r}_k - \alpha_k D_\mathbf{f} \mathbf{u}_k) - \mathbf{r}_k^T \mathbf{r}_k = \\ &= \alpha_k^2 \mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f} \mathbf{u}_k - 2\alpha_k \mathbf{r}_k^T D_\mathbf{f} \mathbf{u}_k \end{aligned} \tag{2.82}$$

Otimizando o comprimento do passo α_k, segundo a metodologia LOC, de maneira tal de minimizar ΔV, obtemos:

$$\frac{\partial \Delta V}{\partial \alpha_k} = 2\alpha_k \mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f} \mathbf{u}_k - 2\mathbf{r}_k^T D_\mathbf{f} \mathbf{u}_k = 0 \tag{2.83}$$

$$\Rightarrow \quad \alpha_k = \frac{\mathbf{r}_k^T D_\mathbf{f} \mathbf{u}_k}{\mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f} \mathbf{u}_k} \tag{2.84}$$

Observe-se que o numerador do comprimento do passo pode ser tanto positivo como negativo, mas para essa escolha de α_k:

$$\Delta V = -\frac{(\mathbf{r}_k^T D_\mathbf{f} \mathbf{u}_k)^2}{\mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f} \mathbf{u}_k} \tag{2.85}$$

o qual é sempre não positivo. Esta variação será igual a zero no ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$, embora existam outras possibilidades para estacionamento do algoritmo, isto é, $\Delta V = 0$, dependendo da escolha particular da variável de controle $\mathbf{u}_k$. Portanto, essa

escolha do comprimento do passo α_k faz a função de Liapunov V não crescente a cada iteração definida ao longo das trajetórias de ϕ_k, e uma adequada escolha da variável de controle $\mathbf{u}_k$ faz o algoritmo assintoticamente convergente, ao menos localmente.

Em seguida, será apresentada a versão discretizada dos algoritmos contínuos apresentados na seção anterior.

1) Primeira escolha

$$\mathbf{u}_k := D_{\mathbf{f}}^{-1} P \mathbf{r}_k \tag{2.86}$$

sendo P uma matriz simétrica positiva definida.

O comprimento do passo é dado por, substituindo em (2.84):

$$\alpha_k = \frac{\mathbf{r}_k^T P \mathbf{r}_k}{\mathbf{r}_k^T P^2 \mathbf{r}_k} \tag{2.87}$$

e a variável de estado em cada iteração será calculada, substituindo (2.86) em (2.74):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k D_{\mathbf{f}}^{-1} P \mathbf{f}(\mathbf{x}_k) \tag{2.88}$$

Observe-se que, para essa primeira escolha da variável de controle $\mathbf{u}_k$, o comprimento do passo é sempre positivo.

Para a escolha particular $P = I$, o comprimento do passo $\alpha_k = 1$ e o algoritmo coincide com o conhecido Newton-Raphson, razão pela qual este algoritmo será chamado Newton discreto (DN).

A variação da função de Liapunov é dada por, substituindo (2.86) em (2.85):

$$\Delta V = -\frac{(\mathbf{r}_k^T P \mathbf{r}_k)^2}{\mathbf{r}_k^T P^2 \mathbf{r}_k} \leq 0 \tag{2.89}$$

sendo portanto o algoritmo assintoticamente convergente, se estacionando apenas no ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$. Cabe destacar, também aqui, o caráter local dessa convergência, pois estamos sempre supondo que $\mathbf{x}_0 \in \mathcal{N}(\mathbf{x}^*)$, onde o jacobiano $D_{\mathbf{f}}(\mathbf{x}_k)$ é assumido não singular. Os pontos tais que $\text{adj}(D_{\mathbf{f}}(\mathbf{x}_k))\mathbf{f}(\mathbf{x}_k) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}_k) \neq \mathbf{0}$, os quais estão obviamente sobre o locus determinado pela hipersuperfície $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$, determinam as singularidades estranhas do algoritmo DN.

Observe-se que, para $P = I$,

$$\Delta V = V_{k+1} - V_k = -\mathbf{r}_k^T \mathbf{r}_k = -V_k \;\Rightarrow\; V_{k+1} = 0 \;\Rightarrow\; \mathbf{r}_{k+1} = \mathbf{0} \;\Rightarrow\; \mathbf{x}_{k+1} = \mathbf{x}^*$$

Evidentemente, isto é o que aconteceria caso a função $\mathbf{f}(\mathbf{x})$ seja linear, pois o resíduo foi linearizado ao redor do ponto $\mathbf{x}_k$, e o própio algoritmo de Newton pode ser visto como uma linearização da função ao redor do ponto $\mathbf{x}_k$ ([24], [43]), isto é:

$$\mathbf{f}(\mathbf{x}^*) \simeq \mathbf{f}(\mathbf{x}) + D_{\mathbf{f}}(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) \;\Rightarrow\; \mathbf{x}^* = \mathbf{x} - D_{\mathbf{f}}^{-1}(\mathbf{x})\mathbf{f}(\mathbf{x}) \tag{2.90}$$

O método de Newton apresenta convergência quadrática e tem uma excelente resposta quando a função e sua derivada são bem comportadas. A desvantagem deste método é que raízes de ordem alta podem causar uma taxa de convergência lenta, e a seqüência poderia apresentar pulos indesejados entre raízes ou tomar passos muitos grandes perto de uma singularidade ([43]).

2) Segunda escolha

$$\mathbf{u}_k := D_{\mathbf{f}}^{-1} P \mathrm{sgn}(\mathbf{r}_k) \tag{2.91}$$

sendo P uma matriz simétrica positiva definida. Com essa escolha da variável de controle, sustituindo em (2.84):

$$\alpha_k = \frac{\mathbf{r}_k^T P \mathrm{sgn}(\mathbf{r}_k)}{\mathrm{sgn}^T(\mathbf{r}_k) P^2 \mathrm{sgn}(\mathbf{r}_k)} \tag{2.92}$$

O comprimento do passo aqui pode resultar negativo, mas para essa escolha de $\mathbf{u}_k$, substituindo (2.91) em (2.85)

$$\Delta V = -\frac{(\mathbf{r}_k^T P \mathrm{sgn}(\mathbf{r}_k))^2}{\mathrm{sgn}^T(\mathbf{r}_k) P^2 \mathrm{sgn}(\mathbf{r}_k)} \leq 0 \tag{2.93}$$

o qual é sempre não positivo a menos do ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$, onde se estacionará o algoritmo, garantindo assim sua convergência assintótica.

A variável de estado em cada iteração será calculada, substituindo (2.91) em (2.74):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k D_{\mathbf{f}}^{-1} P \mathrm{sgn}(\mathbf{f}(\mathbf{x}_k)) \tag{2.94}$$

Este algoritmo coincide com a versão discretizada do Newton variável, a menos a diferença que aqui é admitido um ganho P, sem por isso afetar a convergência do algoritmo, tal como observado. Para a escolha particular $P = I$, a variável de controle $\mathbf{u}_k$ é a mesma que no caso contínuo e por essa razão este algoritmo será chamado de Newton variável discreto (DNV). Observe-se também que para esse ganho o comprimento do passo é positivo. Mais uma vez, destacamos o caráter local da convergência do algoritmo.

3) Terceira escolha

$$\mathbf{u}_k := PD_{\mathbf{f}}^T \mathbf{r}_k \tag{2.95}$$

onde P é uma matriz simétrica positiva definida.

Com essa escolha da variável de controle, substituindo em (2.84), o comprimento do passo resulta:

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathbf{r}_k}{\mathbf{r}_k^T (D_{\mathbf{f}} P D_{\mathbf{f}}^T)^2 \mathbf{r}_k} \tag{2.96}$$

o qual é sempre não negativo. A variação em cada iteração da função de Liapunov é dado por, substituindo (2.95) em (2.85)

$$\Delta V = -\frac{(\mathbf{r}_k^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathbf{r}_k)^2}{\mathbf{r}_k^T D_{\mathbf{f}} P D_{\mathbf{f}}^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \mathbf{r}_k} \leq 0 \tag{2.97}$$

a qual é sempre não positiva, garantindo a convergência assintótica do algoritmo, localmente falando.

Observe-se que a variável de controle é a mesma que a do algoritmo CJT, razão pela qual este algoritmo discreto será chamado de jacobiano trasposto discreto ou DJT.

A variável de estado em cada iteração será calculada, substituindo (2.95) em (2.74), como:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k P D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_k) \tag{2.98}$$

Neste caso, se em alguma iteração o resíduo entrar no espaço nulo de $D_{\mathbf{f}}^T$ mesmo sem estar no ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$, isto é, se para um determinado valor de $\mathbf{x}_k$, $\mathbf{f}(\mathbf{x}_k) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}_k))$, o algoritmo estará diante de uma indeterminação, pois $\Delta\mathbf{x} = \mathbf{0}/0$ e a seguinte iteração não poderá ser calculada. Eis a razão, neste caso, da convergência

local do algoritmo.

4) Quarta escolha

$$\mathbf{u}_k := P\text{sgn}(D_{\mathbf{f}}^T \mathbf{r}_k) \tag{2.99}$$

Com esta escolha da variável de controle, o comprimento do passo, substituindo (2.99) em (2.84), é dado por:

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} P\text{sgn}(D_{\mathbf{f}}^T \mathbf{r}_k)}{\text{sgn}^T(D_{\mathbf{f}}^T \mathbf{r}_k) P D_{\mathbf{f}}^T D_{\mathbf{f}} P\text{sgn}(D_{\mathbf{f}}^T \mathbf{r}_k)} \tag{2.100}$$

Também aqui observa-se que o comprimento do passo pode não ser positivo, mas substituindo a lei de controle (2.99) em (2.85):

$$\Delta V = -\frac{(\mathbf{r}_k^T D_{\mathbf{f}} P\text{sgn}(D_{\mathbf{f}}^T \mathbf{r}))^2}{\text{sgn}^T(D_{\mathbf{f}}^T \mathbf{r}_k) P D_{\mathbf{f}}^T D_{\mathbf{f}} P\text{sgn}(D_{\mathbf{f}}^T \mathbf{r}_k)} \leq 0 \tag{2.101}$$

o qual é sempre não positivo, garantindo assim a convergência do algoritmo. A variável de estado em cada iteração é calculada, substituindo (2.99) em (2.74):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k P\text{sgn}(D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_k)) \tag{2.102}$$

A variável de controle $\mathbf{u}_k$ coincide, a menos do ganho P, com a do algoritmo VJT, razão pela qual este algoritmo discreto será chamado de DVJT. Para a escolha particular $P = I$, ambas variáveis de controle coincidem, e o comprimento do passo α_k é não negativo.

Também aqui, ao igual que no caso anterior, deve-se destacar o caráter local da convergência, pois o algoritmo sofre uma indeterminação naqueles pontos $\mathbf{x}_k$ que provocam que a função entre no espaço nulo de $D_{\mathbf{f}}^T$ mesmo fora do ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$, isto é $\mathbf{f}(\mathbf{x}_k) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}_k))$, como acontecia no exemplo fornecido na terceira escolha dos algoritmos contínuos. Nestes casos, o cálculo da variável de estado na seguinte iteração será indeterminada pois $\Delta\mathbf{x} = \mathbf{0}/0$ e a seguinte iteração não poderá ser calculada.

Uma outra observação interessante sobre este algoritmo diz respeito ao sentido das chamadas linhas de campo, ou vetor $\Delta\mathbf{x}$. Para a escolha $P = I$, este vetor será sempre (para o caso $n = 2$), $\Delta\mathbf{x} = \alpha_k[\pm 1 \ \pm 1]^T$, a menos que uma componente determinada

do vetor $D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$ seja igual a zero. Portanto, as trajetórias entre uma iteração e outra serão sempre segmentos no plano de fase de um ângulo múltiplo de 45°.

5) Quinta escolha

$$\mathbf{u}_k := PD_{\mathbf{f}}^T \text{sgn}(\mathbf{r}_k) \tag{2.103}$$

sendo P uma matriz simétrica positiva definida.

Com esta variável de controle, o comprimento do passo α_k resulta, substituindo em (2.84):

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \text{sgn}(\mathbf{r}_k)}{\text{sgn}^T(\mathbf{r}_k)(D_{\mathbf{f}} P D_{\mathbf{f}}^T)^2 \text{sgn}(\mathbf{r}_k)} \tag{2.104}$$

Observe-se que aqui também o comprimento do passo pode não ser positivo, inclusive, a diferença dos casos anteriores, independentemente do valor do ganho P. Porém, substituindo (2.103) em (2.85), a variação da função de Liapunov em cada iteração é dada por:

$$\Delta V = -\frac{(\mathbf{r}_k^T D_{\mathbf{f}} P D_{\mathbf{f}}^T \text{sgn}(\mathbf{r}_k))^2}{\text{sgn}^T(\mathbf{r}_k)(D_{\mathbf{f}} P D_{\mathbf{f}}^T)^2 \text{sgn}(\mathbf{r}_k)} \leq 0 \tag{2.105}$$

o qual é sempre não positiva, garantindo o decrescimento da função de Liapunov V_k e assim a convergência assintótica do algoritmo, ao menos localmente. Aqui também, a razão de não poder garantir convergência global, deve-se à possibilidade do algoritmo se estacionar em um ponto $\mathbf{x}_k$ tal que $\text{sgn}(\mathbf{f}(\mathbf{x}_k)) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}_k))$, mesmo não estando no ponto de equilíbrio $\mathbf{r}_k = \mathbf{0}$. Nesse ponto o cálculo do ponto seguinte também será indeterminado.

A variável de controle coincide com aquela do sistema contínuo JTV, razão pela qual este algoritmo será chamado de DJTV.

A variável de estado em cada iteração será calculada, substituindo (2.103) em (2.74), como:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k PD_{\mathbf{f}}^T \text{sgn}(\mathbf{f}(\mathbf{x}_k)) \tag{2.106}$$

Cabe mencionar que todos os algoritmos discretos admitem como ganho uma matriz P simétrica positiva definida, mesmo que a sua correspondente versão contínua não o

admitisse, como é o caso dos sistemas NV e VJT. Também aqui nota-se a diferença com os algoritmos apresentados em [13, c. 2] e [14], onde só é mencionada a possibilidade de um ganho no algoritmo discreto DJT.

Na seguinte tabela, são mostrados resumidamente os 5 algoritmos discretos, com ganho $P = I$ a efeitos de simplificação.

$\mathbf{u}_k$	α_k	ΔV	nome	número de equação
$-D_{\mathbf{f}}^{-1}\mathbf{f}(\mathbf{x}_k)$	1	$-\|\mathbf{r}_k\|_2^2$	DN	(2.86)
$-D_{\mathbf{f}}^{-1}\mathrm{sgn}(\mathbf{f}(\mathbf{x}_k))$	$\frac{\|\mathbf{r}_k\|_1}{\|\mathrm{sgn}(\mathbf{r}_k)\|_2^2}$	$-\frac{\|\mathbf{r}_k\|_1^2}{\|\mathrm{sgn}^T(\mathbf{r}_k)\|_2^2}$	DNV	(2.91)
$-D_{\mathbf{f}}^{T}\mathbf{f}(\mathbf{x}_k)$	$\frac{\|D_{\mathbf{f}}^T\mathbf{r}_k\|_2^2}{\|D_{\mathbf{f}}D_{\mathbf{f}}^T\mathbf{r}_k\|_2^2}$	$-\frac{\|D_{\mathbf{f}}^T\mathbf{r}_k\|_2^4}{\|D_f D_{\mathbf{f}}^T\mathbf{r}_k\|_2^2}$	DJT	(2.95)
$-\mathrm{sgn}(D_{\mathbf{f}}^{T}\mathbf{f}(\mathbf{x}_k))$	$\frac{\|D_{\mathbf{f}}^T\mathbf{r}_k\|_1}{\|D_{\mathbf{f}}\mathrm{sgn}(D_{\mathbf{f}}^T\mathbf{r}_k)\|_2^2}$	$-\frac{\|D_{\mathbf{f}}^T\mathbf{r}\|_1^2}{\|D_{\mathbf{f}}\mathrm{sgn}(D_f^T\mathbf{r}_k)\|_2^2}$	DVJT	(2.99)
$-D_{\mathbf{f}}^{T}\mathrm{sgn}(\mathbf{f}(\mathbf{x}_k))$	$\frac{\mathbf{r}_k D_{\mathbf{f}} D_{\mathbf{f}}^T\mathrm{sgn}(\mathbf{r}_k)}{\|D_{\mathbf{f}}D_{\mathbf{f}}^T\mathrm{sgn}(\mathbf{r}_k)\|_2^2}$	$-\frac{(\mathbf{r}_k^T D_{\mathbf{f}} D_f^T\mathrm{sgn}(\mathbf{r}_k))^2}{\|D_{\mathbf{f}}D_{\mathbf{f}}^T\mathrm{sgn}(\mathbf{r}_k)\|_2^2}$	DJTV	(2.103)

Tabela 2.2 Algoritmos discretos de primeira ordem derivados com a metodologia CLF/LOC

2.6.1 Simulações dos algoritmos de primeira ordem discretos

A seguir, serão apresentadas simulações realizadas com as mesmas funções de teste dos algoritmos contínuos. As simulações foram realizadas com o programa Matlab 6.

Foi adotado como critério de parada se deter na iteração tal que a norma da função $\|\mathbf{f}(\mathbf{x}_k)\|_2 < 0.01$.

Os ganhos dos algoritmos discretos foram escolhidos $P = I$.

2.6.1.1 Simulações dos algoritmos discretos com a função de Camelback

Para simular com a função de Camelback (2.52) serão escolhidas as mesmas constantes que no caso contínuo. Os algoritmos também serão testados a partir dos mesmos pontos iniciais, isto é $\mathbf{x}_0 = [-1\ 1.5]^T$, $\mathbf{x}_0 = [0\ 1.5]^T$, $\mathbf{x}_0 = [2\ 1]^T$.

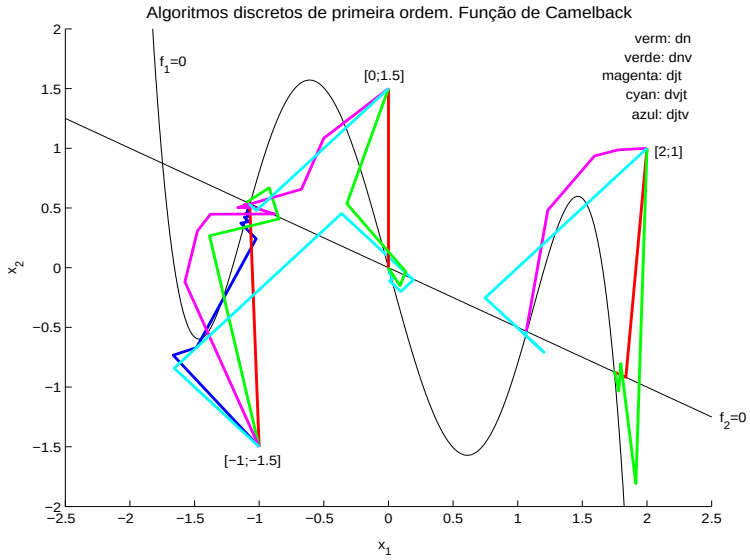

Figura 2.12: Simulações dos algoritmos discretos para a função de Camelback

Observa-se aqui também os algoritmos conseguiram achar zeros a partir dos pontos iniciais testados, embora em alguns casos convergiram para zeros diferentes mesmo quando partindo de um ponto inicial comum. A excecão está dada pelo algoritmo DJTV, que só convergiu a partir do ponto inicial $\mathbf{x}_0 = [-1\ \ -1.5]^T$. Interessante conferir o observado na teoria, as trajetórias como segmentos a múltiplos de 45° do algoritmo DVJT.

2.6.1.2 Simulações dos algoritmos discretos com a função de Rosenbrock

Para simular a função de Rosenbrock (2.55), foram escolhidas as mesmas constantes, $a = 0.5$ e $b = 1$, que para as funções contínuas, apresentando um único zero em $\mathbf{x}^* = [1\ 1]^T$. Também foram testados os algoritmos a partir dos mesmos pontos iniciais

$\mathbf{x}_0 = [-0.5\ 2.5]^T, \quad \mathbf{x}_0 = [0.5\ 2]^T, \quad \mathbf{x}_0 = [1.5\ -0.5]^T.$

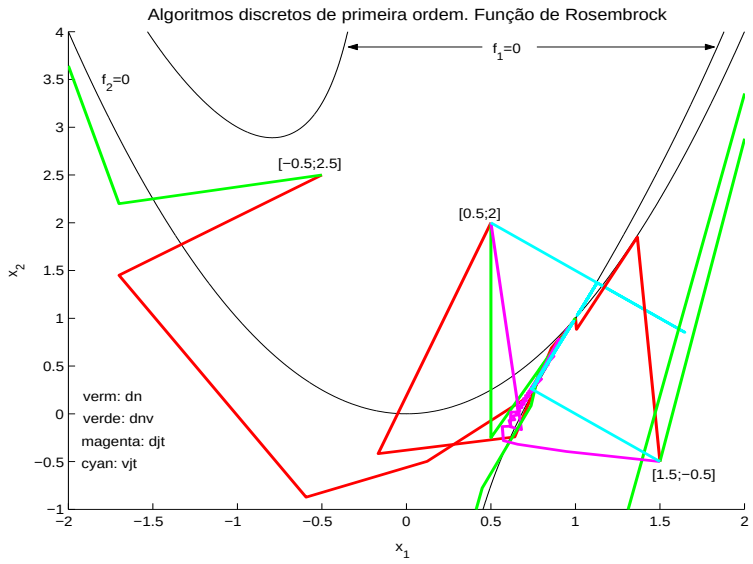

Figura 2.13: Simulações dos algoritmos discretos para a função de Rosenbrock

A partir do ponto $\mathbf{x}_0 = [-0.5\ 2.5]^T$ apenas o algoritmo de Newton conseguiu convergir. O algoritmo DJTV não convergiu a partir de nenhum dos pontos iniciais testados. Em apariência, das funções testadas esta é a que com maior dificuldade se encontraram os algoritmos para achar o zero da função.

2.6.1.3 Simulações dos algoritmos discretos com a função de Branin

Para simular a função de Branin (equação 2.57), foram escolhidas as mesmas constantes. A constante c também foi escolhida inicialmente igual a zero. Os pontos iniciais testados foram os mesmos que para o caso contínuo, isto é $\mathbf{x}_0 = [0.25\ -0.4]^T, \quad \mathbf{x}_0 = [0.75\ 0.6]^T, \quad \mathbf{x}_0 = [1.25\ -0.6]^T.$

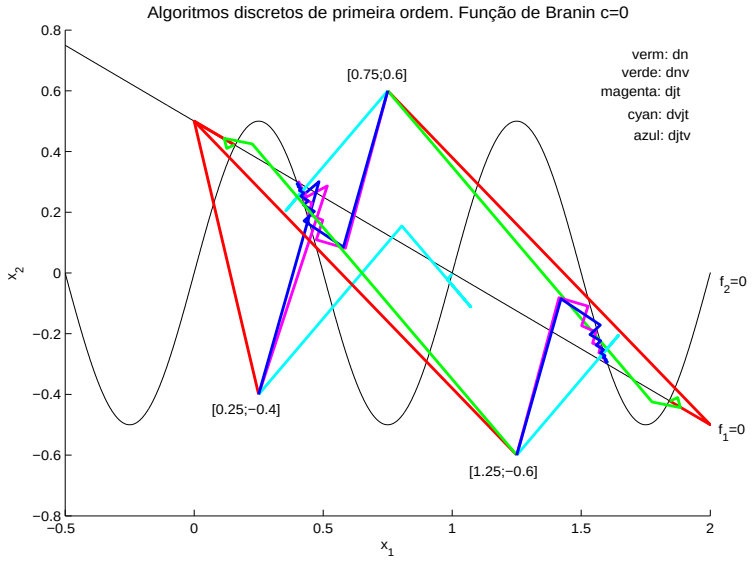

Figura 2.14: Simulações dos algoritmos discretos para a função de Branin, $c = 0$

O algoritmo DNV não convergiu a nenhum zero a partir do ponto inicial $\mathbf{x}_0 = [0.25 \ -0.4]^T$. Os diferentes algoritmos convergem a zeros diferentes mesmo a partir de pontos iniciais comuns, em alguns casos zeros distantes do ponto de partida. Também aqui se aprecia claramente o chattering característico das trajetórias geradas pelos algoritmos a estrutura variável, em particular o DJTV. O algoritmo DVJT também descreve trajetórias a múltiplos de 45°, tal como apreciado nas simulações anteriores.

Aumentando o valor da constante c para 0.13, a função continua apresentando 5 zeros.

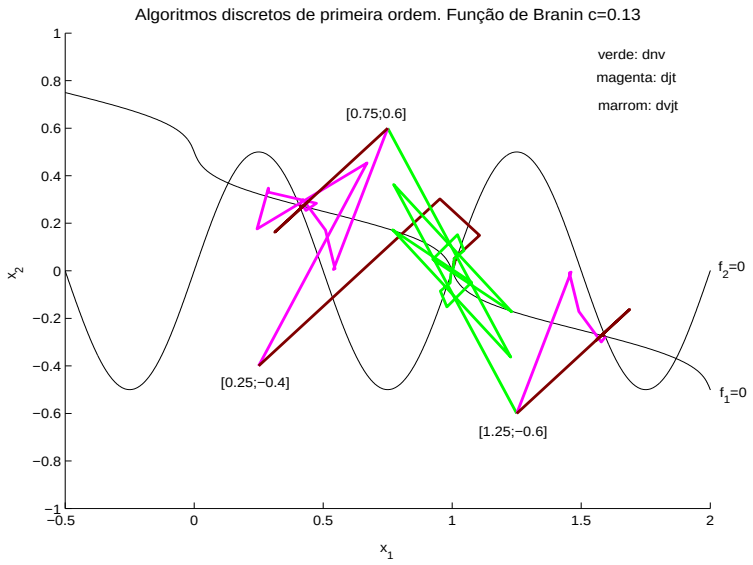

Figura 2.15: Simulações dos algoritmos discretos para a função de Branin, $c = 0.13$

Apenas os algoritmos DJT e DVJT convergiram a algum zero a partir de todos os pontos iniciais testados, sendo que o DNV convergiu a partir dos pontos $\mathbf{x}_0 = [0.75\ 0.6]^T$ e $\mathbf{x}_0 = [1.25\ \ -0.6]^T$. Destes pontos, o DNV atingiu um zero diferente a dos outros algoritmos e somente após um maior número de iterações. Destaca-se a impossibilidade para o algoritmo de Newton de achar zeros a partir dos 3 pontos iniciais testados.

Aumentando ainda a constante $c = 1$, a função de Branin apresentará agora um total de 15 zeros, dos quais apenas 11 deles aparecem na janela do seguinte gráfico.

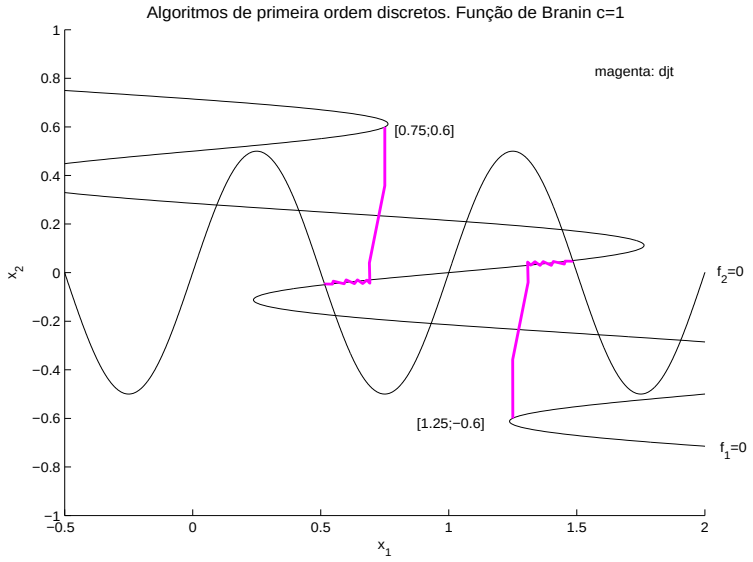

Figura 2.16: Simulações dos algoritmos discretos para a função de Branin, $c = 1$

Apenas o algoritmo DJT conseguiu convergir a zeros da função e em apenas dois dos pontos iniciais testados. Isto evidencia que quanto maior o valor da constante c, mais dificuldade encontram os algoritmos para achar os zeros da função.

A seguinte tabela mostra o número de iterações que empregou cada algoritmo discreto a partir de cada um dos pontos iniciais testados para todas as funções utilizadas.

	Camelback		Rosenbrock		Branin			
	$\mathbf{x}_0$	it.	$\mathbf{x}_0$	it.	$\mathbf{x}_0$	$c=0$ it.	$c=0.13$ it.	$c=1$ it.
DN	$(-1;-1.5)$	3	$(-0.5;2.5)$	7	$(0.25;-0.4)$	4	-	-
	$(0;1.5)$	2	$(0.5;2)$	6	$(0.75;0.6)$	4	-	-
	$(2;1)$	4	$(1.5;-0.5)$	4	$(1.25;-0.6)$	4	-	-
DNV	$(-1;-1.5)$	7	$(-0.5;2.5)$	-	$(0.25;-0.4)$	-	-	-
	$(0;1.5)$	5	$(0.5;2)$	5	$(0.75;0.6)$	5	9	-
	$(2;1)$	7	$(1.5;-0.5)$	10	$(1.25;-0.6)$	5	9	-
DJT	$(-1;-1.5)$	8	$(-0.5;2.5)$	-	$(0.25;-0.4)$	5	10	-
	$(0;1.5)$	7	$(0.5;2)$	93	$(0.75;0.6)$	15	6	14
	$(2;1)$	6	$(1.5;-0.5)$	167	$(1.25;-0.6)$	15	6	14
DVJT	$(-1;-1.5)$	16	$(-0.5;2.5)$	-	$(0.25;-0.4)$	5	17	-
	$(0;1.5)$	6	$(0.5;2)$	494	$(0.75;0.6)$	6	5	-
	$(2;1)$	10	$(1.5;-0.5)$	353	$(1.25;-0.6)$	6	5	-
DJTV	$(-1;-1.5)$	26	$(-0.5;2.5)$	-	$(0.25;-0.4)$	10	-	-
	$(0;1.5)$	-	$(0.5;2)$	-	$(0.75;0.6)$	17	-	-
	$(2;1)$	-	$(1.5;-0.5)$	-	$(1.25;-0.6)$	17	-	-

Tabela 2.3: Número de iterações realizadas pelos algoritmos discretos
onde - indica a divergência da trajetória

Observa-se que, em geral, o algoritmo DN é o que emprega um menor número de iterações para achar um zero. Porém, isto não se verifica a partir de todos os pontos iniciais. Destaca-se, também, a impossibilidade deste algoritmo para achar zeros na função de Branin com $c = 0.13$, a diferença de outros algoritmos testados.

O tempo de demora dos algoritmos foi testado também, embora não tenha sido colocado na tabela por ser um dado inconclusivo (diferentes simulações apresentaram diferentes demoras). Embora os tempos foram em todos os casos muito pequenos (da ordem das centéssimas de segundo), o algoritmo DN foi na maioria dos casos o que mais tempo demorou, apesar da economia de iterações. Cabe supor que para um função com um maior número de variáveis, esta diferença de tempo pode se tornar apreciável.

2.6.2 Simulações com funções de mais de duas variáveis

A seguir serão apresentadas simulações dos algoritmos discretos realizadas com funções de um maior número de variáveis. Estas funções são conhecidas na bibliografia e foram extraidas de [42], [55], [84], [62] e [76]. Cabe destacar que, embora algumas funções são apresentadas de maneira diferente em cada uma destas bibliografias, estas diferenças não parecem ser determinantes no que se refere à forma geral da função, quantidade de zeros, etc. Aqui será escolhida arbitrariamente uma versão da função. Outra consideração é que as funções são escalares, $\phi(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$, e será achado um zero do gradiente destas.

As funções escalares, de N variáveis, escolhidas são:

- Rosenbrock generalizada

$$\phi(\mathbf{x}) = \sum_{i=1}^{N-1} (1 - x_i^2)^2 + 100(x_{i+1} - x_i^2)^2 \tag{2.107}$$

- Michalewicz

$$\phi(\mathbf{x}) = -\sum_{i=1}^{N} \sin(y_i) \sin^{20}\left(\frac{iy_i^2}{\pi}\right) \tag{2.108}$$

onde

$$\begin{aligned} y_i &= x_i \cos(\tfrac{\pi}{6}) - x_{i+1} \sin(\tfrac{\pi}{6}) \quad \text{se } i \bmod 2 = 1 \\ y_i &= x_{i-1} \sin(\tfrac{\pi}{6}) + x_i \cos(\tfrac{\pi}{6}) \quad \text{se } i \bmod 2 = 0 \text{ e } i \neq N \\ y_N &= x_N \end{aligned}$$

Aqui desconsideramos o exponente 20 para dar valores mais adequados. Algumas bibliografias ([42], por exemplo) consideram diretamente $y_i = x_i$ para todo $i \in \{1, .., N\}$.

- Shekel

$$\phi(\mathbf{x}) = -\sum_{i=1}^{5} \frac{1}{\sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i} \tag{2.109}$$

onde

$$A = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

Esta função está definida para um número fixo de variáveis igual a 4.

- Powell

$$\phi(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \tag{2.110}$$

Esta função também está definida para um número fixo de variáveis $N = 4$, embora [42] estende a definição para um número genérico de variáveis.

- Rastrigin

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} \left[x_i^2 - 10\cos(2\pi x_i)\right] + 10N \tag{2.111}$$

- Schwefel

$$\phi(\mathbf{x}) = -\sum_{i=1}^{N} x_i \sin(\sqrt{|x_i|}) \tag{2.112}$$

- Ackley

$$\phi(\mathbf{x}) = -20e^{(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2})} - e^{(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i))} \tag{2.113}$$

- Griewank

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{2.114}$$

- Levy

$$\phi(\mathbf{x}) = \frac{\pi}{N}\{ \quad \sin^2\left(\pi\tfrac{5+x_1}{4}\right) + \textstyle\sum_{i=1}^{N-1}\left(\tfrac{1+x_i}{4}\right)^2 \\ \left[1 + 10\sin^2\left(\pi\tfrac{5+x_{i+1}}{4}\right)\right] + \left(\tfrac{x_N+1}{4}\right)^2 \Big\} \tag{2.115}$$

Em diferentes bibliografias aparece a seguinte versão da função de Levy:

- Levy2

$$\phi(\mathbf{x}) = \frac{1}{10}\,[\quad \sin^2(3\pi x_1) + \sum_{i=1}^{N-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + \qquad (2.116)$$
$$(x_N-1)^2[1+\sin^2(2\pi x_N)]\big]$$

- Bump

$$\phi(\mathbf{x}) = \left| \frac{\sum_{i=1}^{N}\cos^4(x_i) - 2\prod_{i=1}^{N}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{N} i x_i^2}} \right| \qquad (2.117)$$

- Lennard-Jones

$$\phi(\mathbf{x}) = \sum_{i=2}^{N/2}\sum_{j=1}^{i-1} \quad \Big\{ \left[(x_i-x_j)^2 + (x_{N/2+i} - x_{N/2+j})^2\right]^{-6} \qquad (2.118)$$
$$-2\left[(x_i-x_j)^2 + (x_{N/2+i} - x_{N/2+j})^2\right]^{-3} \Big\}$$

Aqui o número de variáveis N deve ser par.

- Hartmann 3

$$\phi(\mathbf{x}) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2} \qquad (2.119)$$

onde

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix} \quad P = 10^{-4}\begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$

Esta função está definida apenas para 3 variáveis.

- Zakharov

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} x_i^2 + \left(\sum_{i=1}^{N} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{N} 0.5 i x_i\right)^4 \qquad (2.120)$$

- Hartmann 6

$$\phi(\mathbf{x}) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2} \qquad (2.121)$$

onde

$$A = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 1.0 \\ 1.2 \\ 3.0 \\ 3.2 \end{bmatrix}$$

$$P = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

Esta função está definida apenas para 6 variáveis.

- Colville

$$\begin{aligned} \phi(\mathbf{x}) = \quad & 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3 - x_4)^2 + \\ & 10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1) \end{aligned} \tag{2.122}$$

Esta função está definida apenas para 4 variáveis.

- Dixon & Price

$$\phi(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{N} i(2x_i^2 - x_{i-1})^2 \tag{2.123}$$

- Perm

$$\phi(\mathbf{x}) = \sum_{j=1}^{N} \left[\sum_{i=1}^{N} (i^j + \beta) \left(\left(\frac{x_i}{i} \right)^j - 1 \right) \right]^2 \tag{2.124}$$

Aqui o valor β foi escolhido unitário.

- Power sum

$$\phi(\mathbf{x}) = \sum_{j=1}^{N} \left[\left(\sum_{i=1}^{N} x_i^j \right) - b_j \right]^2 \tag{2.125}$$

onde foi escolhido $b_j = 1$ para todo $j \in \{1, .., N\}$.

O gradiente desta função apresenta um único zero na origem.

- Trid

$$\phi(\mathbf{x}) = \sum_{i=1}^{N} (x_i - 1)^2 - \sum_{i=1}^{N} x_i x_{i-1} \tag{2.126}$$

O gradiente desta função é linear, e portanto seu hessiano constante.

As simulações foram realizadas no programa Matlab 6. Em todos os casos, o critério de parada foi $\|\mathbf{f}(\mathbf{x}_k)\| \leq 10^{-4}$.

As funções Michalewicz, Shekel e Lennard-Jones tiveram seus gradientes e hessianos calculados em forma simbólica pelo programa Matlab utilizando o pacote *Symbolic Toolbox*. Mesmo que estas expressões sejam calculadas apenas uma vez, e a cada iteração o algoritmo apenas substitua nas expressões o valor do ponto corrente $\mathbf{x}_k$, este procedimento é extremadamente lento. Eis a razão pelo limitado número de variáveis com que foram testadas estas funções (no caso da função de Shekel, este número é fixo).

As simulações foram consideradas falhas se o algoritmo não conseguiu atingir um zero do gradiente da função, manifestando um erro durante a execução deste, ou não atingiu o objetivo em até 300 iterações (no caso das funções mencionadas no parágrafo anterior, em até 30 iterações por causa da lentidão da execução).

Os resultados das simulações são apresentados na seguinte tabela.

	N	$\mathbf{x}_0$	DN		DNV		DJT		DVJT		DJTV	
			it.	t[s]	it.	t[s]	it.	t[s]	it.	t[s]	it.	t[s]
Rosenbrock	10	$x_{0_i} = 0.5$	28	1.15	-		-		-		-	
		$x_{0_i} = -0.2$	23	0.05	-		-		-		-	
	4	$[-0.5; 0; 0.2; 1.5]^T$	21	0.6	-		-		-		-	
		$[1.2; 1; 0; 1.5]^T$	10	0.27	-		-		-		-	
Michalewicz	4	$[1.2; 0.1; 1.3; 0.3]^T$	7	6.37	-		-		-		-	
		$x_{0_i} = 1$	6	6.1	14	13.3	-		-		-	
	6	$x_{0_i} = 0.5$	7	12.2	-		-		27	41.9	-	
		$x_{0_i} = -0.1$	5	12.5	-		28	44.6	19	30.8	-	
Shekel	4	$[3.5; 4.5; 3.5; 4.5]^T$	7	6.65	10	9.39	-		-		-	
		$x_{0_i} = 1$	3	3.41	5	5.39	29	24.3	5	5.43	5	8.08
Powell	4	$[-2; 3; -5; 7]^T$	17	0.88	-		-		-		-	
		$x_{0_i} = 1$	20	1.21	-		-		-		-	
Rastrigin	10	$x_{0_i} = 0.2$	6	0.17	6	0	6	0.11	6	0	6	0
		$x_{0_i} = -0.8$	8	0	8	0	8	0	8	0.28	8	0.27
	100	$x_{0_i} = -1.3$	5	0.27	5	0.44	5	0.83	5	0.66	5	0.71
		$x_{0_i} = 2.3$	6	0.61	6	0.55	6	0.88	6	0.77	6	0.93
Schwefel	10	$x_{0_i} = 0.2$	7	0.71	7	0.33	7	0.33	7	0.22	7	0.22
		$x_{0_i} = -2.3$	3	0.06	3	0.06	3	0.05	3	0.05	3	0.06
	100	$x_{0_i} = 0.2$	8	1.1	8	0.94	8	1.59	8	1.37	8	1.54
		$x_{0_i} = -2.3$	3	0.33	3	0.71	3	0.6	3	0.55	3	0.6
Ackley	10	$x_{0_i} = 2$	2	0.05	2	0	2	0.05	2	0.11	2	0.11
		$x_{0_i} = -3$	2	0.06	2	0.05	2	0.17	2	0	2	0.06
	100	$x_{0_i} = -0.2$	4	0.6	4	1.26	4	1.53	4	1.37	4	1.43
		$x_{0_i} = 0.05$	11	1.97	11	1.87	11	3.9	11	3.73	11	3.85
Griewank	10	$x_{0_i} = -1$	14	1.1	-		-		-		-	
		$x_{0_i} = -0.2$	2	0.06	28	0.88	166	5.22	2	0.05	-	
		$x_{0_i} = 5$	167	5.21	-		-		-		-	
	100	$x_{0_i} = 0.5$	3	0.44	-		-		-		-	
		$x_{0_i} = 10$	1	0.16	1	0.16	1	0.27	1	0.22	1	0.22
		$x_{0_i} = 0.1$	2	0.22	312	42.84	-		-		-	
Levy	10	$x_{0_i} = 0$	13	0.5	-		291	7.53	5	0.16	-	
		$x_{0_i} = -2.5$	-		-		238	7.47	3	0.11	-	
	100	$x_{0_i} = 0$	-		-		136	26.8	5	0.99	-	
		$x_{0_i} = -2.5$	-		-		184	37.9	4	0.76	-	
Levy2	10	$x_{0_i} = 0$	2	0.66	17	0.6	9	0.71	3	0.44	22	0.66
		$x_{0_i} = 2.5$	3	0.22	35	1.04	21	0.66	287	9.12	-	
		$x_{0_i} = -0.7$	7	0.38	24	0.93	38	1.16	68	2.58	-	
	100	$x_{0_i} = 0$	3	0.44	172	18.8	9	1.86	141	25.5	73	14.3
		$x_{0_i} = 2.5$	3	0.33	-		27	5.44	-		-	
		$x_{0_i} = -0.7$	7	0.77	315	34.7	38	7.52	-		-	
Bump	10	$x_{0_i} = -0.275$	10	0.5	-[1]		12	0.61	11	1.05	12	0.6
		$x_{0_i} = \frac{\pi}{4}$	8	0.33	-[1]		8	0.38	8	0.05	8	0.11
	100	$x_{0_i} = -\frac{\pi}{8}$	6	2.04	4	1.26	10	7.96	9	7.03	10	7.91
		$x_{0_i} = 0.7$	8	2.53	5	1.76	8	6.37	8	6.26	8	6.32
Lennard-Jones	4	$[-0.1; 0; 0.1; 0.3]^T$	-		-		25	12.5	25	12.6	25	20.8
		$[-1; 2; 0; -0.5]^T$	-		-		5	3.03	5	2.92	5	4.66
Hartmann 3	3	$x_{0_i} = 0$	8	0.93	2	0.44	8	0.71	8	0.55	8	0.61
		$[-0.5; 1.5; 0.2]^T$	5	0.16	1	0.06	5	0.28	5	0.17	5	0.11
Hartmann 6	6	$x_{0_i} = 0$	-		206	7.36	-		-		-	
		$x_{0_i} = 0.3$	4	0.38	26	0.88	-		-		-	

Tabela 2.4 Resultados das simulações dos algoritmos discretos para funções de N variáveis

	N	$\mathbf{x}_0$	DN		DNV		DJT		DVJT		DJTV	
			it.	t[s]	it.	t[s]	it.	t[s]	it.	t[s]	it.	t[s]
Zakharov	10	$x_{0_i} = 1$	13	.05	88	3.4	-		13	0.49	-	
		$x_{0_i} = -0.5$	11	0.06	80	2.3	-		11	0.39	-	
	100	$x_{0_i} = 1$	24	3.29	-		-		24	4.44	-	
		$x_{0_i} = -0.5$	23	2.96	-		-		23	4.17	-	
Colville	4	$[10; -5; 3; -4]^T$	17	0.06	-		88	3.24	-		-	
		$x_{0_i} = -1$	5	0.16	-		66	2.36	77	2.47	-	
Dixon & Price	10	$x_{0_i} = 0$	-		-		1	0.11	1	0.22	1	0.11
		$x_{0_i} = -1$	9	0.38	-		-		215	7.14	-	
		$x_{0_i} = -0.2$	6	0.33	-		58	1.76	210	6.75	-	
	100	$x_{0_i} = 0$	-		-		1	0.27	1	0.22	1	0.22
		$x_{0_i} = -1$	9	0.77	-		-		-		-	
		$x_{0_i} = -0.2$	6	0.77	-		-		-		-	
Perm	10	$x_{0_i} = 0$	5	0.43	-		49	4.5	38	2.47	367	29.1
		$x_{0_i} = 2$	38	0.71	-		51	3.25	69	4.29	-	
		$x_{0_i} = -1$	25	1.26	-		36	2.42	64	3.13	69	4.67
	30	$x_{0i} = 0$	28	5	-		115	35.7	165	51.5	-	
		$x_{0_i} = 2$	150	26.8	-		-		259	80.4	-	
		$x_{0_i} = -1$	103	18.9	-		18	36.9	216	67.6	160	50.1
Power Sum	10	$x_{0_i} = 0$	2	0.11	2	0.11	2	0.11	2	0.11	2	0.05
		$x_{0_i} = -3$	28	0.99	28	0.22	28	0.33	28	0.38	28	1.21
	30	$x_{0_i} = 1$	12	1.49	12	1.49	12	2.86	12	2.48	12	2.37
		$x_{0_i} = -3$	74	9.06	-		74	14.9	74	14.9	74	15.3
Trid	10	$x_{0_i} = 0$	1	0.05	1	0	-		-		-	
		$x_{0_i} = 20$	1	0.06	25	0.72	-		-		-	
	100	$x_{0_i} = 0$	1	0.11	1	0.11	-		-		-	
		$x_{0_i} = 20$	1	0.11	303	23.95	-		-		-	

Tabela 2.4 (cont.) Resultados das simulações dos algoritmos discretos para funções de N variáveis

Em todos os casos em que o ponto inicial é expressado como x_{0_i}, o índice é $\forall i \in \{1, .., N\}$.

[1] Nestes casos, o algoritmo DNV atingiu o zero que o gradiente na função Bump apresenta no infinito.

O gradiente da função de Shekel também apresenta zeros no infinito, onde podem tender as trajetórias dos algoritmos dependendo da escolha do ponto inicial. Nos dois casos testados, os algoritmos, quando convergiram, o fizeram a zeros finitos.

A função Trid é linear, eis a razão pelo qual o algoritmo de Newton achou o zero em apenas uma iteração, como foi analisado quando apresentado o algoritmo.

Os tempos de simulação muito pequenos (menores a 1s.) não são dados confiáveis, pois diferentes simulações resultaram em diferentes demoras. Para tempos de execução maiores, obviamente o erro relativo decresce.

Aqui, o algoritmo de Newton discreto se mostrou o mais eficiente, mesmo utilizando

o hessiano da função.

2.6.3 Utilização de um conjunto dos algoritmos a fim de aumentar a possibilidade de convergência

A pesar da eficiência do algoritmo de Newton, foi observado que nem sempre este converge a um zero, como mostrado na figura 2.15, por exemplo. Os outros algoritmos também podem não convergir a partir de um determinado ponto inicial. Quando interessa aumentar a possibilidade de convergência para uma determinada função de teste, existe a estratégia de utilizar um "time" de algoritmos. Esta metodologia consiste em calcular, em cada iteração, o ponto seguinte $\mathbf{x}_{k+1}$ de todos os algoritmos do time, e escolher aquele que apresenta um $\|\mathbf{f}(\mathbf{x}_{k+1})\|_2$ menor, garantindo assim a maior diminuição da norma da função de teste a cada iteração. Esta é uma justificativa para a elaboração e utilização de outros algoritmos que, embora possam ser individualmente menos eficientes do que o Newton, podem ser utilizados quando este falha. Na prática, utilizando os cinco algoritmos discretos apresentados, foi observado que nem sempre é escolhido o algoritmo de Newton como o que provoca maior decrescimento da norma da função, como mostra a figura 2.17 para a função de Branin com constante $c = 0.13$ a partir de três pontos iniciais escolhidos arbitrariamente.

Pela construção do time de algoritmos, observa-se que a presença de um algoritmo convergente dentro deste sempre garantirá a convergência do time como um todo.

Os ganhos continuam $P = I$, e a simulação teve como critério de parada a iteração tal que $\|\mathbf{f}(\mathbf{x}_k)\|_2 < 0.01$, critério adotado nas simulações para as funções de duas variáveis. Os pontos iniciais continuaram sendo $\mathbf{x}_0 = [0.25 \;\; -0.4]^T$, $\mathbf{x}_0 = [0.75 \;\; 0.6]^T$, $\mathbf{x}_0 = [1.25 \;\; -0.6]^T$. As cores de cada iteração correspondem com as escolhidas anteriormente para cada um dos algoritmos. Desta forma, cada iteração poderá utilizar um algoritmo diferente, e o trajeto até esse ponto é graficado com a cor correspondente ao algoritmo escolhido.

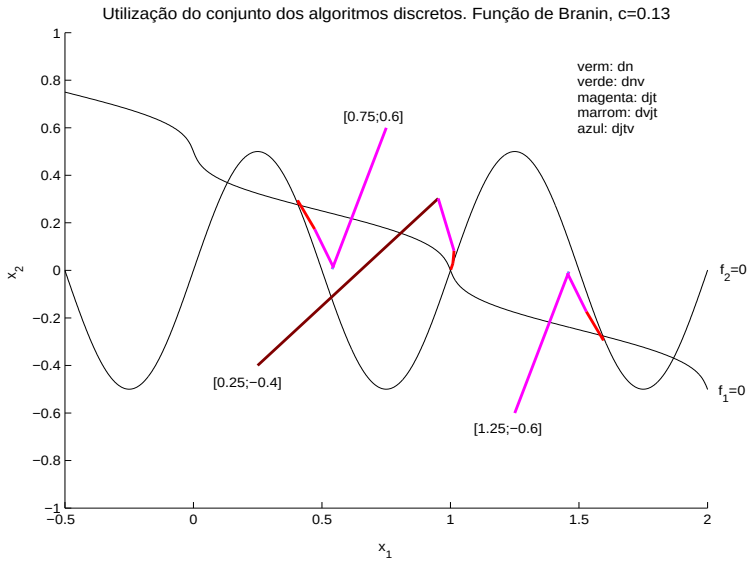

Figura 2.17: Simulações do conjunto dos algoritmos discretos para a função de Branin, $c = 0.13$

A partir do ponto inicial $\mathbf{x}_0 = [0.25 \ \ -0.4]^T$ o número de iterações foi de 5 e o tempo de demora foi 0.66s.

A partir do ponto inicial $\mathbf{x}_0 = [0.75 \ 0.6]^T$ o número de iterações foi de 6 e o tempo de demora foi 0.11s.

A partir do ponto inicial $\mathbf{x}_0 = [1.25 \ \ -0.6]^T$ o número de iterações foi de 6 e o tempo de demora foi 0.05s.

2.7 Comparação entre as bacias de atração dos diferentes algoritmos discretos

Nesta seção será realizado um estudo comparativo dos diferentes algoritmos de primeira ordem discretos. O estudo consiste na análise das diferentes regiões, ou bacias, de atração dos diferentes zeros de uma função a partir de pontos iniciais distribuidos uniformemente numa janela determinada. Como funções de teste serão escolhidas funções de duas variáveis a fim de poder graficar no plano tais bacias de atração de cada zero.

2.7.1 Função de Camelback

Na função de Camelback (2.52), foram escolhidas como constantes $a = -2$, $b = 1.05$, $c = -\frac{1}{6}$, $d = -1$ e $e = 0$. Com estas constantes a função apresenta um máximo global, dois máximos locais, e duas selas. Esses pontos correspondem a zeros do gradiente da primitiva descrito na equação (2.51).

O determinante do hessiano desta função (equação 2.53) é dado, para as constantes escolhidas, pela função $\det(D_{\mathbf{f}}(\mathbf{x})) = 7 - 25.2x_1^2 + 10x_1^4$, o qual possui zeros ao longo das retas verticais definidas por $x_1 = 1.7737$, $x_1 = 0.6739$, $x_1 = -0.6739$, $x_1 = -1.7737$. Como já foi observado, para as constantes escolhidas, $D_{\mathbf{f}}(\mathbf{x}) = D_{\mathbf{f}}(x_1)$.

A seguir, serão apresentados os gráficos das bacias de atração de cada zero da função de Camelback para cada um dos cinco algoritmos de primeira ordem. Os ganhos foram sempre escolhidos $P = I$. Os pontos iniciais foram escolhidos com intervalos de 0.05 para cada variável ao longo da superfície da janela definida entre as coordenadas $x_1 \in [-2.5, 2.5]$ $x_2 \in [-1.5, 1.5]$, dentro da qual se localizam todos os zeros. Foi testado um máximo de 249 iterações, isto é, os pontos em branco correspondem a pontos iniciais onde o algoritmo não atingiu nenhum zero em até esse número de iterações. Após a apresentação de cada um dos gráficos, serão apresentadas conclusões resultantes da análise de cada algoritmo aplicado à função de Camelback e da análise das formas das bacias de atração de cada zero.

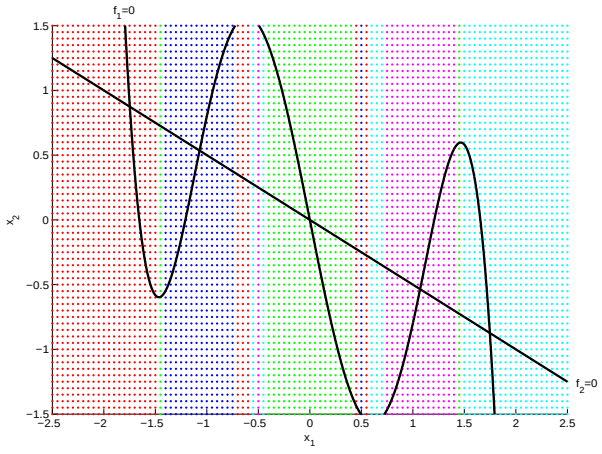

Figura 2.18: Bacias de atração da função de Camelback para o algoritmo de Newton

Sendo o algoritmo de Newton com comprimento do passo ótimo definido por $\mathbf{x}_{k+1} =$

$\mathbf{x}_k - D_{\mathbf{f}}^{-1}\mathbf{f}(\mathbf{x})$ (equação 2.88), conclui-se:

a) $\Delta x_1 = -(7x_1 - 8.4x_1^3 + 2x_1^5)/(7 - 25.2x_1^2 + 10x_1^4)$, portanto, determinado unicamente pela variável x_1.

b) Eis a razão das bacias em forma de faixas verticais, pois a componente horizontal do deslocamento das trajetórias será constante para um determinado valor de x_1.

c) O algoritmo não apresenta singularidades estranhas para esta função. Os zeros de $\text{adj}(D_{\mathbf{f}}(\mathbf{x}))\mathbf{f}(\mathbf{x})$ correspondem em todos os casos a zeros da função de Camelback.

d) Perto das retas verticais determinadas pelo locus de $\det(D_{\mathbf{f}}) = 0$, as linhas de campo de deslocamento, definidas por $\Delta\mathbf{x}_k = (\text{adj}(D_{\mathbf{f}}(\mathbf{x}_k))\mathbf{f}(\mathbf{x}_k))/\det(D_{\mathbf{f}}(\mathbf{x}_k))$ possuem uma norma de valor elevado e por essa razão as trajetórias podem não convergir ao zero mais próximo.

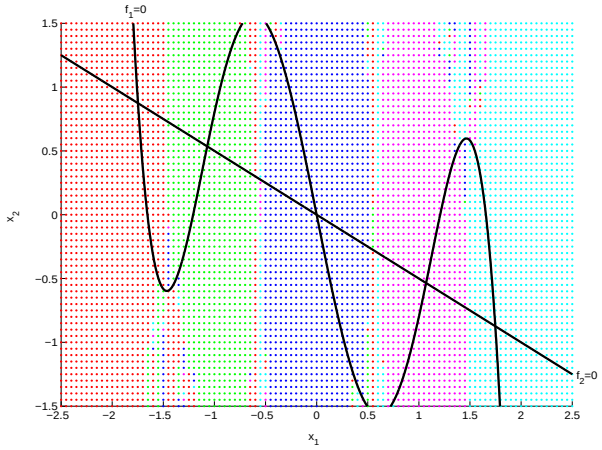

Figura 2.19: Bacias de atração da função de Camelback para o algoritmo DNV

A equação do algoritmo DNV é dada por $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{f}(\mathbf{x}_k))$, (equação 2.88), sendo o comprimento do passo ótimo $\alpha_k = (\mathbf{f}^T\text{sgn}(\mathbf{f}))/\|\text{sgn}(\mathbf{f})\|_2^2$. Também aqui podemos concluir:

a) O algoritmo não apresenta singularidades estranhas.

b) Por ser $\alpha_k > 0$ (exceto nos zeros da função), os sentidos das linhas de campo $\Delta\mathbf{x}_k$ estão unicamente determinados pelos sinais das componentes do vetor $D_{\mathbf{f}}^{-1}\text{sgn}(\mathbf{f})$.

c) As formas das bacias são faixas verticais encerradas pelo locus $\det(D_{\mathbf{f}}) = 7 - 25.2x_1^2 + 10x_1^4 = 0$, embora próximos destas retas verticais as faixas apresentam alguma deformação. A razão é que, a diferença com o algoritmo de Newton, aqui Δx_1 não é mais univocamente determinado pela variável x_1.

d) Perto das fronteiras das faixas, por ser a norma das linhas de campo $\Delta\mathbf{x}_k$ de um valor elevado, as trajetórias podem não converger ao zero mais próximo.

e) Nos zeros da função, as trajetórias do algoritmo apresentam uma indeterminação, pois $\mathbf{x}_{k+1} = [0\ 0]^T/0$ (assumindo sgn(0) = 0), o qual não compromete porque é justamente nestes pontos onde pára o algoritmo, não sendo calculada portanto uma próxima iteração.

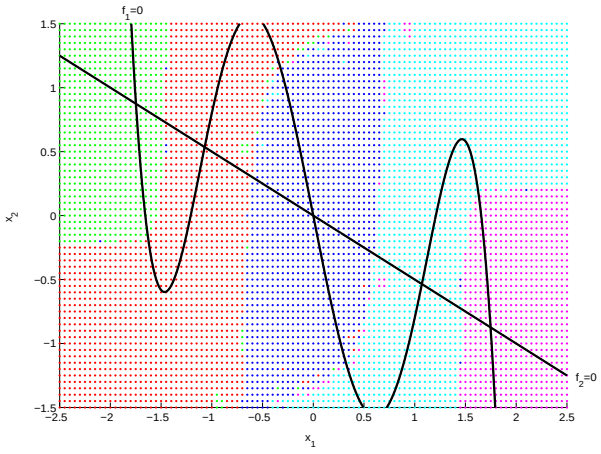

Figura 2.20: Bacias de atração da função de Camelback para o algoritmo de DJT

O algoritmo DJT stá dado por $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k D_\mathbf{f}^T \mathbf{f}(\mathbf{x})$ (equação 2.95), sendo o comprimento do passo $\alpha_k = (\mathbf{f}^T D_\mathbf{f} D_\mathbf{f}^T \mathbf{f})/(\mathbf{f}^T (D_\mathbf{f} D_\mathbf{f}^T)^2 \mathbf{f})$. Analisando as bacias e a equação do algoritmo, pode se concluir:

a) As bacias apresentam formas geométricas regulares. Suas fronteiras são determinadas pelo locus de $\mathbf{f}^T (D_\mathbf{f} D_\mathbf{f}^T)^2 \mathbf{f} = 0$, o lugar das raízes do denominador de α_k.

b) Por ser $\alpha_k > 0$ (exceto nos zeros de $D_\mathbf{f}^T \mathbf{f}$) os sentidos das linhas de campo das trajetórias $\Delta\mathbf{x}_k$ estão determinados pelos sinais das componentes do vetor $D_\mathbf{f}^T \mathbf{f}$.

c) Perto das fronteiras de cada bacia, por ter o vetor $\Delta\mathbf{x}_k$ uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

d) Nos zeros de $D_\mathbf{f}^T \mathbf{f}$, as trajetórias estão indeterminadas, pois $\Delta\mathbf{x}_k = [0\ 0]^T/0$. Esses zeros estarão nos pontos: x_1 nas raízes do polinômio $14.6x_1 + 20x_1^2 - 50.04x_1^3 - 63x_1^4 + 16.44x_1^5 + 25x_1^6 - 14.28x_1^7 - x_1^9 = 0$, $x_2 = -1.2x_1 + 0.84x_1^3 - 0.2x_1^5$, cujas singularidades estranhas correspondem aos pontos $\mathbf{x} = [-1.4842\ 0.4751]^T$, $\mathbf{x} = [1.4842\ -0.4751]^T$, $\mathbf{x} = [-0.5638\ 0.5374]^T$, $\mathbf{x} = [0.5638\ -0.5374]^T$. Os efeitos das raízes dos polinômios são equivalentes aos das singularidades para o algoritmo de Newton, isto é, singularidades

essenciais nos zeros da função $\mathbf{f}(\mathbf{x})$, e singularidades estranhas nos pontos mencionados, pontos onde $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$ e $D_{\mathbf{f}}^T\mathbf{f} = [0\ 0]^T$.

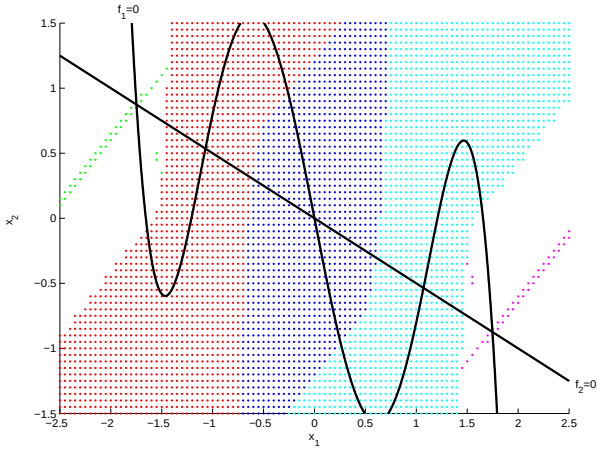

Figura 2.21: Bacias de atração da função de Camelback para o algoritmo de DVJT

O algoritmo DVJT está dado por $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \text{sgn}(D_{\mathbf{f}}^T\mathbf{f})$ (equação 2.95), sendo o comprimento do passo ótimo $\alpha_k = (\mathbf{f}^T D_{\mathbf{f}} \text{sgn}(D_{\mathbf{f}}^T\mathbf{f}))/(\text{sgn}^T(D_{\mathbf{f}}^T\mathbf{f}) D_{\mathbf{f}}^T D_{\mathbf{f}} \text{sgn}(D_{\mathbf{f}}^T\mathbf{f}))$. Analisando as bacias e o algoritmo, conclui-se:

a) Por ser $\alpha_k > 0$ (exceto nos zeros de $D_{\mathbf{f}}^T\mathbf{f}$), os sentidos das linhas de campo $\Delta\mathbf{x}_k$ só podem ser proporcionais a $[\pm 1\ \pm 1]^T$, a não ser naqueles pontos onde alguma componente do vetor $D_{\mathbf{f}}^T\mathbf{f}$ se anula (assumindo sempre $\text{sgn}(0) = 0$), em cujo caso essa componente do vetor de campo $\Delta\mathbf{x}_k$ é zero.

b) Esses sentidos das linhas de campo $\Delta\mathbf{x}_k$ provocam que algumas das fronteiras das bacias sejam retas a 45°. As bacias apresentam fronteiras com formas geométricas regulares.

c) As trajetórias estão indeterminadas nos mesmos pontos que o algoritmo DJT, isto é, nos zeros do vetor $D_{\mathbf{f}}^T\mathbf{f}$. Ambos algoritmos possuem singularidades da mesma classe e nos mesmos pontos.

d) Por ser um algoritmo de estrutura variável, pode demorar um maior número de iterações que os outros algoritmos para achar um zero. De fato, as bacias verde e roxa não existiam quando o algoritmo foi testado com um máximo de 100 iterações, isto é, a partir desses pontos iniciais o algoritmo DVJT demorou mais de 99 iterações para achar um zero. Perto dessas bacias, existe uma grande quantidade de pontos iniciais

a partir dos quais o algoritmo demorou mais de 249 iterações para atingir um zero (representados por pontos em branco).

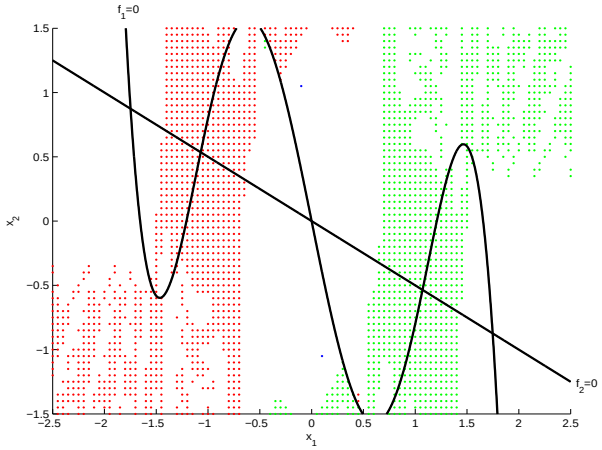

Figura 2.22: Bacias de atração da função de Camelback para o algoritmo de DJTV

O algoritmo DJTV está dado por $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k D_{\mathbf{f}}^T \text{sgn}(\mathbf{f})$ (equação 2.103), sendo o comprimento do passo ótimo dado por
$\alpha_k = (\mathbf{f}^T D_{\mathbf{f}} D_{\mathbf{f}}^T \text{sgn}(\mathbf{f}))/(\text{sgn}^T(\mathbf{f})(D_{\mathbf{f}} D_{\mathbf{f}}^T)^2 \text{sgn}(\mathbf{f}))$.

a) É o único dos cinco algoritmos que pode apresentar comprimento do passo negativo, portanto, o sentido das linhas de campo não está unicamente determinado pelo sinal das componentes do vetor $D_{\mathbf{f}}^T \text{sgn}(\mathbf{f})$.

b) Por ser um algoritmo de estrutura variável, pode demorar um grande número de iterações para achar um zero. Eis a razão pela qual a partir de muitos pontos iniciais o algoritmo não se mostrou capaz de atingir nenhum zero em menos de 250 iterações (pontos iniciais em branco). Inclusive, dois zeros não puderam ser achados a partir de nenhum ponto inicial.

c) As formas das bacias não respondem a formas geométricas regulares, apresentando inclusive contornos de difícil determinação.

d) Dentro das regiões de não convergência, os sentidos das linhas de campo $\Delta\mathbf{x}_k$ não são constantes para todos os pontos.

e) O algoritmo apresenta pontos singulares apenas nos zeros da função $\mathbf{f}(\mathbf{x})$, isto é, todas as suas singularidades são essenciais e não apresenta singularidades estranhas. Para as constantes escolhidas, não existe $\mathbf{x}$ tal que $D_{\mathbf{f}}^T \text{sgn}(\mathbf{f}) = \mathbf{0}$ e $\mathbf{f} \neq \mathbf{0}$.

2.7.2 Função de Branin. Constante $c = 0$

A seguir, serão estudadas as bacias de atração dos zeros da função de Branin. Na função de Branin (2.57) foram escolhidas as constantes $a = 1,\ b = 2,\ d = 4\pi,\ e = 0.5,\ f = 2\pi$. A constante c inicialmente será escolhida com o valor 0.

Como já foi analisado, com estas constantes, $\det(D_{\mathbf{f}}(\mathbf{x})) = -1 - 2\pi\cos(2\pi x_1)$, o qual é igual a zero ao longo das retas verticais definidas por $x_1 = \pm 0.2754 \pm k$, para todo $k \in \mathbb{N}$.

Os diferentes algoritmos foram testados dentro de uma janela definida por $x_1 \in [-0.5, 2] \quad x_2 \in [-1, 1]$, dentro da qual se localizam todos os zeros. Os pontos iniciais dentro da janela foram escolhidos a intervalos de 0.025 de cada variável. Os ganhos dos algoritmos foram escolhidos $P = I$, e estes foram testados até um máximo de 250 iterações. Após a apresentação de cada gráfico, conclusões atingidas através da análise das bacias e da equação do algoritmo para esta função serão fornecidas.

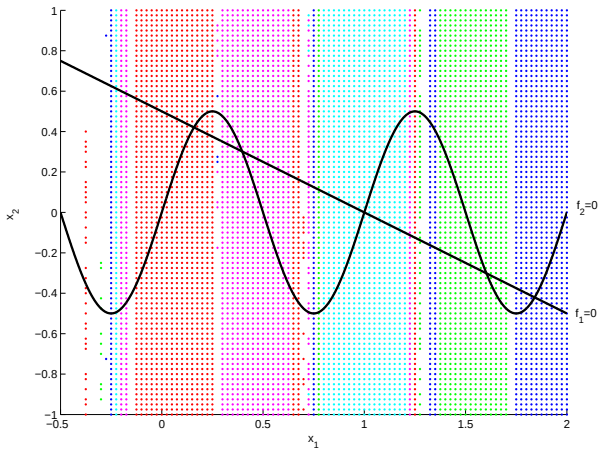

Figura 2.23: Bacias de atração da função de Branin
com $c = 0$ para o algoritmo de Newton

Analisando as bacias e as equações do algoritmo de Newton, conclue-se:

a) $\Delta x_1 = (1 - x_1 - \sin(2\pi x_1))/(1 + 2\pi\cos(2\pi x_1))$, portanto depende exclusivamente de x_1.

b) Eis a razão pela qual as bacias apresentem formas de faixas verticais, similarmente ao que acontece com a função de Camelback.

c) Estas faixas verticais estão encerradas pelo locus $\det(D_{\mathbf{f}}) = 0$.

d) Perto das fronteiras das bacias, por terem as linhas de campo das trajetórias $\Delta\mathbf{x}_k$

uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

e) O algoritmo de Newton não apresenta singularidades estranhas para as constantes escolhidas. Todas as suas singularidades são essenciais e correspondem aos zeros da função.

f) Para $-1+0.2754 < x_1 < 0.2754$, $\Delta x_1 < 0$. Portanto, as trajetórias que começam nesta região possuem uma componente de deslocamento horizontal para a esquerda, onde não existem mais zeros, e podem não retornar à região definida pela janela em menos das 250 iterações testadas, como acontece com a maioria dos pontos iniciais dentro desta faixa (representados por pontos em branco).

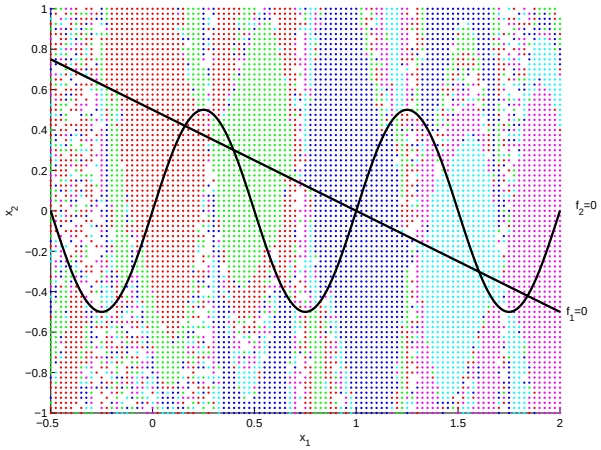

Figura 2.24: Bacias de atração da função de Branin
com $c = 0$ para o algoritmo DNV

a) O algoritmo DNV não apresenta sigularidades estranhas para as constantes escolhidas, todas as suas singularidades são essenciais e correspondem aos zeros da função.

b) As bacias também estão encerradas pelas retas verticais determinadas pelo locus $\det(D_{\mathbf{f}}) = 0$.

c) Dentro das faixas verticais as diferentes bacias estão recortadas por sinusoidais. A diferença deste comportamento com o do algoritmo de Newton se produz porque aqui Δx_1 não depende mais exclusivamente de x_1.

d) Perto das fronteiras de cada bacia, por terem as linhas de campo $\Delta\mathbf{x}_k$ uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

e) As trajetórias que começam em zeros da função de Branin, sofrem uma indeterminação porque, nestes pontos $\Delta\mathbf{x}_k = [0\ 0]^T/0$ (assumindo que $\text{sgn}(0) = 0$). Evidentemente, isto não compromete a eficiência do algoritmo porque estes são exatamente os

pontos finais das trajetórias, não sendo calculada portanto a variável de estado numa próxima iteração.

f) O algoritmo apresenta menos pontos em branco do que o algoritmo de Newton, isto é, possui uma maior quantidade de pontos iniciais a partir dos quais conseguiu atingir um zero em até 249 iterações. Esta evidente vantagem sobre o algoritmo de Newton deve-se a que, longe das retas verticais definidas pelo locus de $\det(D_{\mathbf{f}}) = 0$, o algoritmo NV possui componentes do vetor das linhas de campo $\Delta\mathbf{x}_k$ de valor acotado (o máximo valor é $[3 \quad \pi + 1]^T / \det(D_{\mathbf{f}})$). No algoritmo de Newton, as componentes deste vetor crescem com as variáveis.

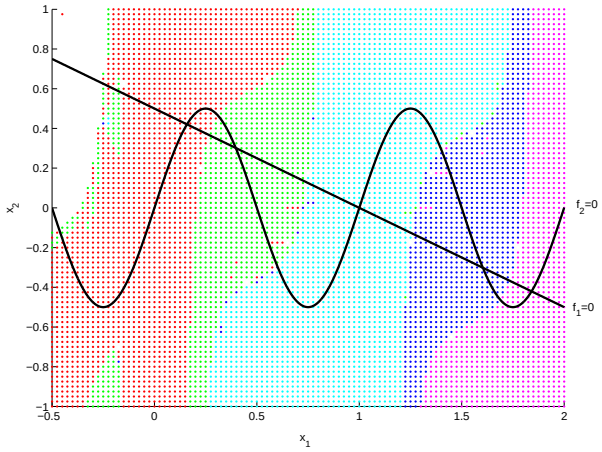

Figura 2.25: Bacias de atração da função de Branin
com $c = 0$ para o algoritmo DJT

a) As bacias estão encerradas por formas geométricas bem determinadas.

b) As formas das bacias estão determinadas pelo locus de $\mathbf{f}^T(D_{\mathbf{f}}D_{\mathbf{f}}^T)\mathbf{f} = 0$, o denominador do comprimento do passo α_k.

c) Perto das fronteiras das bacias, por terem as linhas de campo das trajetórias $\Delta\mathbf{x}_k$ uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

d) Apresenta-se uma região compacta de não convergência no número máximo de iterações testado. Dentro desta região as linhas de campo $\Delta\mathbf{x}_k$ não apresentam um sentido constante, a diferença do que acontece na região de não convergência do algoritmo de Newton.

e) Assim como acontece com a função de Camelback, este algoritmo pode apresentar um comportamento similar ao do algoritmo de Newton diante da presença de singularidades. As singularidades essenciais estariam dadas nos zeros da função, e

os pontos de comportamento similar às singularidades estranhas, nos pontos onde $D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. Foi achado apenas um único ponto com este comportamento $\mathbf{x} = [-0.7251\ 0.7839]^T$. Chama a atenção que a região de não convergência parece rodear este ponto de indeterminação.

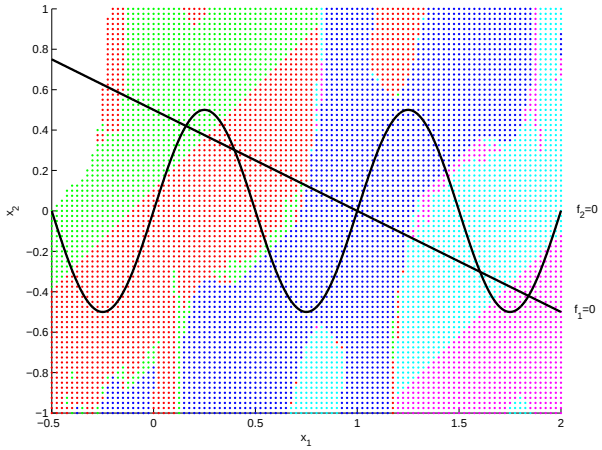

Figura 2.26: Bacias de atração da função de Branin
com $c = 0$ para o algoritmo DVJT

a) O comprimento do passo $\alpha_k > 0$, portanto, os sentidos das linhas de campo $\Delta\mathbf{x}_k = -\alpha_k \text{sgn}(D_{\mathbf{f}}^T\mathbf{f})$ estão no mesmo quadrante que as linhas de campo do algoritmo DJT ($\Delta\mathbf{x}_k = -\alpha_k D_{\mathbf{f}}^T\mathbf{f}$). Isto não implica que, partindo dos mesmos pontos iniciais, os algoritmos convergirão aos mesmos zeros.

b) As bacias apresentam formas geométricas regulares, cujas fronteiras estão determinadas pelo locus de $\text{sgn}^T(D_{\mathbf{f}}^T\mathbf{f})D_{\mathbf{f}}^T D_{\mathbf{f}}\text{sgn}(D_{\mathbf{f}}^T\mathbf{f}) = 0$ (o denominador de α_k).

c) Por ser o comprimento do passo $\alpha_k > 0$ (exceto nos pontos onde $D_{\mathbf{f}}^T\mathbf{f} = \mathbf{0}$), os sentidos das linhas de campo $\Delta\mathbf{x}_k = -\alpha_k[\pm 1 \ \ \pm 1]^T$, a não ser que alguma das componentes do vetor $D_{\mathbf{f}}^T\mathbf{f}$ seja igual a zero, em cujo caso esta componente da linha de campo será igual a zero. Isto justifica que algumas das fronteiras das bacias sejam retas a 45°.

d) Aqui também existe uma região compacta de não convergência, de fronteiras de formas regulares. Esta região é muito similar, embora não idêntica, àquela apresentada pelo algoritmo DJT. Aqui também não há um padrão para os sentidos das linhas de campo dentro desta região.

e) Como acontece com a função de Camelback, os pontos onde o algoritmo DVJT se comporta de maneira similar que diante da presença de singularidades são os mes-

mos que para o algoritmo DJT, isto é, singularidades essenciais nos zeros da função e singularidades estranhas nos pontos onde $D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$, correspondente ao ponto $\mathbf{x} = [-0.7251\ 0.7893]^T$.

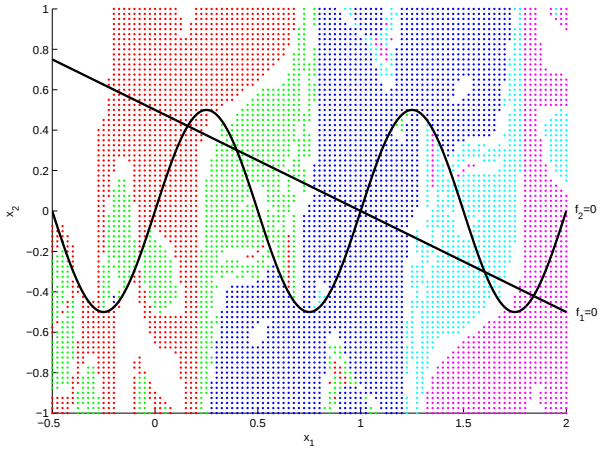

Figura 2.27: Bacias de atração da função de Branin com $c = 0$ para o algoritmo DJTV

a) Este é o único dos cinco algoritmos que pode apresentar comprimento do passo $\alpha_k < 0$, portanto, os sentidos das linhas de campo $\Delta\mathbf{x}_k$ não estão unicamente determinados por $-D_{\mathbf{f}}^T \text{sgn}(\mathbf{f})$.

b) O DJTV é o algoritmo que apresenta maior número de pontos iniciais de não convergência em até 249 iterações. Muitos destes pontos encontram-se próximos das fronteiras das bacias.

c) As formas das bacias são mais irregulares que as dos algoritmos anteriores.

d) O algoritmo apresenta duas regiões maiores de não convergência, sendo que uma delas (a do canto superior esquerdo da figura), é similar, embora não idêntica, às regiões de não convergência dos algoritmos DJT e DVJT.

e) Nestas regiões de não convergência, os sentidos das linhas de campo $\Delta\mathbf{x}_k$ também não são constantes para todos os pontos iniciais.

f) As únicas singularidades existentes são os zeros da função, singularidades essenciais portanto. Não existem pontos tais que $D_{\mathbf{f}}^T(\mathbf{x})\text{sgn}(\mathbf{f}(\mathbf{x})) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. Todavia, podem existir pontos onde a primeira componente do vetor da linha de campo Δx_1 seja igual a zero; a segunda componente Δx_2 só é zero nos zeros da função.

2.7.3 Função de Branin. Constante $c = 0.13$

Nesta seção, aumentamos o valor da constante c para 0.13, similarmente ao realizado por Branin ([18]). Com estas constantes a função continua apresentando cinco zeros. O efeito do aumento desta constante é uma modulação senoidal da reta definida por $f_1 = 0$, como será observado nos gráficos.

A seguir, serão apresentados os gráficos com as bacias de atração de cada zero para cada algoritmo, seguidos de conclusões e análises destas bacias e das equações dos algoritmos.

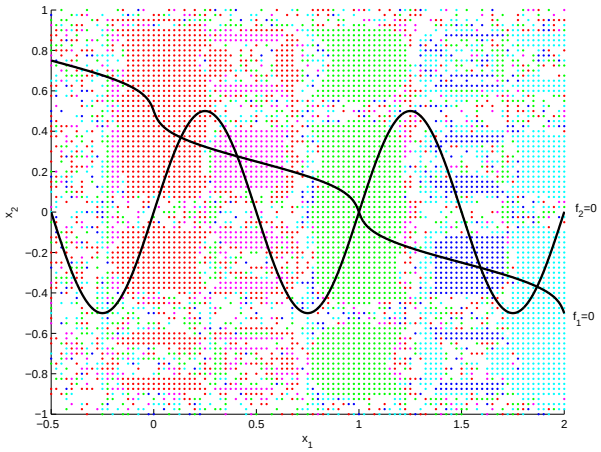

Figura 2.28: Bacias de atração da função de Branin
com $c = 0.13$ para o algoritmo de Newton

a) As bacias de atração estão encerradas dentro do locus determinado por $\det(D_{\mathbf{f}}) = 0$.

b) Perto das fronteiras das bacias, por terem as componentes do vetor das linhas de campo $\Delta\mathbf{x}$ uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

c) A diferença do que acontece com a constante $c = 0$, a primeira componente do vetor de linhas de campo Δx_1 agora não depende exclusivamente de x_1. Eis a razão pela qual as bacias não mais apresentam formas de faixas estritamente verticais.

d) Dentro da região definida por $-1 + 0.2754 < x_1 < 0.2754$, se concentra a maior quantidade de pontos a partir dos quais o algoritmo não conseguiu atingir nenhum zero em até 249 iterações. Porém, a diferença do que acontece com a constante $c = 0$, a primeira componente do vetor de linhas de campo Δx_1 não é mais sempre negativa

dentro desta faixa.

e) Nos pontos próximos às singularidades estranhas, o zero a ser atingido pela trajetória é imprevisível.

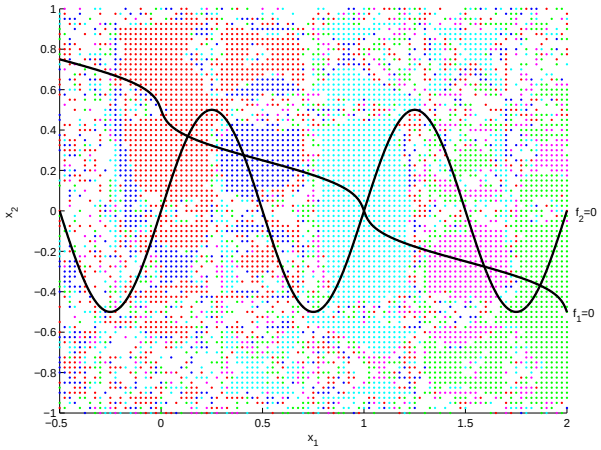

Figura 2.29: Bacias de atração da função de Branin com $c = 0.13$ para o algoritmo DNV

a) As bacias estão encerradas dentro do locus $\det(D_{\mathbf{f}}) = 0$, porém, apresentam formas bem mais irregulares que as apresentadas por este algoritmo com constante $c = 0$.

b) Perto das fronteiras das bacias, por terem as componentes do vetor das linhas de campo $\Delta\mathbf{x}$ uma norma de valor elevado, as trajetórias podem não convergir ao zero mais próximo.

c) A diferença do que acontecia com a constante $c = 0$, agora o número de pontos iniciais a partir dos quais a trajetória não atinge nenhum zero em até 249 iterações é similar à quantidade apresentada pelo algoritmo de Newton, eliminando-se assim uma evidente vantagem deste algoritmo.

d) O algoritmo não apresenta singularidades estranhas dentro da janela definida.

e) As trajetórias só apresentam indeterminações nos zeros da função. Nestes pontos $\mathbf{x}_{k+1} = [0\ 0]^T/0$.

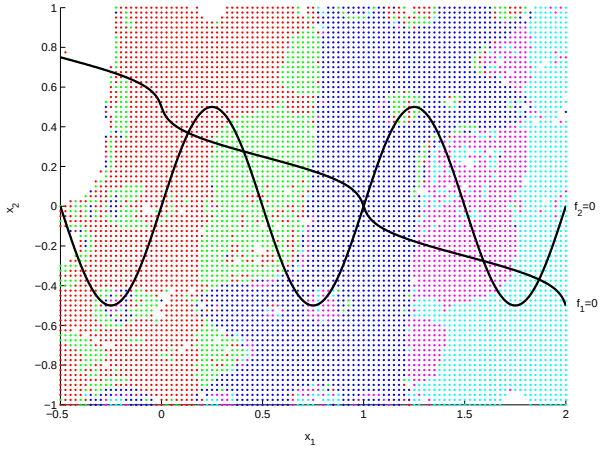

Figura 2.30: Bacias de atração da função de Branin com $c = 0.13$ para o algoritmo DJT

a) As bacias apresentam formas mais regulares que as dos algoritmos anteriores. Estas formas também parecem estar determinadas pelo locus $\det(D_{\mathbf{f}}) = 0$.

b) Perto das fronteiras das bacias, e ao longo de uma área maior do que acontece com a constante $c = 0$, as linhas de campo $\Delta\mathbf{x}$ tem uma norma de valor elevado, razão pela qual as trajetórias podem não atingir o zero mais próximo.

c) Existe uma região de não convergência muito similar, embora não idêntica, àquela apresentada com a constante $c = 0$. Também aqui se verifica que, dentro desta região, as linhas de campo $\Delta\mathbf{x}$ não têm sentidos constantes.

d) Embora as singularidades estranhas não possam ser definidas para este algoritmo, por não haver nenhuma inversão de matriz, determinados pontos provocam que o algoritmo se comporte como diante de tais. Esses pontos são aqueles para os quais $D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$. Nesses pontos as trajetórias são indeterminadas. Foram encontrados os seguintes pontos com estas características $\mathbf{x} = [-1.7360 \;\; 1.2574]^T$, $\mathbf{x} = [-2.7356 \;\; 1.7189]^T$, $\mathbf{x} = [4.6164 \;\; -1.0219]^T$, $\mathbf{x} = [-4.2794 \;\; 1.8882]$ e $\mathbf{x} = [5.3418 \;\; -0.9584]$, todos fora da janela graficada.

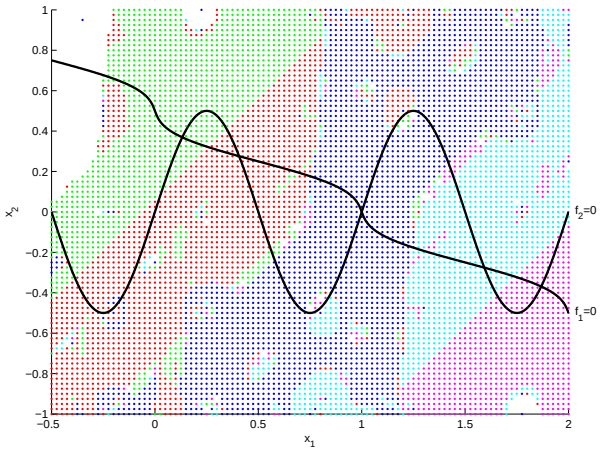

Figura 2.31: Bacias de atração da função de Branin com $c = 0.13$ para o algoritmo DVJT

a) As bacias de atração apresentam formas mais regulares que as dos outros algoritmos. As fronteiras estão determinadas pelo locus de $\text{sgn}^T(D_{\mathbf{f}}^T\mathbf{f})D_{\mathbf{f}}^T D_{\mathbf{f}}\text{sgn}(D_{\mathbf{f}}^T\mathbf{f}) = 0$ (o denominador de α_k).

b) Por ser $\alpha_k > 0$, o sentido das linhas de campo $\Delta\mathbf{x}$ é proporcional a $[\pm 1 \ \pm 1]^T$. Eis a razão pela qual, também aqui, algums das fronteiras das bacias são retas a 45°. Também aqui a exceção se produzirá se alguma componente do vetor de linhas de campo é igual a zero.

c) O algoritmo apresenta, também aqui, os mesmos pontos diante dos quais se comporta como diante de singularidades estranhas do que o algoritmo anterior, isto é, pontos definidos como $D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$ e $\mathbf{f}(\mathbf{x}) \neq \mathbf{0}$, já expressados com o algoritmo anterior.

d) Também aqui os sentidos das linhas de campo $\Delta\mathbf{x}$ estão no mesmo quadrante que as linhas de campo do algoritmo DJT.

e) Aqui também existe uma região de não convergência dentro do número de iterações testado similar à apresentada com a constante $c = 0$. Dentro desta região, o vetor de linhas de campo não apresenta sentido constante para todos os pontos.

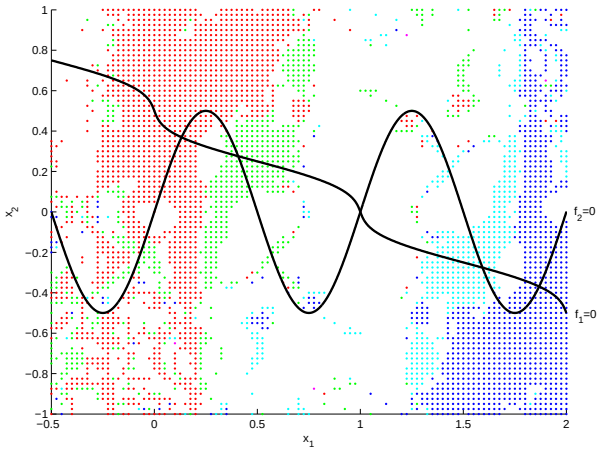

Figura 2.32: Bacias de atração da função de Branin
com $c = 0.13$ para o algoritmo DJTV

a) As bacias de atração apresentam formas totalmente irregulares.

b) Existe uma grande quantidade de pontos iniciais a partir dos quais as trajetórias não conseguiram atingir nenhum zero dentro do número de iterações testado. Embora todos os zeros foram achados a partir de algum ponto inicial testado.

c) O algoritmo continua não apresentando pontos que se comportem como singularidades estranhas, isto é, não existe nenhum ponto tal que $D_{\mathbf{f}}^T(\mathbf{x})\text{sgn}(\mathbf{f}(\mathbf{x})) = 0$ a não ser nos zeros da função (singularides essenciais).

Os algoritmos foram testados também para a mesma função com constante $c = 1$, a qual apresenta 15 zeros; estes gráficos não foram colocados por não mostrar diferenças relevantes com respeito aos apresentados aqui.

2.8 Algoritmos para achar mínimos de funções escalares como sistemas dinâmicos de controle

Como foi mencionado na introdução deste capítulo, o problema de achar os zeros de funções vetoriais inclui o problema de achar o ponto mínimo de uma função escalar convexa diferenciável, pois esse mínimo, quando existe, corresponde com um zero do gradiente, supondo que este está definido em todo o domínio da função.

Definição 2.8.1 *Função convexa ([17, p. 67])*

Uma função escalar $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *é* convexa *se e somente se para todo* $\mathbf{x}_1$, $\mathbf{x}_2 \in \text{dom}(\phi)$ *e para todo escalar* θ *tal que* $0 \leq \theta \leq 1$*:*

$$\phi(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta\phi(\mathbf{x}_1) + (1-\theta)\phi(\mathbf{x}_2) \tag{2.127}$$

A função é estritamente convexa se a desigualdade é estrita para todo $\mathbf{x}_1 \neq \mathbf{x}_2$ *e* $0 < \theta < 1$*.*

Seja $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ uma função contínua, com derivadas parciais contínuas e convexa, e seja $\mathbf{x}^* \in \text{dom}\{\phi(\mathbf{x})\}$ tal que

$$\forall \mathbf{x} \neq \mathbf{x}^* \quad \phi(\mathbf{x}) > \phi(\mathbf{x}^*)$$

chamado ponto mínimo global estrito da função ϕ, os algoritmos provocam que as variáveis de estado $\mathbf{x}(t)$ descrevam uma trajetória a partir de um ponto inicial $\mathbf{x}(0) = \mathbf{x}_0$ até, caso sejam convergentes, o mínimo global da função $\mathbf{x}^*$.

Embora os algoritmos de primeira ordem apresentados possam ser utilizados na resolução deste problema, pois eles acham o único zero da função $\nabla\phi(\mathbf{x})$, que é o mínimo global, existem outros algoritmos específicos para achar o ponto mínimo de funções escalares. Dentre eles, o mais mencionado na bibliografia é o *steepest descent* ([79, c. 5]). Este consiste em estabelecer as trajetórias na direção contrária do gradiente da função, garantindo assim o decrescimento desta.

$$\dot{\mathbf{x}} = -\kappa\nabla\phi(\mathbf{x}) \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.128}$$

onde κ é um ganho escalar positivo.

Observe-se que para ϕ convexa, as curvas de nível definidas pelo contorno de:

$$S_\alpha = \{\mathbf{x} \mid \phi(\mathbf{x}) \leq \alpha\} \tag{2.129}$$

definem conjuntos convexos e aninhados, isto é, sejam dois escalares α_1 e α_2, se $\alpha_1 < \alpha_2$, então $S_{\alpha_1} \subset S_{\alpha_2}$. As trajetórias definidas pelo sistema (2.128) atravessarão as curvas de nível $\mathbf{bd}\{S_\alpha\}$ perpendicularmente, até chegar ao ponto mínimo da função, caso este exista.

Caso a função $\phi(\mathbf{x})$ não seja convexa, e apresente diversos máximos e mínimos locais, as trajetórias geradas pelo algoritmo poderão se estacionar em um mínimo local ou ainda em uma sela, isto é, em qualquer ponto tal que $\nabla\phi(\mathbf{x}) = \mathbf{0}$. Em um máximo local só se estacionará se a trajetória partir deste ponto.

Um outro algoritmo utilizado para este fim é o de Lotka-Volterra, o qual consiste em um *steepest descent* com um ganho variável com a variável de estado.

$$\dot{x}_i(t) = -\kappa x_i(t)\frac{\partial\phi(\mathbf{x}(t))}{\partial x_i} \quad \forall i \in \{1,..,n\},\ \mathbf{x}(0) = \mathbf{x}_0 \tag{2.130}$$

onde κ é um ganho escalar positivo.

Attouch e Teboulle ([6]) apresentaram uma versão chamada de "regularizada" para o algoritmo de Lotka-Volterra, a qual, segundo a escolha de duas constantes, as trajetórias geradas pelo sistema se aproximam mais daquelas determinadas pelo algoritmo *steepest descent*, ou pelo algoritmo de Lotka-Volterra, o que justifica o nome desta versão.

$$\dot{x}_i(t) = -\frac{x_i(t)}{\mu + \nu x_i(t)}\frac{\partial\phi(\mathbf{x}(t))}{\partial x_i} \quad \forall i \in \{1,..,n\},\ \ \mathbf{x}_0 \in \mathbb{R}^n_{++} \tag{2.131}$$

sendo μ e ν ganos escalares positivos. Observe-se que, para $\nu > 0$ e $\mu \to 0$, o sistema se aproxima ao *steepest descent*, e para $\mu > 0$ e $\nu \to 0$, o sistema se aproxima ao de Lotka-Volterra.

Attouch e Cominetti ([5]) apresentaram uma versão parametrizada do algoritmo *steepest descent*, onde admitem a possibilidade da função a ser minimizada ser descontínua.

$$\mathbf{0} \in \dot{\mathbf{x}} + \partial\phi(\mathbf{x}(t), \epsilon(t)) \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.132}$$

onde ϵ é um parâmetro variável, $\{\phi(\cdot, \epsilon), \epsilon > 0\}$ é um conjunto de estritamente convexas funções com mínimo único $\mathbf{x}(\epsilon)$, e $\partial\phi(\mathbf{x}(t), \epsilon(t))$ é o gradiente generalizado de Clarke da função escalar parametrizada.

Assumindo que $\lim_{\epsilon\to 0}\mathbf{x}(\epsilon) = \mathbf{x}^*$, [5] derivam condições suficientes do parâmetro $\epsilon(t)$ que garantem $\lim_{t\to\infty}\mathbf{x}(t) = \mathbf{x}^*$.

A continuação, serão apresentados os algoritmos (2.128) e (2.131) mencionados, como sistemas dinâmicos de controle em malha fechada, além de derivar outros algoritmos utilizando o método de funções de controle de Liapunov.

2.8.1 Algoritmos contínuos para achar mínimos de funções

Os algoritmos destinados a achar mínimos de funções escalares podem ser vistos como sistemas dinâmicos de controle em malha fechada, os quais podem ser representados com o seguinte diagrama de blocos:

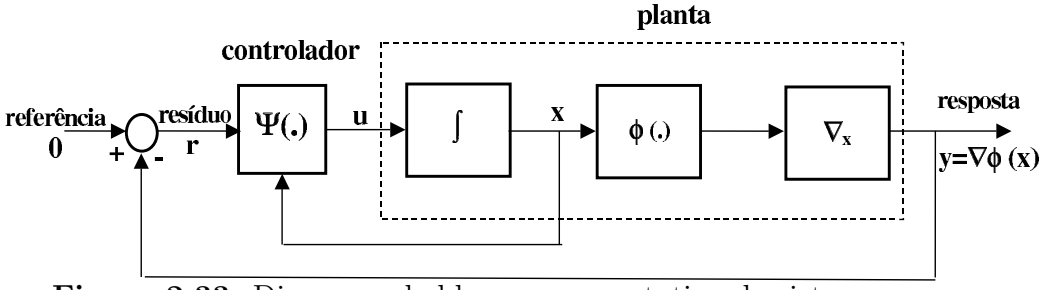

Figura 2.33: Diagrama de blocos representativo do sistema

O controlador é definido pela função $\Psi(\mathbf{x})$. O resíduo ou erro a ser minimizado é definido como

$$\mathbf{r} = -\nabla\phi(\mathbf{x}) \tag{2.133}$$

pois o objetivo é achar o ponto de gradiente zero da função.

A lei de atualização das variáveis de estado estará dada por:

$$\dot{\mathbf{x}} = \mathbf{u} = \Psi(\mathbf{x})\mathbf{r} = -\Psi(\mathbf{x})\nabla\phi(\mathbf{x}) \tag{2.134}$$

Assumindo a função $\phi(\mathbf{x})$ convexa, e assumindo que existe um ponto mínimo $\mathbf{x}^*$ finito, pode se escolher como função de Liapunov candidata

$$V(\mathbf{x}) = \phi(\mathbf{x}) - \phi(\mathbf{x}^*) \tag{2.135}$$

a cual é positiva definida e só apresenta o valor zero em $\mathbf{x} = \mathbf{x}^*$, e cuja derivada temporal é

$$\dot{V}(\mathbf{x}) = \nabla^T\phi(\mathbf{x})\dot{\mathbf{x}} \tag{2.136}$$

Este é o ponto de partida para a dedução dos diversos algoritmos para a minimização de funções escalares convexas, escolhendo a lei de atualização das variáveis $\dot{\mathbf{x}}$ que faz a derivada temporal da função de Liapunov negativa definida, escolhas definidas como funções de Liapunov de controle.

1) Primeira escolha

A primeira escolha será aquela correspondente ao algoritmo *steepest descent* (SD), o qual foi apresentado na equação (2.128). Aqui o controlador $\Psi(\mathbf{x})$ é uma constante κ.

Substituindo (2.128) em (2.136):

$$\dot{V}(\mathbf{x}) = -\kappa\|\nabla\phi(\mathbf{x})\|^2 \leq 0$$

a qual é negativa definida e só toma o valor zero em um ponto tal que $\nabla\phi(\mathbf{x}) = \mathbf{0}$, isto é, no mínimo da função $\mathbf{x}^*$. Pelo lema de Barbalat ([69, s. 4.5]), por ser $V(\mathbf{x})$ contínua e continuamente diferenciável, então $\dot{V}(\mathbf{x}) \to 0$ quando $t \to \infty$ e portanto as trajetórias tendem ao ponto de gradiente zero da função.

Observe-se que também poderia ser escolhida como função de Liapunov candidata

$$V(\mathbf{x}) = \mathbf{r}^T\mathbf{r}$$

a qual é positiva definida e só apresenta valor zero em um zero do gradiente, isto é, em $\mathbf{x} = \mathbf{x}^*$ supondo este o único ponto de gradiente zero da função. Derivando com respeito ao tempo:

$$\Rightarrow\ \dot{V}(\mathbf{x}) = 2\mathbf{r}^T\dot{\mathbf{r}} = -2\kappa\nabla^T\phi(\mathbf{x})\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}) \leq 0$$

sendo $\nabla^2\phi(\mathbf{x})$ o hessiano da função, o qual é simétrico e positivo definido por ser a função convexa. Portanto $\dot{V}(\mathbf{x})$ é negativa definida, e só toma o valor zero no ponto de gradiente zero, o qual prova que o algoritmo é assintoticamente convergente ao ponto mínimo.

2) Segunda escolha

A nossa segunda escolha consiste no sistema (2.131), Lotka-Volterra regularizado (LVR), apresentado em [6]. Este sistema pode ser visto como

$$\dot{\mathbf{x}} = -\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x}) \quad \mathbf{x}_0 \in \mathbb{R}^n_{++} \tag{2.137}$$

onde

$$\psi(\mathbf{x}) := \frac{\nu}{2}\|\mathbf{x}\|^2 + \mu \sum_{i=1}^{n} x_i \log x_i$$

e $\nabla^{-2}\psi(\mathbf{x})$ é a inversa do hessiano da função $\psi(\mathbf{x})$, sendo esta matriz o ganho do controlador do sistema em malha fechada $\Psi(\mathbf{x})$.

Observe-se que se a constante μ for zerada, $\nabla^{-2}\psi(\mathbf{x}) = \nu^{-1}I$ e o sistema corresponde com o *steepest descent* com ganho $\kappa = \nu^{-1}$.

Substituindo (2.137) em (2.136):

$$\dot{V}(\mathbf{x}) = \nabla^T\phi(\mathbf{x})\dot{\mathbf{x}} = -\nabla^T\phi(\mathbf{x})\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x})$$

a qual é negativa definida para qualquer matriz $\nabla^{-2}\psi(\mathbf{x}) \succ 0$. Sob esta condição, pelo lema de Barbalat prova-se que as trajetórias são assintoticamente convergentes.

Observe-se que $\nabla^{-2}\psi(\mathbf{x}) = \text{diag}\left(\frac{x_i}{\mu+\nu x_i}\right)$, não pode ser *a priori* considerada uma matriz positiva definida, pois para valores $x_i \in (-\mu/\nu; 0]$, $x_i \leq 0$ e portanto o elemento i na matriz diagonal é não positivo, não sendo a matriz positiva definida. Porém, Attouch e Teboulle ([6, teorema 3.1]), demonstram que se a trajetória começa no ortante positivo ($\mathbf{x}(0) \succ 0$), permanece nesse ortante para todo $t > 0$, e portanto a matriz permanece positiva definida.

Teorema 2.8.2 *Seja $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ uma função $\in C^2$, a qual é limitada inferiormente em $\mathbb{R}^n_+$, para todo $\mathbf{x}(0) \in R^n_{++}$, existe uma única solução $\mathbf{x} : [0; \infty) \to \mathbb{R}^n$ e $\mathbf{x}(t) \succ 0\ \forall t \geq 0$.*

Prova:

Se $\mathbf{x}(0) \succ 0$, então existe $\tau > 0$ tal que $\mathbf{x}(t) \succ 0\ \forall t \in [0; \tau)$. Por ser $\phi \in C^1$, existe $C > 0$ tal que $\|\frac{\partial\phi(\mathbf{x})}{\partial x_j}\| \leq C$.

Portanto:

$$0 = \dot{x}_j + \frac{x_j}{\mu + \nu x_j}\frac{\partial\phi(\mathbf{x})}{\partial x_j} \leq \dot{x}_j + \frac{x_j}{\mu + \nu x_j}C \leq \dot{x}_j + \frac{C}{\mu}x_j \quad \forall j \in \{1, .., n\}\ \forall t \in [0; \tau)$$

$$\Rightarrow x_j(t) \geq x_j(0)e^{-\frac{C}{\mu}t} \quad \forall j \in \{1, .., n\}\ \forall t \in [0, \tau)$$

$$\Rightarrow x_j(t) > 0 \quad \forall j \in \{1, .., n\}\ \forall t > 0 \tag{2.138}$$

Ver detalhes da prova em [6, teorema 3.1].

Portanto, a função de Liapunov candidata apresenta uma derivada com respeito ao tempo negativa definida e o algoritmo só se estacionará no ponto onde $\nabla\phi(\mathbf{x}) = \mathbf{0}$, isto é, no mínimo global $\mathbf{x}^*$, caso este se encontre no ortante não negativo.

A desvantagem deste algoritmo consiste em que se o objetivo for achar o mínimo global da função escalar $\phi(\mathbf{x})$, este deve se encontrar no ortante não negativo $\mathbb{R}^n_+$, e será alcançado por qualquer trajetória iniciada no ortante positivo $\mathbb{R}^n_{++}$. Caso contrário, as trajetórias que começam no ortante positivo convergirão ao ponto mínimo dentro ou no limite deste ortante. Este caso pode se considerar como um caso particular de resolução de um problema de otimização convexa, consistente em minimizar uma função objetivo escalar convexa sujeita a restrições de desigualdade afins, que seriam os limites inferiores das variáveis de estado iguais a zero.

$$\begin{aligned} \min \quad & \phi(\mathbf{x}) \\ \text{s.t.} \quad & x_i \geq 0 \ \ \forall i \in \{1, .., n\} \end{aligned}$$

3) Terceira escolha

Observe-se que outras funções de ganho variável podem ser utilizadas como controlador LVR, sempre que a matriz de ganho do controlador $\Psi(\mathbf{x})$ seja positiva definida.

Por exemplo, a função $\psi(\mathbf{x}) = -\sum_{i=1}^{n} \log x_i$, tem uma inversa do hessiano igual a $\nabla^{-2}\psi(\mathbf{x}) = \text{diag}\,(x_i^2)$ para todo $i \in \{1, .., n\}$. Esta matriz é positiva semi-definida, e só toma o valor zero em $\mathbf{x} = \mathbf{0}$. Portanto, cada componente do vetor de atualização das variáveis, $\dot{x}_i(t)$, só tomará o valor zero em $x_i = 0$, ou em $\nabla_i\phi(\mathbf{x}) = 0$, isto é, em x_i^*. Portanto, a trajetória só pode se estacionar em um ponto $\mathbf{x}$ tal que $x_i = 0$ ou $x_i = x_i^*$ para todo $i \in \{1, .., n\}$.

Isto implica que se o mínimo da função, $\mathbf{x}^*$, estiver no mesmo ortante (ou no limite) que $\mathbf{x}_0$, o algoritmo será convergente. Caso contrário, a trajetória convergirá ao valor mínimo da função dentro (ou no limite) do ortante da posição inicial, podendo ser considerado este também um problema particular de otimização convexa.

Portanto, o sistema gradiente descendente com este ganho seria assintoticamente estável para esta condição inicial particular.

Esta é a nossa terceira escolha:

$$\dot{\mathbf{x}} = -\kappa \nabla^{-2} \psi(\mathbf{x}) \nabla \phi(\mathbf{x}) \tag{2.139}$$

onde $\psi(\mathbf{x}) = -\sum_{i=1}^{n} \log x_i$ e κ é um ganho escalar positivo. Este algoritmo será chamado de Lotka-Volterra quadrado (LV2) pois o ganho de cada componente do vetor $\dot{\mathbf{x}}$ é justamente o ganho do algoritmo de Lotka-Volterra ao quadrado, x_i^2 para todo $i \in \{1, .., n\}$, vezes o escalar κ.

A prova da convergência é inteiramente similar ao caso anterior, sendo que neste caso $\nabla^{-2}\psi(\mathbf{x}) \succ 0$, para todo $\mathbf{x} \in \mathbb{R}^n$ tal que $x_i \neq 0$, para todo $i \in \{1, .., n\}$, existindo a condição, portanto, do ponto inicial da trajetória estiver no mesmo ortante que o mínimo da função.

Observe-se também que a escolha $\psi(\mathbf{x}) = \phi(\mathbf{x})$, para $\phi(\mathbf{x})$ estritamente convexa, também teria a inversa do hessiano positiva definida, $\nabla^{-2}\psi(\mathbf{x}) \succ 0$, e esta matriz poderia ser utilizada como ganho do controlador $\Psi(\mathbf{x})$, mas este não seria outro que o algoritmo de Newton (equação (2.15)).

A escolha como matriz de ganho do controlador $\Psi(\mathbf{x}) = \nabla^2 \phi(\mathbf{x})$, também seria positiva definida para uma função escalar $\phi(\mathbf{x})$ estritamente convexa, e portanto o algoritmo seria convergente. Mas lembrando que o hessiano da função escalar é simétrico, esta escolha não seria outra o o algortimo CJT, apresentado na equação (2.26).

Finalmente, podemos mencionar que qualquer outro ganho variável, porém positivo definido, forneceria um sistema em malha fechada estável. Um exemplo poderia ser $\Psi(\mathbf{x}) = \text{diag}\,(e^{|x_i|})$, para todo $i \in \{1, .., n\}$, ou $\Psi(\mathbf{x}) = \text{diag}\,(\cosh(x_i))$, para todo $i \in \{1, .., n\}$.

4) Quarta escolha

Finalmente, uma escolha natural impõe um algoritmo a estrutura variável:

$$\dot{\mathbf{x}} = -\kappa \, \text{sgn}(\nabla \phi(\mathbf{x})) \tag{2.140}$$

onde κ é um ganho escalar positivo. Aqui o controlador é a função $\kappa \, \text{sgn}(\cdot)$. Este

sistema será chamado de *steepest descent* a estrutura variável (VSD).

Substituindo (2.140) em (2.136):

$$\dot{V}(\mathbf{x}) = \nabla^T \phi(\mathbf{x}) \dot{\mathbf{x}} = -\kappa \|\nabla \phi(\mathbf{x})\|_1 \leq 0$$

o qual é negativa definida e só toma o valor zero no ponto tal que $\nabla\phi(\mathbf{x}) = \mathbf{0}$, isto é, no mínimo da função, sendo portanto este algoritmo também assintoticamente convergente.

A seguinte tabela resume as escolhas apresentadas.

$\mathbf{u}(\mathbf{r})$	$\Psi(\mathbf{x})$	**função** $\psi(\mathbf{x})$ (escolhas 2 e 3)	nome	número de equação
$-\kappa\nabla\phi(\mathbf{x})$	κ		SD	(2.128)
$-\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x})$	$\nabla^{-2}\psi(\mathbf{x})$	$\frac{\nu}{2}\|\mathbf{x}\|^2 + \mu\sum_{i=1}^n x_i \log x_i$	LVR	(2.131)
$-\kappa\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x})$	$\kappa\nabla^{-2}\psi(\mathbf{x})$	$-\sum_{i=1}^n \log x_i$	LV2	(2.139)
$-\kappa\ \mathrm{sgn}(\nabla\phi(\mathbf{x}))$	$-\kappa\ \mathrm{sgn}(\cdot)$		VSD	(2.140)

Tabela 2.5 Algoritmos contínuos para minimização de funções escalares derivados com a metodologia CLF/LOC

2.8.1.1 Simulações dos algoritmos contínuos

A seguir, serão implementadas simulações dos algoritmos apresentados. As simulações foram realizadas com o aplicativo Simulink do programa Matlab 6, discretizando as equações diferenciais pelo método de Euler, com comprimento do passo fixo igual a 0.01s. e simulando durante 10 segundos.

Primeiro será utilizada a função de Rosenbrock (2.54) a partir de vários pontos iniciais. As constantes da função foram escolhidas $a = 0.5$ e $b = 1$. Com estas constantes a função apresenta um mínimo global em $\mathbf{x}^* = [1\ 1]^T$. Observe-se que a função de Rosenbrock não é convexa para as constantes escolhidas, porém, isto não afeta o desempenho dos algoritmos contínuos.

Os pontos iniciais das trajetórias escolhidos foram $\mathbf{x}_0 = [-0.5\ 2]^T$, $\mathbf{x}_0 = [-1\ -0.5]^T$, $\mathbf{x}_0 = [1\ 3]^T$, $\mathbf{x}_0 = [1.5\ -0.5]^T$, $\mathbf{x}_0 = [1.5\ 0.5]^T$ e $\mathbf{x}_0 = [0.2\ 2.5]^T$.

Os algoritmos Lotka-Volterra regularizado (LVR) e Lotka-Volterra quadrado (LV2), só foram testados a partir dos pontos iniciais localizados no ortante positivo, o ortante onde se localiza o mínimo da função, pois caso contrário as trajetórias seriam divergentes ou se estacionariam no ponto mínimo dentro do ortante da posição inicial.

O ganho κ dos sistemas SD, LV2, e VSD foi mantido unitário; a constante μ do sistema LVR também foi mantida unitária, ao tempo que a constante ν recebeu primeiramente o valor 1 e posteriormente o valor 0.1.

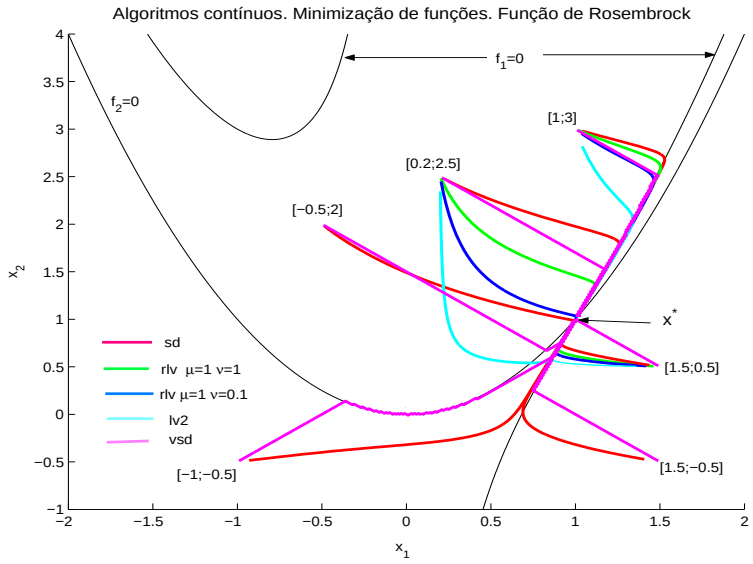

Figura 2.34: Algoritmos contínuos para achar mínimos de funções implementado com a função de Rosenbrock mostrando a convergência das trajetórias

Todos os algoritmos foram capazes de achar o mínimo global a partir dos pontos iniciais testados. Observe-se as trajetórias em modo deslizante geradas pelo algoritmo VSD.

A continuação serão testados os algoritmos com a função apresentada no exemplo 1 do trabalho de Attouch e Teboulle [6]. A função exemplificada é dada por

$$\phi(\mathbf{x}) = \frac{1}{2}\left[(x_1 + x_2 + 1)^2 + 4(x_1 - x_2 - 1)^2\right] \tag{2.141}$$

Esta função tem um gradiente linear, e portanto um hessiano constante, e apresenta um único mínimo global em $\mathbf{x}^* = [0 \ -1]^T$.

Observe-se que este mínimo não corresponde ao primeiro quadrante, estando o mínimo deste quadrante localizado em $\mathbf{x} = [3/5 \ 0]^T$. No segundo quadrante o mínimo está localizado na origem de coordenadas.

Os pontos iniciais testados foram $\mathbf{x}_0 = [-1 \ 0.5]^T$, $\mathbf{x}_0 = [1.5 \ 0.5]^T$, $\mathbf{x}_0 = [-1 \ -2]^T$, e $\mathbf{x}_0 = [1 \ -2]^T$, um em cada quadrante, sendo que o algoritmo LVR só foi testado a partir do ponto inicial localizado no primeiro quadrante.

Aqui também, o ganho κ dos sistemas SD, LV2, e VSD foi mantido unitário; a constante μ do sistema LVR também foi mantida unitária, ao tempo que a constante ν recebeu primeiramente o valor 1 e posteriormente o valor 0.1.

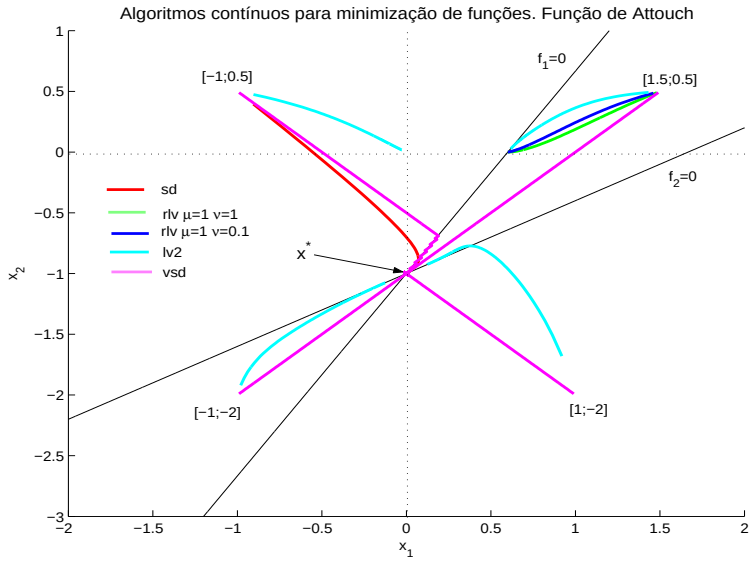

Figura 2.35: Algoritmos contínuos para achar mínimos de funções implementado com a função de Attouch mostrando a convergência das trajetórias

Observe-se que as trajetórias geradas pelo algoritmo LVR convergem ao mínimo restrito ao primeiro quadrante $\mathbf{x} = [3/5\ 0]^T$, assim como a trajetória gerada pelo algoritmo LV2 desde igual posição inicial. O algoritmo LV2 iniciado no segundo quadrante, converge ao mínimo restrito a este quadrante, localizado na origem de coordenadas, $\mathbf{x} = [0\ 0]^T$. As trajetórias geradas a partir do terceiro e quarto quadrante convergem ao mínimo global, por se encontrar este na fronteira entre ambos quadrantes. Os algoritmos SD e VSD encontram o mínimo global a partir de todos os pontos iniciais testados.

O algoritmo LVR, portanto, só serve para achar mínimos de funções convexas restritas ao ortante não negativo, o qual constitui um problema particular de otimização convexa. Os algoritmos propostos aqui, entretanto, são capazes de achar o mínimo dentro de ortantes não positivos (LV2) ou ainda o mínimo global, independentemente do ortante onde este se encontre (SD e VSD).

2.8.2 Discretização dos algoritmos

A seguir, os algoritmos apresentados na seção anterior serão discretizados pelo método de Euler, com seus comprimentos dos passos escolhidos de maneira ótima a fim de minimizar uma função de Liapunov candidata discreta, segundo a metodologia de controle ótimo de Liapunov.

Chamado $\mathbf{u}_k$ à lei de atualizaçao das variáveis no instante k, de maneira tal que

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k$$

definindo o resíduo ou erro discreto do sistema como em (2.133) no instante k

$$\mathbf{r}_k = -\nabla\phi(\mathbf{x}_k)$$

de maneira tal que na seguinte iteração, linearizando a expressão por Taylor,

$$\mathbf{r}_{k+1} \simeq -\nabla\phi(\mathbf{x}_k) - \alpha_k \nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k$$

Escolhendo como função de Liapunov candidata discreta

$$V_k = \mathbf{r}_k^T \mathbf{r}_k$$

a qual é positiva definida, a variação desta função a cada iteração é dada por:

$$\Delta V_k = V_{k+1} - V_k = 2\alpha_k \nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k + \alpha_k^2 \|\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k\|^2 \tag{2.142}$$

Derivando esta equação com respeito a α_k e igualando a zero, obtemos o seguinte comprimento do passo:

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k}{\|\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k\|^2} \tag{2.143}$$

Em seguida serão aplicados estes resultados para cada um dos algoritmos apresentados na seção anterior.

1) Primeira escolha

No algoritmo *steepest descent*, a lei de atualização das variáveis (2.128) discretizada

por Euler é dada por:

$$\mathbf{u}_k := -\nabla\phi(\mathbf{x}_k) \tag{2.144}$$

a partir de um ponto inicial $\mathbf{x}_0$.

Substituindo esta equação em (2.143), obtemos o seguinte comprimento do passo ótimo:

$$\alpha_k = \frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)}{\|\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2} \tag{2.145}$$

o qual é, caso a função $\phi(\mathbf{x})$ seja estritamente convexa, e portanto o hessiano positivo definido, sempre não negativo.

A variação da função de Liapunov candidata a cada iteração, substituindo (2.144) e (2.145) em (2.142):

$$\Delta V_k = -\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))^2}{\|\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2} < 0$$

o qual garante o decrescimento do resíduo a cada iteração e portanto a convergência do algoritmo.

Cada ponto na seqüência, portanto, será calculado

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\nabla\phi(\mathbf{x}_k) \tag{2.146}$$

Observe-se que também poderia ter sido escolhido a seguinte função de Liapunov candidata discreta

$$V_k = (\phi(\mathbf{x}) - \phi(\mathbf{x}^*))^2$$

a qual é positiva definida e cuja variação a cada iteração é dada por:

$$\begin{aligned}\Delta V_k = V_{k+1} - V_k &= (\phi(\mathbf{x}_{k+1}) - \phi(\mathbf{x}^*))^2 - (\phi(\mathbf{x}_k) - \phi(\mathbf{x}^*))^2 \\ &= -2\alpha_k(\phi(\mathbf{x}_k) - \phi(\mathbf{x}^*))\|\nabla\phi(\mathbf{x})\|^2 + \alpha_k^2\|\nabla\phi(\mathbf{x}_k)\|^4\end{aligned}$$

e derivando com respeito a α_k:

$$\frac{\partial V_k}{\partial \alpha_k} = -2(\phi(\mathbf{x}_k) - \phi(\mathbf{x}^*))\|\nabla\phi(\mathbf{x}_k)\|^2 + 2\alpha_k\|\nabla\phi(\mathbf{x}_k)\|^4 = 0$$

$$\Rightarrow \ \alpha_k = \frac{\phi(\mathbf{x}_k) - \phi(\mathbf{x}^*)}{\|\nabla\phi(\mathbf{x}_k)\|^2}$$

e

$$\Delta V_k = -(\phi(\mathbf{x}_k) - \phi(\mathbf{x}^*))^2 < 0 \quad \forall \mathbf{x} \neq \mathbf{x}^*$$

o que garante o decrescimento global da função de Liapunov. Mas para calcular o comprimento do passo seria preciso conhecer o valor da função no ponto mínimo, que é justamente o valor que pretende ser achado com o algoritmo, e portanto *a priori* desconhecido.

Para solucionar este problema é possível substituir $\phi(\mathbf{x}^*)$ por um valor γ estimado tal que $\gamma < \phi(\mathbf{x}^*)$, como utilizado em [61], ou, como realizado aqui, utilizar $V_k = \mathbf{r}_k$.

Observe-se que, se a função $\phi(\mathbf{x})$ não for convexa, o hessiano não é positivo definido, e a convergência do algoritmo não pode ser garantida. Por exemplo, o algoritmo não está definido nos pontos tais que $\nabla\phi(\mathbf{x}_k) \in \mathcal{N}(\nabla^2\phi(\mathbf{x}_k))$, que inviabiliza o cálculo do comprimento do passo. Os pontos tais que $\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}) \neq \mathbf{0}$ mas $\nabla^T\phi(\mathbf{x})\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}) = 0$ zeram o comprimento do passo α, e portanto o algoritmo estaciona-se nestes pontos, mesmo que $\mathbf{x}_k \neq \mathbf{x}^*$.

Isto é o que acontece com a função de Rosenbrock, com as constantes $a = 0.5$ e $b = 1$, no ponto $\mathbf{x} = [-0.9776\ 2.6979]^T$.

Se a função for quaseconvexa, é possível que existam pontos onde $\nabla^2\phi(\mathbf{x}) \prec 0$, e portanto o comprimento do passo α_k será negativo; neste caso, a trajetória será descrita no sentido do gradiente ascendente, tendendo a máximos locais, mas podendo também se estacionar em selas.

2) Segunda e terceira escolha

Ambos casos serão tratados conjuntamente por serem inteiramente similares, mudando apenas a expressão da função do controlador $\Psi(\mathbf{x})$.

A lei de atualização das variáveis é dada por

$$\mathbf{u}_k := -\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k) \tag{2.147}$$

e substituindo em (2.143), obtemos um comprimento do passo:

$$\alpha_k = \frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2} \tag{2.148}$$

Este comprimento do passo é não negativo para funções estritamente convexas; no caso do algoritmo LVR, o comprimento do passo permanece positivo para trajetórias iniciadas no primeiro ortante. Observe-se que, tanto para o algoritmo LVR como para LV2, o comprimento do passo é indeterminado no ponto $\mathbf{x}_k = \mathbf{0}$.

Substituindo este comprimento do passo e a lei de atualização das variáveis em (2.142):

$$\Delta V_k = -\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))^2}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2} \leq 0$$

garantindo assim o decrescimento da função de Liapunov candidata, ao menos localmente, e portanto a convergência do algoritmo.

Observe-se que, para ambos algoritmos, o cálculo do comprimento do passo seria indeterminado naqueles pontos tais que $\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^2\phi(\mathbf{x})\nabla^{-2}\psi(\mathbf{x}))$, ou $\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^2\phi(\mathbf{x}))$ ao tempo que, naqueles pontos onde

$\nabla^2\phi(\mathbf{x})\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x}) \neq \mathbf{0}$, mas $\nabla^T\phi(\mathbf{x})\nabla^2\phi(\mathbf{x})\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x}) = 0$, as trajetórias se estacionam pois esses pontos zeram o cálculo do comprimento do passo α_k, mesmo não se localizando em um ponto extremo da função $\mathbf{x}^*$.

Isto é o que acontece com a função de Attouch, para o algoritmo LV2, nos pontos $\mathbf{x} = [-0.1979 \quad -1.1723]^T$, $\mathbf{x} = [-0.4191 \quad -1.3439]^T$, $\mathbf{x} = [-0.1887 \quad -0.8378]^T$, $\mathbf{x} = [0.8583 \quad -0.0269]^T$, por exemplo.

Cada ponto na sequência, portanto, deverá ser calculado como

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k) \tag{2.149}$$

com a escolha de $\psi(\mathbf{x})$ correspondente a cada algoritmo.

Observe-se que aqui também, para o algoritmo LVR, se forem escolhidas as cons-

tantes $\mu = 0$ e $\nu = 1$, então $\nabla^{-2}\psi(\mathbf{x}) = I$ e o sistema LVR (2.149) corresponde com o *steepest descent* (2.146), e os comprimentos dos passos (2.148) e (2.145) também coincidem.

No caso do algoritmo de Lotka-Volterra regularizado, Attouch e Teboulle ([6]) apresentam uma discretização baseada no método "proximal-like", a qual consiste em escolher um ponto da sequência como:

$$\mathbf{x}_k = \arg\min_{\mathbf{x}\in\mathbb{R}^n}\{\phi(\mathbf{x}) + \lambda_k^{-1} d(\mathbf{x} - \mathbf{x}_{k-1})\}$$

onde λ_k é uma sequência de números positivos e $d(\mathbf{x} - \mathbf{y})$ é uma função de distância que responde a determinadas propriedades ([6, p. 544]). Em particular, nesse trabalho é escolhida:

$$d(\mathbf{x}, \mathbf{y}) := \frac{\nu}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \mu \sum_{i=1}^{n} y_i \varphi\left(\frac{x_j}{y_j}\right)$$

onde

$$\varphi(r) := r - \log r - 1$$

Porém, a própria escolha do ponto seguinte na sequência consiste em um problema de otimização, que os autores não esclarecem como resolver, assim como não apresentam nenhuma simulação do algoritmo discreto.

3) Quarta escolha

No algoritmo *steepest descent* a estrutura variável (VSD), a lei de atualização das variáveis define-se como:

$$\mathbf{u}_k := -\text{sgn}(\mathbf{x}_k) \tag{2.150}$$

Substituindo em (2.143):

$$\alpha_k = \frac{\nabla^T \phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k))}{\|\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k))\|^2} \tag{2.151}$$

Observe-se que nada se pode afirmar com respeito ao sinal deste comprimento do passo; porém, a variação da função de Liapunov candidata discreta a cada iteração é,

substituindo (2.150) e (2.151) em (2.142):

$$\Delta V_k = -\frac{(\nabla^T \phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k)))^2}{\|\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla(\mathbf{x}_k))\|^2} \leq 0$$

garantindo o decrescimento da função, ao menos localmente.

Aqui também, existe a possibilidade do algoritmo se estacionar naqueles pontos que zeram o cálculo do comprimento do passo α_k, pontos onde $\nabla^T\phi(\mathbf{x})\nabla^2\phi(\mathbf{x})\text{sgn}(\nabla\phi(\mathbf{x})) = 0$, mesmo não se encontrando no mínimo global da função $\mathbf{x}^*$. O cálculo do comprimento do passo será indeterminado naqueles pontos onde $\text{sgn}(\nabla\phi(\mathbf{x})) \in \mathcal{N}(\nabla^2\phi(\mathbf{x}))$.

Cada ponto na sequência, portanto, deverá ser calculado como

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \text{ sgn}(\nabla\phi(\mathbf{x}_k)) \tag{2.152}$$

Na seguinte tabela são apresentadas em forma resumida os algoritmos discretos derivados nesta seção.

$\mathbf{u}_k$	α_k	ΔV	nome	núm. eq.
$-\nabla\phi(\mathbf{x}_k)$	$\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)}{\|\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	$-\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)P\nabla\phi(\mathbf{x}_k))^2}{\|\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	SD	(2.146)
$-\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)$	$\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	$-\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))^2}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	LVR	(2.149)
$-\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)$	$\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	$-\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))^2}{\|\nabla^2\phi(\mathbf{x}_k)\nabla^{-2}\psi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)\|^2}$	LV2	(2.149)
$-\text{sgn}(\nabla\phi(\mathbf{x}_k))$	$\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k))}{\|\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k))\|^2}$	$-\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla\phi(\mathbf{x}_k)))^2}{\|\nabla^2\phi(\mathbf{x}_k)\text{sgn}(\nabla(\mathbf{x}_k))\|^2}$	VSD	(2.152)

Tabela 2.6 Algoritmos discretos para minimização de funções escalares

2.8.2.1 Simulações dos algoritmos discretos

A seguir serão apresentadas simulações para os algoritmos discretos SD (2.146), LVR, LV2 (2.149) e VSD (2.152) com os comprimentos dos passos (2.145), (2.148) e (2.151), respectivamente, utilizando a função de Rosenbrock (2.54) com as constantes $a = 0.5$ e $b = 1$, e a função de Attouch (2.141).

As constantes do algoritmo LVR foram escolhidas $\mu = 1$ e ν variável com valores 1 e 0.1.

O critério de parada foi a iteração tal que a função $\|\nabla\phi(\mathbf{x}_k)\| < 0.01$.

Primeiro, serão testados os algoritmos com a função de Rosenbrock, a partir dos pontos iniciais $\mathbf{x}_0 = [-0.5\ \ 2]^T$, $\mathbf{x}_0 = [-1\ \ -0.5]^T$, $\mathbf{x}_0 = [1\ \ 3]^T$, $\mathbf{x}_0 = [1.5\ \ -0.5]^T$, $\mathbf{x}_0 = [1.5\ 0.5]^T$ e $\mathbf{x}_0 = [0.2; 2.5]^T$.

Os algoritmos de Lotka-Volterra regularizado (LVR), e Lotka-Volterra quadrado (LV2), foram testados apenas a partir dos pontos iniciais localizados no ortante positivo, ortante onde se encontra o mínimo, como exigido por hipótese.

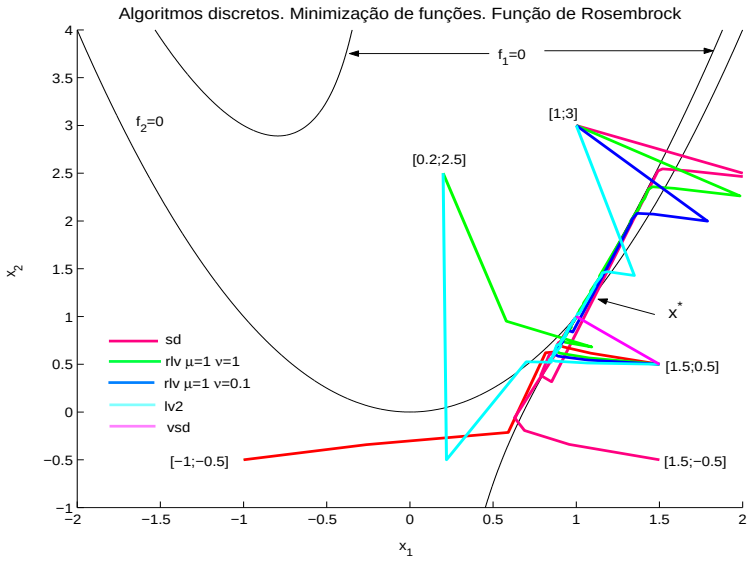

Figura 2.36: Algoritmos para minimização de funções discretas implementados com a função de Rosenbrock mostrando a convergência das trajetórias

A partir dos pontos iniciais $\mathbf{x}_0 = [-0.5\ \ 2]^T$ e $\mathbf{x}_0 = [0.2\ \ 2.5]^T$ o algoritmo *steepest descent* não convergiu, pois em poucas iterações a trajetória se estacionou no ponto

$\mathbf{x} = [-0.9776\ 2.6979]^T$, ponto que, como foi mencionado, zera o cálculo do comprimento do passo α_k. O algoritmo Lotka-Volterra regularizado também não convergiu a partir do ponto inicial $\mathbf{x}_0 = [0.2\ 2.5]^T$ para as constante $\mu = 1$ e $\nu = 0.1$. A partir desse ponto inicial, só convergiram os algoritmos de Lotka-Volterra regularizado com as constantes $\mu = 1$ e $\nu = 1$ e Lotka-Volterra quadrado. O algoritmo *steepest descent* a estrutura variável só convergiu a partir do ponto inicial $\mathbf{x}_0 = [1.5\ 0.5]^T$.

Os algoritmos com a função de Attouch foram testados a partir dos pontos iniciais $\mathbf{x}_0 = [-1\ 0.5]^T$, $\mathbf{x}_0 = [1.5\ 0.5]^T$, $\mathbf{x}_0 = [-1\ -2]^T$ e $\mathbf{x}_0 = [1\ -2]^T$, sendo que o algoritmo LVR só foi testado a partir do ponto inicial localizado no ortante positivo.

As constantes deste algoritmo foram mantidas $\mu = 1$ e ν variável, primeiro com o valor 1 e posteriormente com 0.1.

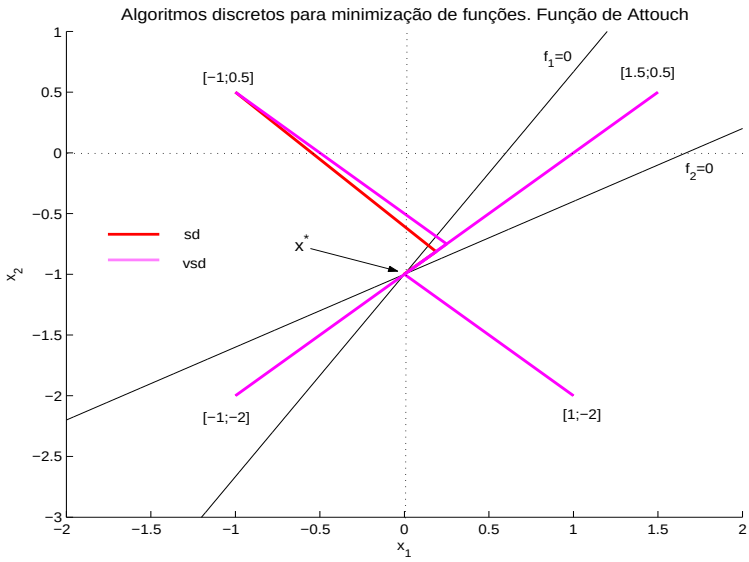

Figura 2.37: Algoritmos para minimização de funções discretas implementados com a função de Attouch mostrando a convergência das trajetórias

As trajetórias geradas pelo algoritmo LV2 sempre se estacionaram em um ponto tal que $\nabla\phi(\mathbf{x})^T\nabla^2\phi(\mathbf{x})\nabla^{-2}\psi(\mathbf{x})\nabla\phi(\mathbf{x}) = 0$, ponto que zera o comprimento do passo α_k, e portanto onde se estaciona a trajetória mesmo que $\nabla\phi(\mathbf{x}) \neq \mathbf{0}$. Alguns destes pontos são $\mathbf{x} = [-0.1979\ \ -1.1723]^T$, $\mathbf{x} = [-0.4191\ \ -1.3439]^T$, $\mathbf{x} = [-0.1887\ \ -0.8378]^T$, $\mathbf{x} = [0.8583\ \ -0.0269]^T$, como já foi mencionado. O algoritmo LVR gerou trajetórias que também se estacionaram nestes pontos, não atingindo o mínimo no ortante positivo localizado em $\mathbf{x} = [3/5\ 0]^T$. Os algoritmos SD e VSD atingiram o mínimo global desde

todos os pontos iniciais testados.

A seguinte tabela mostra a quantidade de iterações de cada algoritmo a partir dos diversos pontos iniciais testados. Os tempos de demora das trajetórias também foram testados, embora esse dado não é colocado na tabela por ser inconclusivo: diferentes execuções demoraram diferentes durações. Em todo caso, esclarece-se que todos os tempos foram menores a 1 segundo.

	$\mathbf{x}_0$	SD	Lotka-Volterra reg.		LV2	VSD
			$\mu = 1 \;\; \nu = 1$	$\mu = 1 \;\; \nu = 0.1$		
Rosenbrock	$[1; 3]^T$	25	19	14	9	-
	$[1.5; 0.5]^T$	6	7	25	18	3
	$[0.2; 2.5]^T$	-	6	-	13	-
	$[-1; -0.5]^T$	24	N.T.	N.T.	N.T.	-
	$[1.5; -0.5]^T$	14	N.T.	N.T.	N.T.	-
	$[-0.5; 2]^T$	-	N.T.	N.T.	N.T.	-
Attouch	$[-1\ 0.5]^T$	3	N.T.	N.T.	-	2
	$[1.5\ 0.5]^T$	1	-	-	-	1
	$[-1\ -2]^T$	1	N.T.	N.T.	-	1
	$[1\ 2]^T$	1	N.T.	N.T.	-	1

Tabela 2.7 Resultados das execuções dos algoritmos discretos para minimização de funções utilizando a função de Rosenbrock e de Attouch, onde - significa que o algoritmo não convergiu ao mínimo e N.T. é não testado

2.9 Conclusões deste capítulo

A seguir, serão apresentadas resumidamente algumas conclusões dos estudos comparativos realizados com os algoritmos de primeira ordem destinados a achar zeros de funções.

1) A convergência exponencial do algoritmo de Newton contínuo nem sempre implica em um tempo menor de convergência, pois os algoritmos contínuos a estrutura variável apresentam convergência em tempo finito.

2) Quanto maior o número de singularidades estranhas, maior a possibilidade do algoritmo de Newton discreto não convergir. Na função de Branin, por exemplo, na região próxima dos zeros esta quantidade é proporcional à constante c; observa-se nesse caso o aumento das regiões de não convergência com o aumento desta constante. Eis outra justificativa para a elaboração e utilização de outros algoritmos que não sejam afetados pela presença destas singularidades.

3) A visualização das bacias de atração de cada zero para as funções testadas e para cada algoritmo, mostrou que algumas apresentam formas geométricas mais regulares. Esta pode ser uma característica desejável pois implica um comportamento mais regular do algoritmo.

4) A utilização do time de algoritmos, embora implica um tempo de convergência obviamente maior do que a utilização de um algoritmo único, aumenta a possibilidade de convergência, podendo diminuir o número total de iterações, pois inicializa o algoritmo com melhor desempenho em um ponto de menor norma da função objetivo.

5) A escolha ótima do comprimento do passo calculado por LOC para os algoritmos discretos, garante a maior diminuição da norma do erro a cada iteração. Porém, existe a possibilidade de este cálculo ser indeterminado, o que provocaria a não convergência do algoritmo. Foram localizados estes pontos de indeterminação para cada algoritmo para as funções de Branin, de Rosenbrock e de Camelback.

A seguir, serão apresentadas resumidamente algumas conclusões sobre os algoritmos para minimização de funções escalares convexas.

1) Quando o ortante onde se localiza o mínimo global da função é desconhecido, evidentemente os algoritmos SD e VSD contínuos são os únicos dos quatro capazes de garantir a convergência. O algoritmo LV2 só seria convergente se fosse testado a partir de pontos iniciais em cada ortante.

2) Outros algoritmos, inclusive globalmente convergentes poderiam ter sido derivados escolhendo como controlador $\Psi(\mathbf{x})$ uma matriz positiva definida, como foi mencionado.

3) Dentre os algoritmos discretos, o SD é o que apresentou um menor número de iterações. O VSD também se mostrou eficiente a partir dos pontos iniciais testados para ambas funções de teste.

Capítulo 3

Algoritmos de segunda ordem destinados a achar zeros de funções vetoriais ou mínimos de funções escalares como sistemas dinâmicos de controle

3.1 Introdução

No capítulo 2 foram abordados algoritmos de primeira ordem, isto é, algoritmos cujas leis de atualização das variáveis, no caso contínuo, podiam ser representadas por equações diferenciais de primeira ordem.

Foram estudados algoritmos para achar zeros de funções vetoriais, apresentados por Bhaya e Kaszkurewicz em [13], [11] e [10], estudando-os como sistemas dinâmicos de controle em malha fechada e derivados através da metodologia de funções de Liapunov de controle e controle ótimo de Liapunov. Tais algorimos foram discretizados por Euler e seu comprimento de passo ótimo calculado pela metodologia de controle ótimo de Liapunov.

Também, pelas mesmas metodologias, foram projetados algoritmos destinados a achar mínimos de funções escalares convexas.

Neste capítulo, a proposta é atingir iguais objetivos utilizando algoritmos de segunda ordem, interpretando estes como sistemas dinâmicos de controle. Na sua versão contínua, estes algoritmos poderão ser representados por equações diferenciais de segunda ordem.

A figura 3.1 mostra um exemplo de diagrama de blocos do sistema em malha fechada, ou realimentado, que representa a dinâmica das variáveis que interatuam neste sistema de controle.

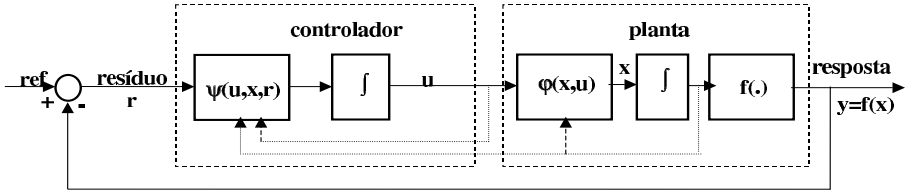

Figura 3.1: Diagrama de blocos representativo do sistema de segunda ordem contínuo

3.2 Algoritmos de segunda ordem destinados a achar zeros de funções vetoriais

Seja uma função $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ diferenciável em todo seu domínio, o objetivo será projetar algoritmos de segunda ordem para achar algum zero desta função, ou valor $\mathbf{x}^*$ tal que $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$.

A referência do sistema será, portanto, o valor zero, ao qual se deseja que tenda a função. Procura-se que as trajetórias das variáveis de estado $\mathbf{x}(t)$ tendam a uma raíz da função $\mathbf{x}^*$. A variável de resíduo ou erro será definida como o valor de referência menos o valor da função, isto é

$$\mathbf{r} = -\mathbf{f}(\mathbf{x})$$

assumindo também aqui que o mapeamento que relaciona ambas variáveis existe.

Derivando com respeito ao tempo:

$$\dot{\mathbf{r}} = -D_{\mathbf{f}}(\mathbf{x})\dot{\mathbf{x}}$$

onde $D_{\mathbf{f}}(\mathbf{x})$ representa a mariz jacobiana tradicional.

A diferença dos algoritmos de primeira ordem, aqui será escolhida como função de Liapunov candidata uma função escalar dependente de todos os estados do sistema, que neste caso serão a variável de controle $\mathbf{u}$ e a variável de estado $\mathbf{x}$.

$$V(\mathbf{x}, \mathbf{u}) = \frac{\mathbf{r}^T \mathbf{r}}{2} + \frac{\mathbf{u}^T \mathbf{u}}{2} \tag{3.1}$$

a qual é positiva definida e só assume o valor zero para o sinal de controle $\mathbf{u} = \mathbf{0}$ e em uma raíz da função.

Derivando (3.1) com respeito ao tempo:

$$\dot{V}(\mathbf{x}, \mathbf{u}) = \mathbf{r}^T \dot{\mathbf{r}} + \mathbf{u}^T \dot{\mathbf{u}} = -\mathbf{r}^T D_{\mathbf{f}}(\mathbf{x}) \dot{\mathbf{x}} + \mathbf{u}^T \dot{\mathbf{u}} \tag{3.2}$$

Este é o ponto de partida para o projeto de diferentes algoritmos pela metodologia de funções de Liapunov de controle. A mesma consiste em escolher planta e controlador (isto é, $\dot{\mathbf{x}}$ e $\dot{\mathbf{u}}$) de maneira tal de fazer (3.2) negativa semidefinida, garantindo assim a estabilidade do sistema.

1) Primeira escolha

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Substituindo em (3.2): $\dot{V} = -\mathbf{r}^T D_{\mathbf{f}} \mathbf{u} + \mathbf{u}^T \dot{\mathbf{u}} = \mathbf{u}^T (-D_{\mathbf{f}}^T \mathbf{r} + \dot{\mathbf{u}})$
onde não foi explicitada a dependência da variável de estado $\mathbf{x}$.

Controlador $\quad \dot{\mathbf{u}} = -\Gamma \mathbf{u} + D_{\mathbf{f}}^T \mathbf{r} \quad$ onde $\Gamma = \Gamma^T \succ 0$
o que implica que a derivada temporal da função de Liapunov candidata será

$$\dot{V} = -\mathbf{u}^T \Gamma \mathbf{u} \leq 0$$

Pelo lema de Barbalat, $\dot{V}$ tende a zero, razão pela qual $\mathbf{u}$ e $\dot{\mathbf{u}}$ tendem para zero também. A partir da escolha do controlador, $D_{\mathbf{f}}^T(\mathbf{x})\mathbf{r}$ tende para zero também. Observe-se que isto não implica que o resíduo tenderá para zero, e portanto a trajetória para um valor $\mathbf{x}^*$, raíz de $\mathbf{f}(\mathbf{x})$, pois a trajetória poderia se estacionar em um ponto tal que $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}))$. Eis a razão pela qual a convergência assintótica das trajetórias a

uma raíz da função é apenas local.

Exemplo 3.2.1 *Seja* $\mathbf{f}(\mathbf{x}) = [\mathbf{x}_1 \ \sin(\mathbf{x}_2)]^T$, *esta função apresenta zeros em* $\mathbf{x}^* = [0 \ (k\pi)]^T$ *para todo* $k \in \mathbb{N}$.

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 \\ 0 & \cos(\mathbf{x}_2) \end{bmatrix} = D_{\mathbf{f}}^T(\mathbf{x}) \ \Rightarrow \ D_{\mathbf{f}}^T(\mathbf{x})\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_1 \\ \cos(\mathbf{x}_2)\sin(\mathbf{x}_2) \end{bmatrix}$$

E portanto a trajetória $\mathbf{x}(t)$ *poderia se estacionar em um ponto* $\mathbf{x} = \left[0 \ (2k+1)\frac{\pi}{2}\right]^T$ *para todo* $k \in \mathbb{N}$, *sendo* $\mathbf{f}(\mathbf{x}) = [0 \ \pm 1]^T \neq \mathbf{0}$

Este algoritmo pode ser escrito como uma ODE de segunda ordem da forma

$$\ddot{\mathbf{x}} + \Gamma\dot{\mathbf{x}} + D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{3.3}$$

O algoritmo será chamado simplesmente de DC1 (dinâmica no controlador 1).

2) Segunda escolha

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Substituindo em (3.2) novamente: $\dot{V} = \mathbf{u}^T(-D_{\mathbf{f}}^T\mathbf{r} + \dot{\mathbf{u}})$.

Controlador $\quad \dot{\mathbf{u}} = -D_{\mathbf{f}}^T D_{\mathbf{f}}\mathbf{u} + D_{\mathbf{f}}^T\mathbf{r} = D_{\mathbf{f}}^T(-D_{\mathbf{f}}\mathbf{u} + \mathbf{r})$

e substituindo na equação da derivada da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T D_{\mathbf{f}} D_{\mathbf{f}}^T\mathbf{u} \leq 0$$

Analisando de maneira similar, vemos que este algoritmo também poderia se estacionar em um ponto tal que $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}))$, razão pela qual nos referimos à convergência local das trajetórias a uma raíz da função.

Este algoritmo pode ser escrito como uma ODE de segunda ordem da forma

$$\ddot{\mathbf{x}} + D_{\mathbf{f}}^T D_{\mathbf{f}}\dot{\mathbf{x}} + D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{3.4}$$

Este algoritmo será chamado de DC2.

3) Terceira escolha

Planta $\quad \dot{\mathbf{x}} = D_{\mathbf{f}}^{-1}\Gamma\mathbf{u}$ onde $\Gamma = \Gamma^T \succ 0$

Substituindo em (3.2): $\dot{V} = -\mathbf{r}^T\Gamma\mathbf{u} + \mathbf{u}^T\mathbf{u} = \mathbf{u}^T(\dot{\mathbf{u}} - \Gamma\mathbf{r})$.

Controlador $\quad \dot{\mathbf{u}} = \Gamma(\mathbf{r} - \mathbf{u})$

e subsituindo na equação da derivada temporal da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T\Gamma\mathbf{u} \leq 0$$

Por Barbalat, se $\dot{V}$ tende para zero, as variáveis $\mathbf{u}$ e $\dot{\mathbf{u}}$ também tendem para zero, e portanto, pela equação do controlador, $\mathbf{r}$ só pode tender para zero, sendo $\mathbf{x}^*$, raíz da função $\mathbf{f}(\mathbf{x})$, a única possibilidade de estacionamento da trajetória gerada pelo algoritmo. Em contrapartida, este algoritmo não está definido naqueles pontos onde a matriz jacobiana é singular, como acontece com o algoritmo de Newton. Isto é, as trajetórias não poderão atravesar o locus definido por $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$.

Este algoritmo não pode ser facilmente representado por uma ODE de segunda ordem. Fazendo uma mudança conveniente de variáveis, chega-se a uma relação próxima que pode ser escrita como a seguinte ODE:

$$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma^2\mathbf{y} = \mathbf{0} \tag{3.5}$$

onde $\Gamma\mathbf{y} = \mathbf{f}(\mathbf{x})$.

Este algoritmo será chamado de DC3.

4) Quarta escolha

Planta $\quad \dot{\mathbf{x}} = D_{\mathbf{f}}^T\Gamma\mathbf{u}$, onde $\Gamma = \Gamma^T \succ 0$

Substituindo em (3.2): $\dot{V} = -\mathbf{r}^T D_{\mathbf{f}} D_{\mathbf{f}}^T\Gamma\mathbf{u} + \mathbf{u}^T\dot{\mathbf{u}} = \mathbf{u}^T(\dot{\mathbf{u}} - \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T\mathbf{r})$

Controlador $\quad \dot{\mathbf{u}} = -\Gamma\mathbf{u} + \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T\mathbf{r}$

e substituindo na equação da derivada temporal da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T \Gamma \mathbf{u}$$

Aqui o algoritmo poderia se estacionar em um ponto onde $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_\mathbf{f}(\mathbf{x}))$ ou ainda em um ponto onde $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_\mathbf{f}(\mathbf{x}) D_\mathbf{f}^T(\mathbf{x}))$, razao pela qual também aqui a convergência assintótica é garantida apenas localmente.

Também neste caso resulta difícil chegar a uma representação do algoritmo como uma ODE de segunda ordem. Fazendo uma mudança conveniente de variáveis, pode-se chegar a seguinte representação:

$$\ddot{\mathbf{y}} + \Gamma \dot{\mathbf{y}} + \Gamma D_\mathbf{f} D_\mathbf{f}^T \mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{3.6}$$

onde $D_\mathbf{f}^T \Gamma \dot{\mathbf{y}} = \dot{\mathbf{x}}$.

Ese algoritmo será chamado de DC4.

5) Quinta escolha

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Substituindo em (3.2): $\dot{V} = -\mathbf{r}^T D_\mathbf{f} \mathbf{u} + \mathbf{u}^T \dot{\mathbf{u}} = \mathbf{u}^T (\dot{\mathbf{u}} - D_\mathbf{f}^T \mathbf{r})$

Controlador $\quad \dot{\mathbf{u}} = D_\mathbf{f}^T \mathbf{r} - (\mathbf{u}^T \Gamma \mathbf{u}) \mathrm{sgn}(\mathbf{u})$ onde $\Gamma = \Gamma^T \succ 0$

e substituindo na equação que representa a derivada temporal da função de Liapunov candidata

$$\dot{V} = -(\mathbf{u}^T \Gamma \mathbf{u}) \mathbf{u}^T \mathrm{sgn}(\mathbf{u}) \leq 0$$

Aqui também o algoritmo poderia se estacionar em um ponto tal que $\mathbf{f}(\mathbf{x}) \in \mathcal{N}(D_\mathbf{f}^T(\mathbf{x}))$, razão pela qual a convergência assintótica das trajetórias é apenas local.

Este algoritmo pode ser representado como uma ODE de segunda ordem da forma:

$$\ddot{\mathbf{x}} + \dot{\mathbf{x}}^T \Gamma \dot{\mathbf{x}} \, \mathrm{sgn}(\dot{\mathbf{x}}) + D_\mathbf{f}^T \mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{3.7}$$

Ese algoritmo será chamado de DC5.

Resumindo, a metodologia CLF/LOC se mostrou capaz de derivar 5 algoritmos de segunda ordem destinados a achar zeros de funções vetoriais. Certamente, outras escolhas de planta e controlador poderiam ter sido utilizadas para fazer a derivada da função de Liapunov candidata negativa semidefinida. Também, uma outra função de Liapunov candidata positiva definida e dependente da variável de estado e da variável de controle poderia ter sido escolhida. A partir da derivada temporal desta, com certeza outras escolhas de planta e controlador fariam esta derivada temporal negativa semidefinida, originando assim novos algoritmos de segunda ordem.

Os cinco algoritmos apresentados são resumidos na seguinte tabela.

nome	Planta	Controlador	ODE
DC1	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\Gamma\mathbf{u} + D_{\mathbf{f}}^T\mathbf{r}$	$\ddot{\mathbf{x}} + \Gamma\dot{\mathbf{x}} + D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0}$
DC2	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = D_{\mathbf{f}}^T(\mathbf{r} - D_{\mathbf{f}}\mathbf{u})$	$\ddot{\mathbf{x}} + D_{\mathbf{f}}^T D_{\mathbf{f}}\dot{\mathbf{x}} + D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0}$
DC3	$\dot{\mathbf{x}} = D_{\mathbf{f}}^{-1}\Gamma\mathbf{u}$	$\dot{\mathbf{u}} = \Gamma(\mathbf{r} - \mathbf{u})$	$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma^2\mathbf{y} = \mathbf{0}$ $\Gamma\mathbf{y} = \mathbf{f}(\mathbf{x})$
DC4	$\dot{\mathbf{x}} = D_{\mathbf{f}}^T\Gamma\mathbf{u}$	$\dot{\mathbf{u}} = \Gamma(D_{\mathbf{f}}D_{\mathbf{f}}^T\mathbf{r} - \mathbf{u})$	$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma D_{\mathbf{f}}D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0}$ $D_{\mathbf{f}}^T\Gamma\dot{\mathbf{y}} = \dot{\mathbf{x}}$
DC5	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = D_{\mathbf{f}}^T\mathbf{r} - (\mathbf{u}^T\Gamma\mathbf{u})\mathrm{sgn}(\mathbf{u})$	$\ddot{\mathbf{x}} + \dot{\mathbf{x}}^T\Gamma\dot{\mathbf{x}}\mathrm{sgn}(\dot{\mathbf{x}}) + D_{\mathbf{f}}^T\mathbf{f}(\mathbf{x}) = \mathbf{0}$

Tabela 3.1 Algoritmos contínuos de segunda ordem deduzidos com a metodologia CLF/LOC

3.2.1 Simulações dos algoritmos de segunda ordem contínuos

A seguir, serão apresentadas, a modo de ilustração, simulações realizadas com diversas funções de teste. As funções foram escolhidas de duas variáveis a fim de poderem ser graficadas. As simulações foram realizadas no aplicativo Simulink do Matlab 6, discretizando os algoritmos pelo método de Euler; o comprimento do passo é de 0.01s. constante e foi simulado um tempo de 10 segundos.

O valor inicial da variável de controle foi escolhido zero.

3.2.1.1 Simulações dos algoritmos contínuos com a função de Camelback

A primeira função escolhida como teste será a função de Camelback, apresentada em [18, p. 521], descrita pela primitiva:

$$\phi(x_1, x_2) = ax_1^2 + bx_1^4 + cx_1^6 - x_1x_2 + dx_2^2 + ex_2^4 \tag{3.8}$$

Foram escolhidas como constantes $a = -2,\ b = 1.05,\ c = -\frac{1}{6},\ d = -1$ e $e = 0$. Com estas constantes a função apresenta um máximo global, dois máximos locais, e duas selas. Estes pontos correspondem a zeros do gradiente da função primitiva, cuja expressão e valor para as constantes escolhidas estão dadas por:

$$\mathbf{f}(\mathbf{x}) = \nabla\phi(\mathbf{x}) = \begin{bmatrix} 2ax_1 + 4bx_1^3 + 6cx_1^5 - x_2 \\ -x_1 + 2dx_2 + 4ex_2^3 \end{bmatrix} = \begin{bmatrix} -4x_1 + 4.2x_1^3 - x_1^5 - x_2 \\ -x_1 - 2x_2 \end{bmatrix} \tag{3.9}$$

Os zeros desta função correspondem aos pontos $\mathbf{x} = [-1.7475\ 0.8737]^T$,
$\mathbf{x} = [-1.0705\ 0.5352]^T$, $\mathbf{x} = [0\ 0]^T$, $\mathbf{x} = [1.0705\ -0.5352]^T$,
$\mathbf{x} = [1.7475\ -0.8737]^T$

O hessiano da função primitiva, e seu cálculo para as constantes escolhidas, são dados por:

$$D_{\mathbf{f}}(\mathbf{x}) = \nabla^2\phi(\mathbf{x}) = \begin{bmatrix} 2a + 12bx_1^2 + 30cx_1^4 & -1 \\ -1 & 2d + 12ex_2^2 \end{bmatrix} = \begin{bmatrix} -4 + 12.6x_1^2 - 5x_1^4 & -1 \\ -1 & -2 \end{bmatrix} \tag{3.10}$$

cujo determinante é dado pela função $\det(D_{\mathbf{f}}(\mathbf{x})) = 7 - 25.2x_1^2 + 10x_1^4$, o qual possui zeros ao longo das retas verticais definidas por $x_1 = 1.7737,\ x_1 = 0.6739,\ x_1 =$

-0.6739, $x_1 = -1.7737$. Observe-se que, para as constantes escolhidas, $D_{\mathbf{f}}(\mathbf{x}) = D_{\mathbf{f}}(x_1)$.

Os ganhos dos algoritmos, excetuando DC2 cujos ganhos dependem unicamente da matriz jacobiana, foram testados com diversos valores e escolhidos os que apresentaram melhores resultados. Estes foram: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = 0.1I$, e DC5: $\Gamma = 5I$. Os pontos iniciais a partir dos quais foram testados os cinco algoritmos contínuos são $\mathbf{x}_0 = [-1 \ -1.5]^T$, $\mathbf{x}_0 = [2 \ 1]^T$.

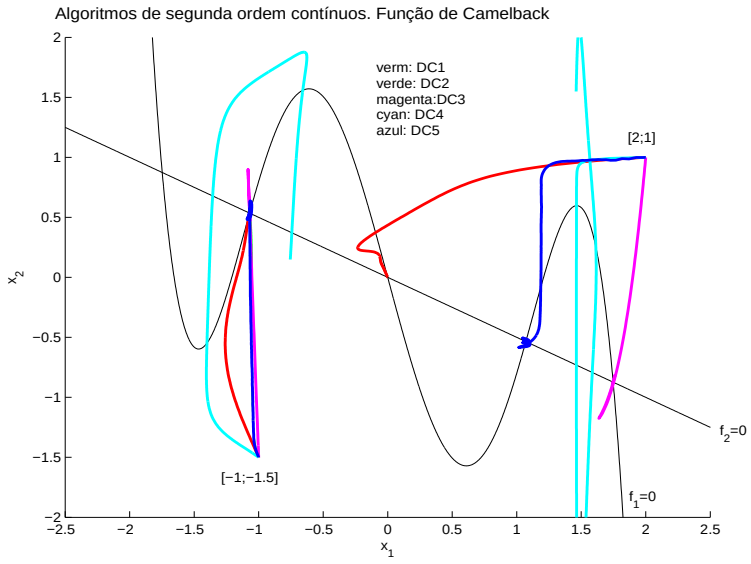

Figura 3.2: Simulações dos algoritmos contínuos para a função de Camelback mostrando a convergência das trajetórias

As curvas não graficadas correspondem a trajetórias geradas por algoritmos que não convergiram a nenhum zero. A partir do ponto $\mathbf{x}_0 = [-1 \ -1.5]^T$, todas as trajetórias conseguiram convergir a um zero da função, excetuando a gerada pelo algoritmo DC4, que apresenta uma trajetória errante, sendo impossível afirmar que convergirá a um zero. A partir do ponto inicial $\mathbf{x}_0 = [2 \ 1]^T$, apenas as trajetórias geradas pelos algoritmos DC1, DC3 e DC5 conseguiram convergir a zeros, mesmo diferentes.

3.2.1.2 Simulações dos algoritmos contínuos com a função de Rosenbrock

A função de Rosenbrock ([18, p. 512]) está definida pela primitiva:

$$\phi(x_1, x_2) = a(x_1^2 - x_2)^2 + (x_1 - b)^2 \tag{3.11}$$

As constantes escolhidas são $a = 0.5$ e $b = 1$. Com estas constantes, a função apresenta um mínimo global em $x_1 = x_2 = 1$, que corresponde a um único zero do gradiente. A expressão deste e seu valor para as constantes escolhidas estão dadas por:

$$\mathbf{f}(\mathbf{x}) = \nabla\phi(\mathbf{x}) = \begin{bmatrix} 4ax_1(x_1^2 - x_2) + 2(x_1 - b) \\ -2a(x_1^2 - x_2) \end{bmatrix} = \begin{bmatrix} 2x_1(x_1^2 - x_2) + 2(x_1 - 1) \\ -(x_1^2 - x_2) \end{bmatrix} \tag{3.12}$$

O hessiano da função primitiva, e seu cálculo para as constantes escolhidas, são dados por:

$$D_{\mathbf{f}}(\mathbf{x}) = \nabla^2\phi(\mathbf{x}) = \begin{bmatrix} 12ax_1^2 - 4ax_2 + 2 & -4ax_1 \\ -4ax_1 & 2a \end{bmatrix} = \begin{bmatrix} 6x_1^2 - 2x_2 + 2 & -2x_1 \\ -2x_1 & 1 \end{bmatrix} \tag{3.13}$$

cujo determinante é $\det(D_{\mathbf{f}}(\mathbf{x})) = 8a^2x_1^2 - 8a^2x_2 + 4a = 2x_1^2 - 2x_2 + 2$, o qual possui raízes determinadas pelo locus $x_2 = x_1^2 + 1$.

Os ganhos foram escolhidos: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = 0.1I$, e DC5: $\Gamma = 5I$. Os pontos iniciais a partir dos quais foram testados os cinco algoritmos contínuos foram $\mathbf{x}_0 = [0.5\ 2]^T$, $\mathbf{x}_0 = [1.5\ -0.5]^T$.

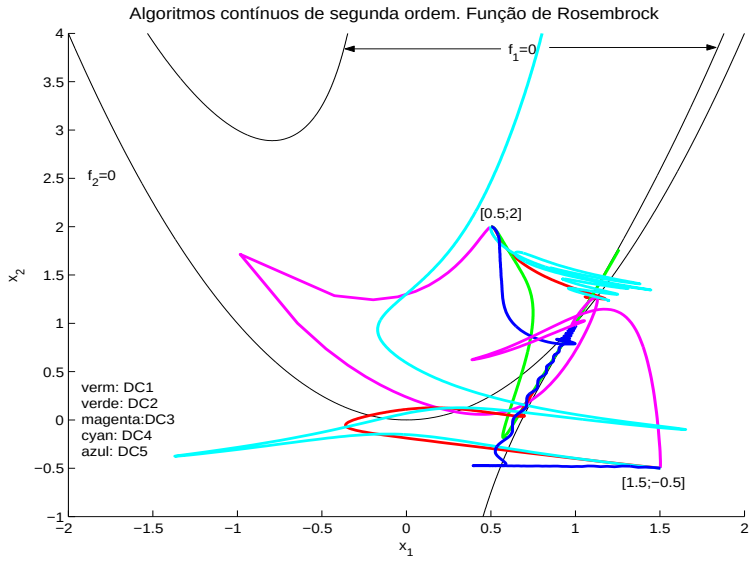

Figura 3.3: Simulações dos algoritmos contínuos para a função de Rosenbrock

A partir do ponto inicial $\mathbf{x}_0 = [0.5\ 2]^T$ apenas a trajetória gerada pelo algoritmo DC4 não convergiu ao zero da função, embora pareça tender a este. Observe-se que a trajetória gerada pelo algoritmo DC3, que não está definido sobre o locus dado pela

equação $\det(D_{\mathbf{f}}(\mathbf{x})) = x_1^2 - x_2 + 1 = 0$, parece acompanhar esta curva, e só consegue atravessá-la devido à discretização do algoritmo, tal como acontecia com o algoritmo de Newton. A partir do ponto inicial $\mathbf{x}_0 = [1.5 \ -0.5]^T$, apenas as trajetórias geradas pelos algoritmos DC1, DC3 e DC5 conseguiram convergir ao zero da função.

3.2.1.3 Simulações dos algoritmos contínuos com a função de Branin

A função apresentada por Branin em [18, p. 510], está definida como uma função vetorial de duas variáveis, e portanto não corresponde com o gradiente de uma função primitiva escalar. A função define-se:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} a - bx_2 + c\sin(dx_2) - x_1 \\ x_2 - e\sin(fx_1) \end{bmatrix} \tag{3.14}$$

As constantes utilizadas são $a = 1,\ b = 2,\ d = 4\pi,\ e = 0.5,\ f = 2\pi$. Inicialmente, a constante c é escolhida com valor 0, o qual faz de f_1 uma função linear. Com estas constantes a função apresenta 5 zeros. A derivada da função e seu valor para as constantes escolhidas são:

$$D_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} -1 & -b + cd\cos(dx_2) \\ -ef\cos(fx_1) & 1 \end{bmatrix} = \begin{bmatrix} -1 & -2 \\ -\pi\cos(2\pi x_1) & 1 \end{bmatrix} \tag{3.15}$$

Observe-se que, para as constantes escolhidas, $D_{\mathbf{f}}(\mathbf{x}) = D_{\mathbf{f}}(x_1)$.

Analisando a função de Branin, $\det(D_{\mathbf{f}}(\mathbf{x})) = -1 - 2\pi\cos(2\pi x_1)$, o qual é igual a zero ao longo das retas verticais definidas por $x_1 = \pm 0.2754 \pm k$, para todo $k \in \mathbb{N}$.

Os ganhos foram escolhidos: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = 0.5I$, e DC5: $\Gamma = 5I$. Os algoritmos serão testados a partir dos pontos iniciais $\mathbf{x}_0 = [0.25 \ -0.4]^T$, $\mathbf{x}_0 = [1.25 \ -0.6]^T$, $\mathbf{x}_0 = [0.75 \ 0.6]^T$.

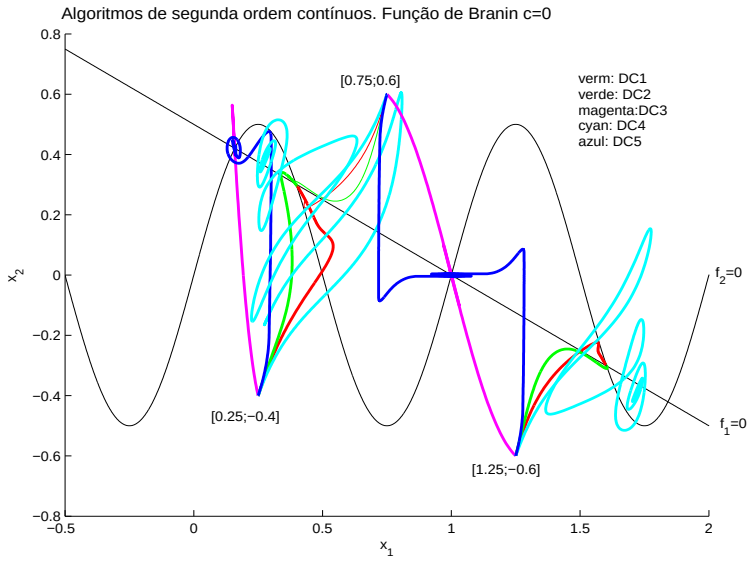

Figura 3.4: Simulações dos algoritmos contínuos para a função de Branin com $c = 0$

As trajetórias geradas pelos algoritmos, em geral, conseguiram convergir a zeros da função, mesmo que diferentes para igual ponto inicial. A exceção está dada pela trajetória gerada pelo algoritmo DC4, que também aqui apresenta um comportamento errante pelo plano de fase e resulta impossível afirmar se as trajetórias convergirão para zeros da função.

A seguir, será aumentado o valor da constante c para 0.13, e os algoritmos serão testados a partir dos mesmos pontos iniciais e com os mesmos ganhos. Com estas constantes, a função de Branin continua apresentando 5 zeros. O lugar das raízes da equação $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$ continuam sendo as retas verticais localizadas em $x_1 = \pm 0.2754 \pm k$, para todo $k \in N$, só que estas retas verticais agora se vêem moduladas horizontalmente. A amplitude desta modulação parece proporcional ao valor da constante c.

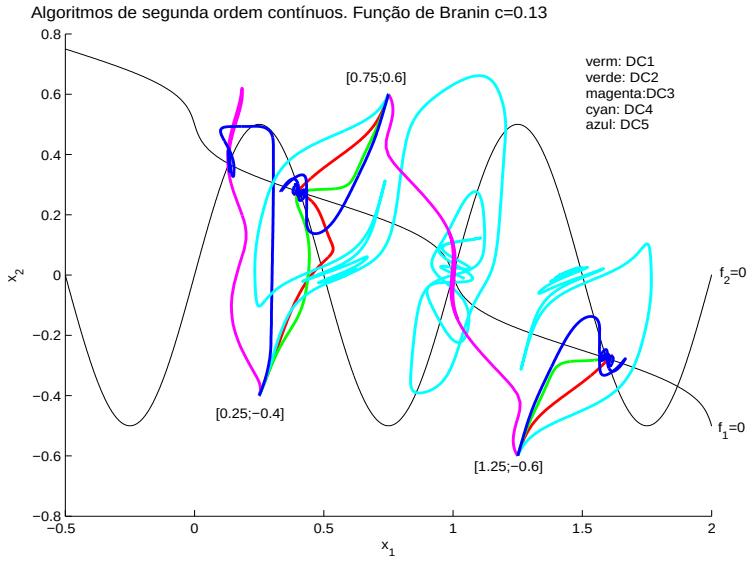

Figura 3.5: Simulações dos algoritmos contínuos para a função de Branin com $c = 0.13$

Observa-se um comporamento similar das trajetórias. Também aqui a gerada pelo algoritmo DC4 apresenta amplas oscilações, e as que partem dos pontos iniciais $\mathbf{x}_0 = [1.25 \quad -0.6]^T$ e $\mathbf{x}_0 = [0.75 \;\; 0.6]^T$ também resultam impossíveis de afirmar se tenderiam a zeros caso as trajetórias tivessem continuado.

Em geral, observa-se nestas simulações que as trajetórias geradas pelos algoritmos contínuos de segunda ordem DC4 e DC5 apresentam excursões e oscilações maiores do que aquelas apresentadas pelos algoritmos de primeira ordem. As geradas pelos algoritmos DC1, DC2 e DC3 sempre parecem convergir de maneira direta para um zero da função. Outra observação interessante é a falta do chattering característico das trajetórias geradas pelo algoritmo DC5, que é a estrutura variável. A razão desta ausência deve-se a que apenas o controlador, e não a planta, possui uma dinâmica a estrutura variável, e além disso amortecida pelo termo $\mathbf{u}^T\Gamma\mathbf{u}$, o que também diminui a amplitude do chattering. Também foi notado que as trajetórias são muito sensíveis à escolha do ganho Γ, exceção feita para o algoritmo DC2, que não depende deste ganho. Eis a razão pela qual diversos valores de ganhos devieram ser testados com cada algoritmo e para cada função, e foram escolhidos os que pareciam apresentar um melhor comportamento (menores oscilações, excursões pelo plano de fase de menor amplitude, e convergência das trajetórias).

3.2.2 Discretização dos algoritmos

Nesta seção, os algoritmos de segunda ordem serão discretizados pelo método de Euler, e seus comprimentos dos passos de iteração calculados de maneira ótima pela metodologia de controle ótimo de Liapunov.

Por serem algoritmos de segunda ordem, existe uma dinâmica tanto na planta quanto no controlador, e as duas variáveis de estado $\mathbf{x}$ e $\mathbf{u}$ devem convergir aos valores desejados. Esta é a razão pela qual o sistema comporta-se como um sistema bilinear, e duas funções de Liapunov discretas serão escolhidas a fim de que os comprimentos dos passos para as atualizações de ambas variáveis apresentem o maior decrescimento destas funções de Liapunov candidatas discretas, segundo a metodologia de controle ótimo de Liapunov.

Genericamente

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \varphi_k \tag{3.16}$$

onde $\varphi_k = \dot{\mathbf{x}}$ e dependerá da escolha da planta.

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \beta_k \psi_k \tag{3.17}$$

onde $\psi_k = \dot{\mathbf{u}}$ e dependerá da escolha do controlador.

O resíduo discreto também aqui será escolhido como o valor de referência, $\mathbf{0}$, menos o valor da função.

$$\mathbf{r}_k = -\mathbf{f}(\mathbf{x}_k)$$

e aproximando por Taylor até o termo de primeira ordem

$$\mathbf{r}_{k+1} \simeq \mathbf{r}_k - \alpha_k D_{\mathbf{f}} \varphi_k$$

onde resumimos $D_{\mathbf{f}} = D_{\mathbf{f}}(\mathbf{x}_k)$.

Escolhendo como função de Liapunov discreta candidata para a variável de estado

$$V_{\mathbf{r}_k} = \mathbf{r}_k^T \mathbf{r}_k$$

$$\Rightarrow \quad \Delta V_{\mathbf{r}} = V_{\mathbf{r}_{k+1}} - V_{\mathbf{r}_k} = \|\mathbf{r}_{k+1}\|^2 - \|\mathbf{r}_k\|^2 = -2\alpha_k \mathbf{r}_k^T D_{\mathbf{f}} \varphi_k + \alpha_k^2 \varphi_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \varphi_k$$

e derivando com respeito a α_k e igualando a zero, segundo a metodologia LOC:

$$\frac{\partial \Delta V_{\mathbf{r}}}{\partial \alpha_k} = -2\mathbf{r}_k^T D_{\mathbf{f}} \varphi_k + 2\alpha_k \varphi_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \varphi_k = 0$$

Portanto

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} \varphi_k}{\varphi_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \varphi_k} \tag{3.18}$$

Com este comprimento do passo

$$\Delta V_{\mathbf{r}} = -\frac{(\mathbf{r}_k^T D_{\mathbf{f}} \varphi_k)^2}{\varphi_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \varphi_k} \leq 0$$

o que garante o não crescimento desta função de Liapunov ao longo das trajetórias do sistema em malha fechada.

A função de Liapunov discreta candidata para a variável de controle será escolhida

$$V_{\mathbf{u}_k} = \mathbf{u}_k^T \mathbf{u}_k$$

$$\Rightarrow \quad \Delta V_{\mathbf{u}} = V_{\mathbf{u}_{k+1}} - V_{\mathbf{u}_k} = \|\mathbf{u}_{k+1}\|^2 - \|\mathbf{u}_k\|^2 = 2\beta_k \mathbf{u}_k^T \psi_k + \beta_k^2 \psi_k^T \psi_k$$

e derivando com respeito a β_k e igualando a zero segundo a metodologia LOC:

$$\frac{\partial \Delta V_{\mathbf{u}}}{\partial \beta_k} = 2\mathbf{u}_k^T \psi_k + 2\beta_k \psi_k^T \psi_k = 0$$

Portanto

$$\beta_k = -\frac{\mathbf{u}_k^T \psi_k}{\psi_k^T \psi_k} \tag{3.19}$$

Com este comprimento do passo

$$\Delta V_{\mathbf{u}} = -\frac{(\mathbf{u}_k^T \psi_k)^2}{\psi_k^T \psi_k} \leq 0$$

o que garante o não crescimento desta função de Liapunov ao longo das trajetórias do sistema em malha fechada.

Observe-se que para o valor de β_k calculado:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \beta_k \psi_k = \mathbf{u}_k - \frac{\mathbf{u}_k^T \psi_k}{\psi_k^T \psi_k} \psi_k = \mathbf{u}_k - \frac{\psi_k \psi_k^T}{\psi_k^T \psi_k} \mathbf{u}_k = \left(I - \frac{\psi_k \psi_k^T}{\psi_k^T \psi_k} \right) \mathbf{u}_k$$

isto é, $\mathbf{u}_{k+1}$ é a projeção de $\mathbf{u}_k$ sobre o plano normal a ψ_k.

A partir deste ponto serão subsituidas as expressões φ_k segundo cada escolha de planta e ψ_k segundo cada escolha de controlador.

1) Primeira escolha

$\varphi_k = \mathbf{u}_k \qquad \psi_k = -\Gamma\mathbf{u}_k + D_\mathbf{f}^T\mathbf{r}_k \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.18) e (3.19):

$$\alpha_k = \frac{\mathbf{r}_k^T D_\mathbf{f}\mathbf{u}_k}{\mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f}\mathbf{u}_k} \qquad \beta_k = \frac{\mathbf{u}_k^T(\Gamma\mathbf{u}_k - D_\mathbf{f}^T\mathbf{r}_k)}{\|\Gamma\mathbf{u}_k - D_\mathbf{f}^T\mathbf{r}_k\|^2} \tag{3.20}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados naqueles pontos onde $\mathbf{u}_k \in \mathcal{N}(D_\mathbf{f}(\mathbf{x}_k))$, se $\Gamma\mathbf{u}_k = -D_\mathbf{f}^T\mathbf{f}(\mathbf{x}_k)$, ou ainda se $\mathbf{u}_k = \mathbf{0}$.

2) Segunda escolha

$\varphi_k = \mathbf{u}_k \qquad \psi_k = -D_\mathbf{f}^T(D_\mathbf{f}\mathbf{u}_k - \mathbf{r}_k)$

E substituindo em (3.18) e (3.19):

$$\alpha_k = \frac{\mathbf{r}_k^T D_\mathbf{f}\mathbf{u}_k}{\mathbf{u}_k^T D_\mathbf{f}^T D_\mathbf{f}\mathbf{u}_k} \qquad \beta_k = \frac{\mathbf{u}_k^T D_\mathbf{f}^T(D_\mathbf{f}\mathbf{u}_k - \mathbf{r}_k)}{\|D_\mathbf{f}^T(D_\mathbf{f}\mathbf{u}_k - \mathbf{r}_k)\|^2} \tag{3.21}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados naqueles pontos onde $\mathbf{u}_k \in \mathcal{N}(D_\mathbf{f}(\mathbf{x}_k))$, se $D_\mathbf{f}(\mathbf{x}_k)\mathbf{u}_k = -\mathbf{f}(\mathbf{x}_k)$, ou ainda se $\mathbf{u}_k = \mathbf{0}$.

3) Terceira escolha

$\varphi_k = D_\mathbf{f}^{-1}\Gamma\mathbf{u}_k \qquad \psi_k = \Gamma(\mathbf{r}_k - \mathbf{u}_k) \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.18) e (3.19):

$$\alpha_k = \frac{\mathbf{r}_k^T\Gamma\mathbf{u}_k}{\mathbf{u}_k^T\Gamma^2\mathbf{u}_k} \qquad \beta_k = \frac{\mathbf{u}_k^T\Gamma(\mathbf{u}_k - \mathbf{r}_k)}{\|\Gamma(\mathbf{u}_k - \mathbf{r}_k)\|^2} \tag{3.22}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = -\mathbf{f}(\mathbf{x}_k)$, ou ainda se $\mathbf{u}_k = \mathbf{0}$.

4) Quarta escolha

$\varphi_k = D_\mathbf{f}^T\Gamma\mathbf{u}_k \qquad \psi_k = -\Gamma\mathbf{u}_k + \Gamma D_\mathbf{f} D_\mathbf{f}^T\mathbf{r}_k \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.18) e (3.19):

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} D_{\mathbf{f}}^T \Gamma \mathbf{u}_k}{\mathbf{u}_k^T \Gamma (D_{\mathbf{f}} D_{\mathbf{f}}^T)^2 \Gamma \mathbf{u}_k} \qquad \beta_k = \frac{\mathbf{u}_k^T (\Gamma \mathbf{u}_k - \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T \mathbf{r}_k)}{\|\Gamma \mathbf{u}_k - \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T \mathbf{r}_k\|^2} \tag{3.23}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados naqueles pontos onde $\Gamma \mathbf{u}_k \in \mathcal{N}(D_{\mathbf{f}}^T(\mathbf{x}_k))$, $\Gamma \mathbf{u}_k \in \mathcal{N}(D_{\mathbf{f}}(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k))$, se $\Gamma \mathbf{u}_k = -\Gamma D_{\mathbf{f}}(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)$, ou ainda se $\mathbf{u}_k = \mathbf{0}$.

5) Quinta escolha

$$\varphi_k = \mathbf{u}_k \qquad \psi_k = D_{\mathbf{f}}^T \mathbf{r}_k - (\mathbf{u}_k^T \Gamma \mathbf{u}_k) \mathrm{sgn}(\mathbf{u}_k) \quad \Gamma = \Gamma^T \succ 0$$

E substituindo em (3.18) e (3.19):

$$\alpha_k = \frac{\mathbf{r}_k^T D_{\mathbf{f}} \mathbf{u}_k}{\mathbf{u}_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \mathbf{u}_k} \qquad \beta_k = \frac{\mathbf{u}_k^T (\mathbf{u}_k^T \Gamma \mathbf{u}_k \mathrm{sgn}(\mathbf{u}_k) - D_{\mathbf{f}}^T \mathbf{r}_k)}{\|\mathbf{u}_k^T \Gamma \mathbf{u}_k \mathrm{sgn}(\mathbf{u}_k) - D_{\mathbf{f}}^T \mathbf{r}_k\|^2} \tag{3.24}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados naqueles pontos onde $\mathbf{u}_k \in \mathcal{N}(D_{\mathbf{f}}(\mathbf{x}_k))$, se $\mathbf{u}_k^T \Gamma \mathbf{u}_k \mathrm{sgn}(\mathbf{u}_k) = -D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)$, ou ainda se $\mathbf{u}_k = \mathbf{0}$.

Observe-se que, em geral, os comprimentos dos passos α_k e β_k não têm por que ser necessariamente positivos, mas mesmo que sejam negativos está garantido o não crescimento das variações das funções de Liapunov $\Delta V_{\mathbf{r}}$ e $\Delta V_{\mathbf{u}}$. Outra observação pertinente é que essas variações só serão zero se $\alpha_k = 0$ ou se $\beta_k = 0$, respectivamente, e portanto a trajetória da variável de controle ou da variável de estado se estacionarão nesse ponto. Evidentemente, se apenas a variável de estado se estacionar em uma iteração k tal que $\alpha_k = 0$, mas $\mathbf{u}_k \neq \mathbf{0}$, na iteração seguinte a trajetória continuará seu percurso.

Os algoritmos discretos serão apresentados resumidamente na seguinte tabela.

nome	α_k	β_k
DC1	$\frac{\mathbf{r}_k^T D_{\mathbf{f}} \mathbf{u}_k}{\mathbf{u}_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \mathbf{u}_k}$	$\frac{\mathbf{u}_k^T (\Gamma \mathbf{u}_k - D_{\mathbf{f}}^T \mathbf{r}_k)}{\|\Gamma \mathbf{u}_k - D_{\mathbf{f}}^T \mathbf{r}_k\|^2}$
DC2	$\frac{\mathbf{r}_k^T D_{\mathbf{f}} \mathbf{u}_k}{\mathbf{u}_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \mathbf{u}_k}$	$\frac{\mathbf{u}_k^T D_{\mathbf{f}}^T (D_{\mathbf{f}} \mathbf{u}_k - \mathbf{r}_k)}{\|D_{\mathbf{f}}^T (D_{\mathbf{f}} \mathbf{u}_k - \mathbf{r}_k)\|^2}$
DC3	$\frac{\mathbf{r}_k^T \Gamma \mathbf{u}_k}{\mathbf{u}_k^T \Gamma^2 \mathbf{u}_k}$	$\frac{\mathbf{u}_k^T \Gamma (\mathbf{u}_k - \mathbf{r}_k)}{\|\Gamma (\mathbf{u}_k - \mathbf{r}_k)\|^2}$
DC4	$\frac{\mathbf{r}_k^T D_{\mathbf{f}} D_{\mathbf{f}}^T \Gamma \mathbf{u}_k}{\mathbf{u}_k^T \Gamma (D_{\mathbf{f}} D_{\mathbf{f}}^T)^2 \Gamma \mathbf{u}_k}$	$\frac{\mathbf{u}_k^T (\Gamma \mathbf{u}_k - \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T \mathbf{r}_k)}{\|\Gamma \mathbf{u}_k - \Gamma D_{\mathbf{f}} D_{\mathbf{f}}^T \mathbf{r}_k\|^2}$
DC5	$\frac{\mathbf{r}_k^T D_{\mathbf{f}} \mathbf{u}_k}{\mathbf{u}_k^T D_{\mathbf{f}}^T D_{\mathbf{f}} \mathbf{u}_k}$	$\frac{\mathbf{u}_k^T (\mathbf{u}_k^T \Gamma \mathbf{u}_k \mathrm{sgn}(\mathbf{u}_k) - D_{\mathbf{f}}^T \mathbf{r}_k)}{\|\mathbf{u}_k^T \Gamma \mathbf{u}_k \mathrm{sgn}(\mathbf{u}_k) - D_{\mathbf{f}}^T \mathbf{r}_k\|^2}$

Tabela 3.2 Algoritmos discretos de segunda ordem deduzidos com a metodologia CLF/LOC

3.2.3 Simulações dos algoritmos de segunda ordem discretos

Os algoritmos serão implementados com diversas funções de teste de duas variáveis, a fim de poderem ser graficados. O valor inicial da variável de controle $\mathbf{u}_0$ foi diferente segundo o algoritmo, a fim de evitar divisões por zero tanto no cálculo do comprimento do passo α_k como no de β_k, e também com o intuito de direcionar a variável $\dot{\mathbf{x}}_0$ de uma maneira conveniente.

Estes valores iniciais foram escolhidos:

DC1: $\mathbf{u}_0 = -\Gamma^{-1}\mathbf{f}(\mathbf{x}_0)$

DC2: $\mathbf{u}_0 = -D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_0)$

DC3: $\mathbf{u}_0 = -\Gamma^{-1} D_{\mathbf{f}} D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_0)$

DC4: $\mathbf{u}_0 = -\Gamma^{-1} \mathbf{f}(\mathbf{x}_0)$

DC5: $\mathbf{u}_0 = -D_{\mathbf{f}}^T \mathbf{f}(\mathbf{x}_0)$

A seguir, serão apresentadas simulações realizadas com as mesmas funções de teste dos algoritmos contínuos. As simulações foram realizadas com o programa Matlab 6. Foi adotado como critério de parada se deter na iteração tal que a norma da função $\|\mathbf{f}(\mathbf{x}_k)\|_2 < 0.01$. As trajetórias foram consideradas divergentes se não conseguiram atingir um zero em até 500 iterações.

Os tempos de demora das execuções também foram testados, embora esses resultados não foram agregados aqui por serem inconclusivos; isto porque diferentes execuções de um mesmo algoritmo e a partir do mesmo ponto inicial demoraram diferentes durações, sempre na ordem das décimas de segundo.

3.2.3.1 Simulações dos algoritmos discretos com a função de Camelback

Para simular com a função de Camelback (3.9) foram escolhidas as mesmas constantes que no caso contínuo. Os ganhos foram escolhidos: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = 0.5I$, e DC5: $\Gamma = 5I$. Os algoritmos também são testados a partir dos pontos iniciais $\mathbf{x}_0 = [-1\ 1.5]^T$, $\mathbf{x}_0 = [0\ 1.5]^T$, $\mathbf{x}_0 = [2\ 1]^T$.

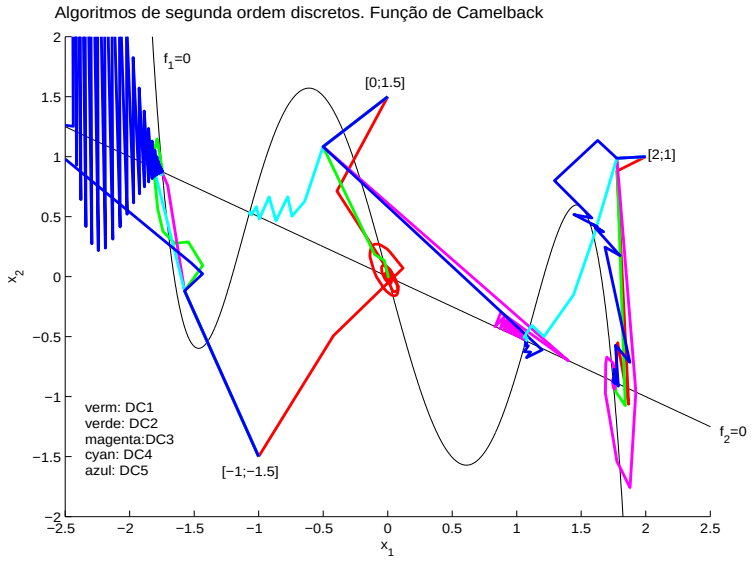

Figura 3.6: Simulações dos algoritmos discretos para a função de Camelback

Observa-se aqui também que as trajetórias convergiram a zeros a partir dos pontos iniciais testados, embora em alguns casos convergiram para zeros diferentes mesmo quando partindo de um ponto inicial comum. Aqui pode se apreciar o chattering da trajetória gerada pelo algoritmo DC5 a partir do ponto inicial $\mathbf{x}_0 = [-1 \ -1.5]^T$.

3.2.3.2 Simulações dos algoritmos discretos com a função de Rosenbrock

Para simular a função de Rosenbrock (3.12), foram escolhidas as mesmas constantes, $a = 0.5$ e $b = 1$, que para as funções contínuas. Com estas constantes, a função apresenta um único zero em $\mathbf{x}^* = [1 \ 1]^T$. Os ganhos foram escolhidos: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = I$, e DC5: $\Gamma = I$. Foram testados os algoritmos a partir dos pontos iniciais $\mathbf{x}_0 = [0.5 \ 2]^T$, $\mathbf{x}_0 = [1.5 \ -0.5]^T$, $\mathbf{x}_0 = [-0.5 \ 1]^T$.

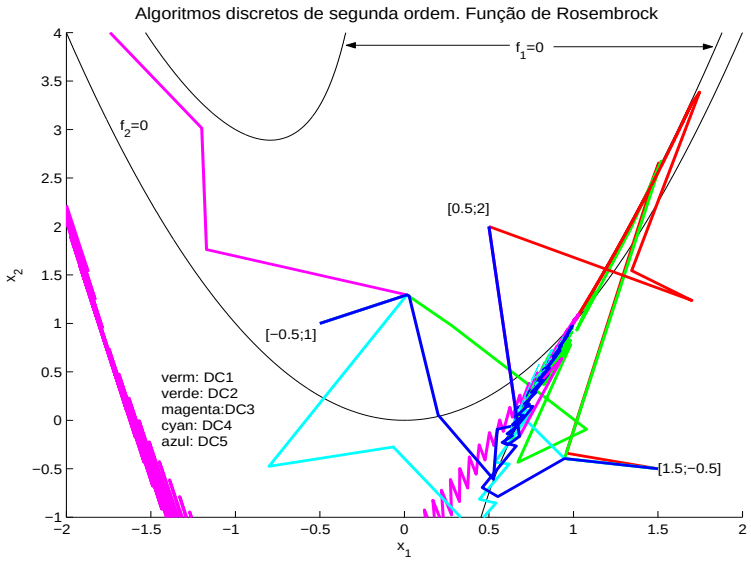

Figura 3.7: Simulações dos algoritmos discretos para a função de Rosenbrock

A função de Rosenbrock é uma função que apresenta altos valores do gradiente. Por esta razão, os algoritmos de segunda ordem encontraram uma dificuldade maior para atingir o único zero desta função. A partir do ponto inicial $\mathbf{x}_0 = [-0.5 \ 1]^T$, a trajetória gerada pelo algoritmo DC1 não convergiu em até 500 iterações; a partir de $\mathbf{x}_0 = [1.5 \ -0.5]^T$, foi a gerada pelo algoritmo DC3 que não convergiu. Observe-se que aqui, assim como no caso contínuo, a trajetória gerada pelo algoritmo DC3 a partir do ponto inicial $\mathbf{x}_0 = [-0.5 \ 1]^T$ parece acompanhar o locus da função $\det(D_{\mathbf{f}}(\mathbf{x})) = 0$.

3.2.3.3 Simulações dos algoritmos discretos com a função de Branin

Para simular a função de Branin (equação 3.14), foram escolhidas as mesmas constantes que no caso contínuo. A constante c também foi escolhida inicialmente igual a zero. Os ganhos foram escolhidos: DC1: $\Gamma = 5I$, DC3: $\Gamma = 5I$, DC4: $\Gamma = 0.5I$, e DC5: $\Gamma = 5I$. Os pontos iniciais testados são $\mathbf{x}_0 = [0.25 \ -0.4]^T$, $\mathbf{x}_0 = [0.75 \ 0.6]^T$, $\mathbf{x}_0 = [1.25 \ -0.6]^T$.

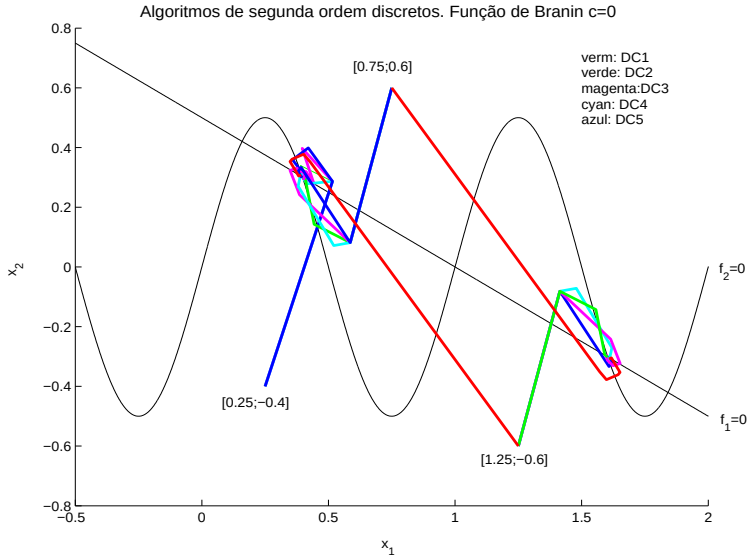

Figura 3.8: Simulações dos algoritmos discretos para a função de Branin, $c = 0$

Aqui, apenas o algoritmo DC1 não gerou uma trajetória capaz de atingir um zero em até 500 iterações a partir do ponto inicial $\mathbf{x}_0 = [0.25 \ -0.4]^T$. Em todos os outros casos, todas as trajetórias atingiram um zero, mesmo que, partindo do mesmo ponto inicial, as trajetórias geradas pelos diferentes algoritmos tenham atingido zeros diferentes.

Aumentando o valor da constante c para 0.13, a função continua apresentando 5 zeros. Neste caso, os pontos iniciais testados foram $\mathbf{x}_0 = [0.25 \ -0.4]^T$, $\mathbf{x}_0 = [1.25 \ -0.6]^T$, $\mathbf{x}_0 = [0.75 \ 0.6]^T$, $\mathbf{x}_0 = [0.7 \ 0.4]^T$, $\mathbf{x}_0 = [0.25 \ -0.2]^T$, e $\mathbf{x}_0 = [1.7 \ 0.2]^T$, mantendo iguais ganhos.

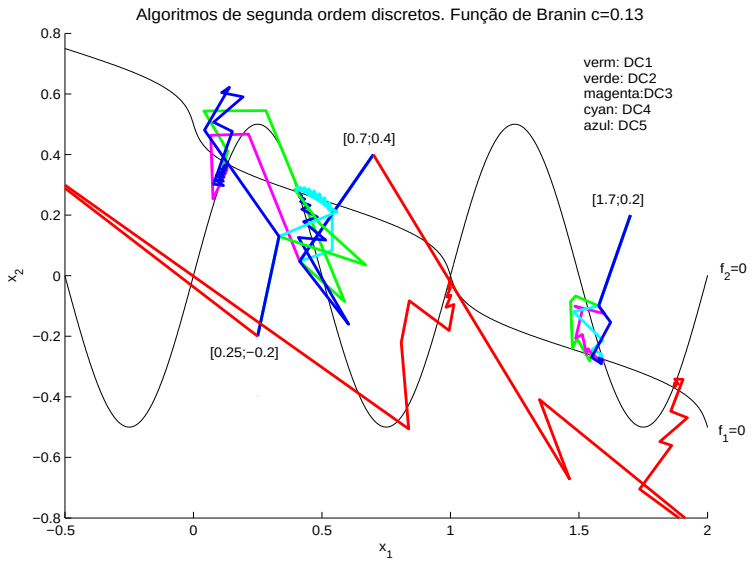

Figura 3.9: Simulações dos algoritmos discretos para a função de Branin, $c = 0.13$

A partir dos três primeiros pontos iniciais mencionados $\mathbf{x}_0 = [0.25 \ -0.4]^T$, $\mathbf{x}_0 = [1.25 \ -0.6]^T$, $\mathbf{x}_0 = [0.75 \ 0.6]^T$, apenas o algoritmo DC1 conseguiu atingir um zero da função, razão pela qual as trajetórias a partir destes pontos foram omitidas no gráfico. A partir de $\mathbf{x}_0 = [0.25 \ -0.2]^T$, a trajetória gerada pelo algoritmo DC3 não conseguiu atingir um zero em até 500 iterações, ao tempo que a partir de $\mathbf{x}_0 = [1.7 \ 0.2]^T$, foi a gerada pelo algoritmo DC1 que não convergiu. Note-se que agora é a trajetória gerada pelo algoritmo DC1 que apresenta grandes excursões pelo plano de fase.

Aumentando ainda a constante c para 1, a função de Branin apresentará agora um total de 15 zeros, dos quais apenas 11 deles aparecem na janela do seguinte gráfico. Os algoritmos foram testados a partir dos pontos iniciais $\mathbf{x}_0 = [0.75 \ 0.6]^T$, $\mathbf{x}_0 = [1.25 \ -0.6]^T$, e $\mathbf{x}_0 = [0.25 \ -0.4]^T$, mantendo iguais os ganhos.

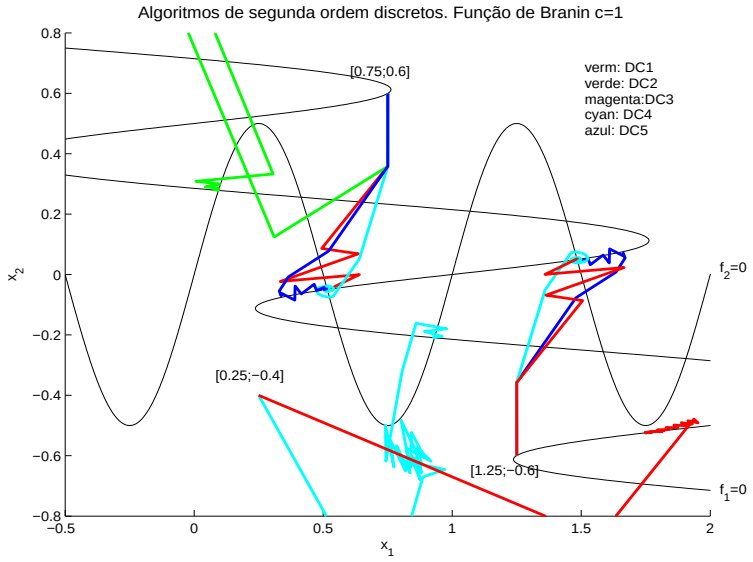

Figura 3.10: Simulações dos algoritmos discretos para a função de Branin, $c = 1$

A partir do ponto inicial $\mathbf{x}_0 = [0.75\ 0.6]^T$, as trajetórias geradas por DC1, DC2, DC4 e DC5 atingiram zeros da função. A partir de $\mathbf{x}_0 = [1.25\ \ -0.6]^T$, apenas a gerada pelos algoritmos DC1, DC4 e DC5 convergiram. A partir de $\mathbf{x}_0 = [0.25\ -0.4]^T$, a gerada por DC4 convergiu. Uma conclusão importante é que os algoritmos de segunda ordem, e nesta simulação se observa claramente esse resultado, embora geram trajetórias com excursões de maior amplitude, parecem ter maior capacidade de atingir um zero da função se comparados com os algoritmos de primeira ordem.

A seguinte tabela mostra o número de iterações empregados por cada algoritmo discreto a partir de cada um dos pontos iniciais testados para todas as funções utilizadas.

função	$\mathbf{x}_0$	DC1	DC2	DC3	DC4	DC5
Branin $c = 0$	$[0.25\ -0.4]^T$	-	5	8	8	5
	$[1.25\ -0.6]^T$	6	4	8	17	6
	$[0.75\ 0.6]^T$	6	4	8	17	6
Branin $c = 0.13$	$[0.25\ -0.4]^T$	23	-	-	-	-
	$[1.25\ -0.6]^T$	42	-	-	-	-
	$[0.75\ 0.6]^T$	44	-	-	-	-
	$[0.7\ 0.4]^T$	16	8	26	32	17
	$[0.25\ -0.2]^T$	14	6	-	6	24
	$[1.7\ 0.2]^T$	-	12	17	8	8
Branin $c = 1$	$[0.75\ 0.6]^T$	8	82	-	71	16
	$[1.25\ -0.6]^T$	8	-	-	71	16
	$[0.25\ -0.4]^T$	408	-	-	33	-
Camelback	$[-1\ -1.5]^T$	103	12	18	4	59
	$[2\ 1]^T$	10	5	74	17	12
	$[0\ 1.5]^T$	10	5	74	17	12
Rosenbrock	$[0.5\ 2]^T$	469	7	21	111	29
	$[1.5\ -0.5]^T$	357	8	-	41	17
	$[-0.5\ 1]^T$	-	15	141	21	41

Tabela 3.3: Número de iterações realizadas pelos algoritmos discretos onde - indica que o algoritmo não atingiu um zero em até 500 iterações

3.3 Algoritmos de segunda ordem para achar mínimos de funções escalares como sistemas dinâmicos de controle

No capítulo 2, referente a algoritmos de primeira ordem, também foram desenvolvidos algoritmos para achar mínimos de funções escalares convexas. Esses algoritmos foram interpretados como sistemas dinâmicos de controle em malha fechada e derivados através da metodologia de funções de Liapunov de controle. Posteriormente foram discretizados por Euler e o comprimento do passo otimizado pelo método de controle ótimo de Liapunov.

Propõe-se aqui atingir idêntico objetivo mas utilizando algoritmos de segunda ordem, isto é, algoritmos cuja versão contínua pode ser representado por uma ODE de segunda ordem. Também aqui os algoritmos serão interpretados como sistemas dinâmicos de controle em malha fechada, onde agora não existirá apenas uma dinâmica na planta do sistema (de escolha arbitrária), mas também no controlador.

Seja uma função escalar $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, contínua e com derivadas parciais contínuas, tal que para todo $\mathbf{x} \in \text{dom}(\phi(\mathbf{x}))$, $\phi(\mathbf{x}) \geq \phi(\mathbf{x}^*) = \phi_{\min}$, sendo $\mathbf{x}^*$ o ponto correspondente ao mínimo global da função, assumindo que este existe e é finito.

A principal motivação para a utilização de algoritmos de segunda ordem para achar o mínimo desta função consiste no seguinte: caso a função seja não convexa, as trajetórias $\mathbf{x}(t)$, a partir de um ponto inicial $\mathbf{x}(0) = \mathbf{x}_0 \in \text{dom}(\phi(\mathbf{x}))$ geradas pelos algoritmos de primeira ordem convergirão a um ponto de gradiente zero, assumindo que estes existem, o qual pode corresponder a um máximo, uma sela, ou um mínimo local; utilizando algoritmos de segunda ordem, espera-se que, com uma adequada escolha dos ganhos, as trajetórias geradas por estes sejam capazes de ultrapassar os pontos de gradiente zero não desejados para convergir ao mínimo global da função $\mathbf{x}^*$.

Algoritmos de segunda ordem para achar mínimos de funções escalares não são novos na bibliografia.

O mais conhecido é o algoritmo chamado *"heavy ball with friction"* (HBF), mencionado em [3] e [19]. Este algoritmo pode ser representado pela seguinte equação diferencial de segunda ordem:

$$\ddot{\mathbf{x}} + \Gamma\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = 0 \tag{3.25}$$

onde $\Gamma = \Gamma^T \succ 0$ (nas referências este ganho aparece como um escalar positivo).

Este é um sistema dissipativo com analogias mecânicas, onde o ganho Γ pode ser interpretado como um coeficiente de atrito. Assim, as trajetórias $\mathbf{x}(t)$ a partir de um ponto inicial $\mathbf{x}(0) = \mathbf{x}_0$ geradas por este algoritmo serão similares às trajetórias descritas por uma bola de massa unitária, lançada sobre uma superfície descrita pela função $\phi(\mathbf{x})$, e com coeficiente de atrito Γ. Como acontece neste caso, dependendo do coeficiente de atrito, e da posição e velocidade iniciais, a bola, e assim as trajetórias geradas por HBF, pode ser capaz de ulrapassar um mínimo local para se estacionar em um mínimo global.

Alvarez, Attouch, *et.al.* ([3]), apresentam uma modificação do algoritmo HBF, ao qual chamaram *"dynamical inertial Newton-like system"* (DIN). Esta consiste no agregado de um termo de amortecimento espacial, e não apenas temporal, como no caso do HBF.

$$\ddot{\mathbf{x}} + a\dot{\mathbf{x}} + b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = 0 \tag{3.26}$$

onde a e b são ganhos escalares positivos.

Os autores mencionam que o algoritmo HBF é uma melhora de algoritmo de Newton, e quando a função $\phi(\mathbf{x})$ é convexa as trajetórias convergem rapidamente ao mínimo global, caso este exista. Porém, as trajetórias podem apresentar oscilações espaciais, as quais estariam amortecidas pelo termo $b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}}$ no algoritmo DIN, mesmo que o hessiano da função seja degenerado (ver [3, s. 1]). Assim, as trajetórias geradas por este seriam melhor comportadas no sentido de convergir mais rapidamente ao mínimo da função.

Cabot ([19]), propõe o seguinte sistema dinâmico de segunda ordem:

$$\ddot{\mathbf{x}} + \gamma\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) + \epsilon(t)\nabla U(\mathbf{x}) = 0 \tag{3.27}$$

onde γ é um ganho escalar positivo, $\epsilon(t)$ é uma função escalar positiva que tende a zero quando t tende a infinito, e $U(\mathbf{x})$ é uma função de costo escalar positiva.

O autor demonstra que se $\phi(\mathbf{x})$ e $U(\mathbf{x})$ são convexas, as trajetórias de (3.27) convergem a um mínimo de U sobre o conjunto argmin(ϕ), assumindo este não vazio. A aplicação deste algoritmo é a seguinte: se $\phi(\mathbf{x})$ é convexa e o conjunto argmin(ϕ) for convexo, o problema transforma-se em um caso particular de otimização convexa onde o conjunto viável $\chi = \{\text{argmin}(\phi(\mathbf{x}))\}$, o qual pode ser descrito como:

$$\begin{aligned} \min \quad & U(\mathbf{x}) \\ \text{s.t} \quad & \mathbf{x} \in \text{argmin}(\phi(\mathbf{x})) \end{aligned}$$

Fradkov e Pogromsky ([29, s. 2.5]) tratam de um tipo particular de algoritmos chamados *"speed gradient"*. Estes algoritmos podem ser algoritmos de segunda ordem também, dependendo da escolha das funções involucradas. Resumidamente, a idéia é a seguinte: seja um sistema dinâmico não autônomo da forma

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

onde $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_+ \to \mathbb{R}^n$ é contínua por partes em t e continuamente diferenciável em $\mathbf{x}$ e $\mathbf{u}$, e seja uma função objetivo escalar não negativa $Q(\mathbf{x}(t), t) : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^1$, para projetar o algoritmo de controle procura-se uma função escalar

$$\dot{Q} = \varpi(\mathbf{x}, \mathbf{u}, t) = \frac{\partial Q(\mathbf{x}, t)}{\partial t} + \nabla Q^T(\mathbf{x}, t)\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

Então calcula-se o gradiente de ϖ com respeito a variável de controle $\mathbf{u}$

$$\left(\frac{\partial \varpi(\mathbf{x}, \mathbf{u}, t)}{\partial \mathbf{u}}\right)^T = \left(\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u}, t)}{\partial \mathbf{u}}\right)^T \nabla Q(\mathbf{x}, t)$$

Finalmente se calcula o algoritmo de controle da seguinte forma

$$\dot{\mathbf{u}} = -\Gamma \frac{\partial \varpi(\mathbf{x}, \mathbf{u}, t)}{\partial \mathbf{u}} \tag{3.28}$$

onde $\Gamma = \Gamma^T \succ 0$.

Observe-se que o fato de existir uma dinâmica na lei de controle, dada por (3.28), é o que caracteriza o sistema como de segunda ordem.

Por exemplo, caso a função escalar de custo $Q(\mathbf{x}, t)$ seja a função objetivo $\phi(\mathbf{x})$, e a

lei de atualização das variáveis $\mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ seja $\mathbf{f}(\mathbf{u}) = \mathbf{u}$, o sistema pode ser representado como:

$$\ddot{\mathbf{x}} + \Gamma \nabla \phi(\mathbf{x}) = 0 \tag{3.29}$$

onde fica explícita a qualidade de segunda ordem do sistema.

3.4 Algoritmos contínuos de segunda ordem para achar mínimos de funções

Seja uma função $\phi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ limitada inferiormente no ponto $\mathbf{x}^*$ tal que para todo $\mathbf{x} \in \mathbb{R}^n : \ \phi(\mathbf{x}) \geq \phi(\mathbf{x}^*) = \phi_{\min}$, contínua e com derivadas parciais contínuas, será escolhida como função de Liapunov candidata a uma função escalar dependente de todos os estados do sistema, isto é, da variável de estado $\mathbf{x}$ e da variável de controle $\mathbf{u}$.

$$V(\mathbf{x}, \mathbf{u}) = \phi(\mathbf{x}) - \phi_{\min} + \frac{\mathbf{u}^T \mathbf{u}}{2} \geq 0 \tag{3.30}$$

a qual é maior que zero para todo $\mathbf{x} \neq \mathbf{x}^*$ e $\mathbf{u} \neq \mathbf{0}$.

A escolha desta função de Liapunov candidata, a qual depende de todos os estados do sistema (variável de controle e variável de estado), equivale a assumir, no sistema dinâmico em malha fechada, o sinal de referência como $\phi_{\min}$, e a resposta da planta como $\phi(\mathbf{x})$, de maneira tal que o primeiro termo de (3.30) corresponde a $-r = \phi(\mathbf{x}) - \phi_{\min}$.

Derivando (3.30) com respeito ao tempo:

$$\dot{V}(\mathbf{x}, \mathbf{u}) = \nabla^T \phi(\mathbf{x}) \dot{\mathbf{x}} + \mathbf{u}^T \dot{\mathbf{u}} \tag{3.31}$$

Este é o ponto de partida para a dedução de diferentes algoritmos de segunda ordem como sistemas dinâmicos de controle em malha fechada, onde $\dot{\mathbf{x}}$ dependerá da escolha da planta e $\dot{\mathbf{u}}$ dependerá da escolha do controlador, e ambos devem satisfacer que $\dot{V}$ seja negativa semidefinida, garantindo assim a estabilidade do sistema, segundo o método de funções de Liapunov de controle.

1) Primeira escolha

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Substituindo em (3.31): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla\phi(\mathbf{x}))$

Controlador $\quad \dot{\mathbf{u}} = -\kappa \text{sgn}(\mathbf{u}) - \nabla\phi(\mathbf{x})$

onde $\kappa > 0$. Substituindo na derivada da função de Liapunov candidata:

$$\dot{V} = -\kappa\|\mathbf{u}\|_1 \leq 0$$

Pelo lema de Barbalat, $\dot{V}$ tende a zero, pelo qual $\dot{\mathbf{u}}$ e $\mathbf{u}$ tendem para zero também. Pela equação do controlador, portanto, $\nabla\phi(\mathbf{x})$ também tende para zero.

Observe-se que isto garante que o algoritmo tenderá a um zero do gradiente da função, mas, caso a função seja não convexa, podem existir outros pontos de gradiente zero diferentes do mínimo global. Portanto, pode acontecer que $V(\mathbf{x}, \mathbf{u}) = \phi(\mathbf{x}) - \phi_{\min} > 0$, podendo depender da escolha do ganho escalar κ que as trajetórias sejam capazes de ultrapassar este ponto para se estacionarem no mínimo global. Máximos e selas efetivamente são pontos de equilíbrio do sistema, porém instáveis.

O controlador é um sistema a estrutura variável, por possuir sua equação característica um lado direito descontínuo. O fato da estrutura variável existir no controlador e não na planta, indicará que as trajetórias $\mathbf{u}(t)$ poderão apresentar um modo deslizante, e não as trajetórias $\mathbf{x}(t)$.

Este algoritmo pode ser escrito como uma ODE de segunda ordem, da forma:

$$\ddot{\mathbf{x}} + \kappa \text{sgn}(\dot{\mathbf{x}}) + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{3.32}$$

O algoritmo apresentado será chamado simplesmente de MI1 (algoritmo de minimização 1).

2) Segunda escolha

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Substituindo em (3.31): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \nabla\phi(\mathbf{x}))$

Controlador $\quad \dot{\mathbf{u}} = -(\nabla^2\phi(\mathbf{x}))^2\mathbf{u} - \nabla\phi(\mathbf{x})$

onde $\nabla^2\phi(\mathbf{x})$ é o hessiano da função. Substituindo na derivada da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T(\nabla^2\phi(\mathbf{x}))^2\mathbf{u} \leq 0$$

Similarmente ao caso MI1, este algoritmo poderia se estacionar em um ponto de gradiente zero, não necessariamente no mínimo global, e neste caso não há ganhos a ajustar para permitir a ultrapassagem de um ponto de convergência não desejado.

Este algoritmo pode ser representado como uma ODE de segunda ordem, da forma:

$$\ddot{\mathbf{x}} + (\nabla^2\phi(\mathbf{x}))^2\dot{\mathbf{x}} + \nabla\phi(\mathbf{x}) = \mathbf{0} \tag{3.33}$$

O algoritmo será chamado simplesmente de MI2.

3) Terceira escolha

Planta $\quad \dot{\mathbf{x}} = \nabla^{-2}\phi(\mathbf{x})\Gamma\mathbf{u}$

onde $\Gamma = \Gamma^T \succ 0$. Substituindo em (3.31): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \Gamma\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}))$

Controlador $\quad \dot{\mathbf{u}} = -\Gamma\mathbf{u} - \Gamma\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x})$

e substituindo na derivada da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T\Gamma\mathbf{u}$$

Pelo lema de Barbalat, $\dot{V}$ tende para zero quando t tende a infinito, com o qual $\mathbf{u}$ e $\dot{\mathbf{u}}$ também tendem para zero. Pela equação do controlador, $\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x})$ tende para zero também. Observe-se que isto implica que as trajetórias poderiam se estacionar não apenas em um ponto de gradiente zero, mas também em um ponto tal que $\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^{-2}\phi(\mathbf{x}))$, razão pela qual a convergência é apenas local. Além disso, por utilizar a inversa do hessiano no cálculo da planta e do controlador, este algoritmo não está definido naqueles pontos onde o hessiano é singular, isto é, onde $\det(\nabla^2\phi(\mathbf{x})) = 0$.

Este algoritmo não pode ser diretamente representado como uma ODE de segunda

ordem, mas realizando uma conveniente mudança de variáveis chega-se a:

$$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma D_{\mathbf{y}}^{-1}\mathbf{y} \tag{3.34}$$

onde $\Gamma\mathbf{y} = \nabla\phi(\mathbf{x})$ e $\Gamma D_{\mathbf{y}} = \nabla^2\phi(\mathbf{x})$.

Este algoritmo será chamado simplesmente de MI3.

4) Quarta escolha

Planta $\quad \dot{\mathbf{x}} = \nabla^2\phi(\mathbf{x})\Gamma\mathbf{u}$

onde $\Gamma = \Gamma^T \succ 0$. Substituindo em (3.31): $\dot{V} = \mathbf{u}^T(\dot{\mathbf{u}} + \Gamma\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}))$

Controlador $\quad \dot{\mathbf{u}} = -\Gamma\mathbf{u} - \Gamma\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x})$

e substituindo na derivada da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T\Gamma\mathbf{u}$$

Realizando a mesma análise que nos casos anteriores, observa-se que este algoritmo poderia se estacionar não apenas em um ponto de gradiente zero, mas também em um ponto $\mathbf{x}$ tal que $\nabla\phi(\mathbf{x}) \in \mathcal{N}(\nabla^2\phi(\mathbf{x}))$, razão pela qual a convergência das trajetórias é apenas local.

Este algoritmo também não pode ser facilmente representado como uma ODE de segunda ordem, mas realizando uma mudança conveniente de variáveis, chega-se a:

$$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma\nabla^2\phi(\mathbf{x})\nabla\phi(\mathbf{x}) = \mathbf{0} \tag{3.35}$$

onde $\nabla^2\phi(\mathbf{x})\Gamma\dot{\mathbf{y}} = \dot{\mathbf{x}}$.

Este algoritmo será chamado simplesmente de MI4.

5) Quinta escolha

É possível também derivar outros algoritmos utilizando outras funções de Liapunov

candidatas. Por exemplo, escolhendo

$$V(\mathbf{x},\mathbf{u}) = (a+b)(\phi(\mathbf{x}) - \phi_{\min}) + \frac{\|\nabla\phi(\mathbf{x}) + \mathbf{u}\|^2}{2} \geq 0 \tag{3.36}$$

onde a e b são ganhos escalares tal que $a + b > 0$.

Observe-se que esta função de Liapunov candidata só é igual a zero para $\mathbf{x} = \mathbf{x}^*$ tal que $\phi(\mathbf{x}^*) = \phi_{\min}$, ponto onde $\nabla\phi(\mathbf{x}^*) = \mathbf{0}$, e para $\mathbf{u} = \mathbf{0}$.

Derivando (3.36) com respeito ao tempo

$$\dot{V} = (a+b)\nabla^T\phi(\mathbf{x})\dot{\mathbf{x}} + (\nabla\phi(\mathbf{x}) + \mathbf{u})^T(\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}} + \dot{\mathbf{u}}) \tag{3.37}$$

Planta $\quad \dot{\mathbf{x}} = \nabla^{-2}\phi(\mathbf{x})\mathbf{u}$

e substituindo em (3.37): $\dot{V} = (a+b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + (\nabla\phi(\mathbf{x}) + \mathbf{u})^T(\mathbf{u} + \dot{\mathbf{u}})$

Controlador $\quad \dot{\mathbf{u}} = -\Gamma\nabla\phi(\mathbf{x}) - (I - \Gamma)\mathbf{u} - \nabla^{-2}\phi(\mathbf{x})(a\mathbf{u} + b\nabla\phi(\mathbf{x}))$

onde $\Gamma = \Gamma^T$. Substituindo na equação derivada da função de Liapunov candidata:

$$\begin{gathered}
\dot{V} = (a+b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + (\nabla\phi(\mathbf{x}) + u)^T \\
(\mathbf{u} - \Gamma\nabla\phi(\mathbf{x}) - (I-\Gamma)\mathbf{u} - \nabla^{-2}\phi(\mathbf{x})(a\mathbf{u} + b\nabla\phi(\mathbf{x}))) = \\
(a+b)\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} + \nabla^T\phi(\mathbf{x})\mathbf{u} - \nabla^T\phi(\mathbf{x})\Gamma\nabla\phi(\mathbf{x}) + \mathbf{u}^T\mathbf{u} \\
-\mathbf{u}^T\Gamma\nabla\phi(\mathbf{x}) - \nabla^T\phi(\mathbf{x})(I-\Gamma)\mathbf{u} - \mathbf{u}^T(I-\Gamma)\mathbf{u} - a\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\mathbf{u} \\
-b\nabla^T\phi(\mathbf{x})\nabla^{-2}\phi(\mathbf{x})\nabla\phi(\mathbf{x}) - a\mathbf{u}^T\nabla^{-2}\phi(\mathbf{x})\mathbf{u} - b\mathbf{u}^T\nabla^{-2}\phi(\mathbf{x})\nabla(\mathbf{x}) = \\
-\nabla^T\phi(\mathbf{x})(\Gamma + b\nabla^{-2}\phi(\mathbf{x}))\nabla\phi(\mathbf{x}) - \mathbf{u}^T(-\Gamma + a\nabla^{-2}\phi(\mathbf{x}))\mathbf{u} \leq 0
\end{gathered}$$

onde foi utilizado o fato que o hessiano de uma função escalar é simétrico.

Observe-se que se a, b e Γ forem escolhidos de maneira tal que $\Gamma + b\nabla^{-2}\phi(\mathbf{x}) \succ 0$ e $-\Gamma + a\nabla^{-2}\phi(\mathbf{x}) \succ 0$ em uma determinada região, então $\dot{V}$ tende para zero dentro dessa região, razão pela qual nos referimos ao carácter local da convergência. Se $\dot{V}$ tende para zero, então $\mathbf{u}$, $\dot{\mathbf{u}}$ e $\nabla\phi(\mathbf{x})$ tendem para zero também, mas o algoritmo poderá se estacionar em qualquer ponto de gradiente zero.

Também aqui, resulta difícil representar este algoritmo como uma ODE de segunda

ordem, mas fazendo uma conveniente mudança de variáveis, chega-se a:

$$\ddot{\mathbf{y}} + (I - \Gamma + a\nabla^{-2}\phi(\mathbf{x}))\dot{\mathbf{y}} + (\Gamma + b\nabla^{-2}\phi(\mathbf{x}))\nabla\phi(\mathbf{x}) = \mathbf{0} \tag{3.38}$$

onde $\dot{\mathbf{y}} = \nabla^2\phi(\mathbf{x})\dot{\mathbf{x}}$.

Este algoritmo será chamado de MI5.

6) HBF

O algoritmo HBF (3.25), também pode ser representado como um sistema dinâmico de controle em malha fechada. Escolhendo a função de Liapunov candidata (3.30), e as seguintes escolhas de planta e controlador:

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$
e substituindo em (3.31): $\dot{V} = \mathbf{u}^T(\nabla\phi(\mathbf{x}) + \dot{\mathbf{u}})$

Controlador $\quad \dot{\mathbf{u}} = -\Gamma\mathbf{u} - \nabla\phi(\mathbf{x})$

onde $\Gamma = \Gamma^T \succ 0$. Substituindo na equação da derivada da função de Liapunov candidata

$$\dot{V} = -\mathbf{u}^T\Gamma\mathbf{u} \leq 0$$

Fazendo a mesma análise dos casos anteriores, por Barbalat, $\mathbf{u}$ e $\dot{\mathbf{u}}$ tendem para zero, e a partir da equação do controlador, observa-se que o algoritmo pode se estacionar em qualquer ponto $\mathbf{x}$ tal que $\nabla\phi(\mathbf{x}) = \mathbf{0}$, ficando na dependência do ganho Γ a possibilidade da trajetória ultrapassar mínimos locais para convergir ao mínimo global.

Fazendo a mudança de variáveis realizada na definição da planta $\dot{\mathbf{x}} = \mathbf{u}$, e aplicando a equação do controlador $\dot{\mathbf{u}} = \ddot{\mathbf{x}} = -\nabla\phi(\mathbf{x}) - \Gamma\dot{\mathbf{x}}$, chega-se a (3.25).

6) DIN

O algoritmo DIN (3.26), também pode ser visto como um sistema dinâmico de controle em malha fechada, com as seguintes escolhas de planta e controlador:

Planta $\quad \dot{\mathbf{x}} = \mathbf{u}$

Controlador $\quad \dot{\mathbf{u}} = -\nabla\phi(\mathbf{x}) - a\mathbf{u} - b\nabla^2\phi(\mathbf{x})\mathbf{u}$

Para analisar a convergência das trajetórias geradas por este algoritmo, será escolhida a seguinte função de Liapunov candidata

$$V(\mathbf{x}, \mathbf{u}) = (ab+1)\phi(\mathbf{x}) + \frac{\|\mathbf{u} + b\nabla\phi(\mathbf{x})\|^2}{2} \tag{3.39}$$

Derivando (3.39) com respeito ao tempo e aplicando as equações da planta e do controlador, isto é, ao longo das trajetórias do sistema em malha fechada:

$$\begin{aligned}
\dot{V} &= (ab+1)\nabla^T\phi(\mathbf{x})\mathbf{u} + (\mathbf{u} + b\nabla\phi(\mathbf{x}))^T(-\nabla\phi(\mathbf{x}) - a\mathbf{u} - b\nabla^2\phi(\mathbf{x})\mathbf{u} + b\nabla^2\phi(\mathbf{x})\mathbf{u}) \\
&= ab\nabla^T\phi(\mathbf{x})\mathbf{u} + \nabla^T\phi(\mathbf{x})\mathbf{u} - \mathbf{u}^T\nabla\phi(\mathbf{x}) - a\|\mathbf{u}\|^2 - b\|\nabla\phi(\mathbf{x})\|^2 + ab\nabla^T\phi(\mathbf{x})\mathbf{u} \\
&= -a\|\mathbf{u}\|^2 - b\|\nabla\phi(\mathbf{x})\|^2
\end{aligned}$$

Por Barbalat, $\dot{V}$ tende para zero e portanto $\mathbf{u}$ e $\nabla\phi(\mathbf{x})$ também tendem para zero, mas a trajetória poderia se estacionar então em um ponto de gradiente zero, dependendo da escolha dos ganhos a possibilidade da trajetória ultrapassar mínimos locais.

Fazendo a mudança de variáveis proposta na planta e no controlador, isto é, $\dot{\mathbf{x}} = \mathbf{u}$ e $\ddot{\mathbf{x}} = \dot{\mathbf{u}} = -\nabla\phi(\mathbf{x}) - a\dot{\mathbf{x}} - b\nabla^2\phi(\mathbf{x})\dot{\mathbf{x}}$, chega-se facilmente a (3.26).

Aqui também, com certeza existem outras funções de Liapunov candidatas, e outras escolhas de planta e controlador que façam a derivada temporal desta negativa semidefinida. As apresentadas aqui são apenas alguns exemplos.

A seguir, serão mostrados os algoritmos de maneira resumida na seguinte tabela:

nome	planta	controlador	ODE
MI1	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\kappa \mathrm{sgn}(\mathbf{u}) - \nabla\phi$	$\ddot{\mathbf{x}} + \kappa \mathrm{sgn}(\dot{\mathbf{x}}) + \nabla\phi = \mathbf{0}$
MI2	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -(\nabla^2\phi)^2\mathbf{u} - \nabla\phi$	$\ddot{\mathbf{x}} + (\nabla^2\phi)^2\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$
MI3	$\dot{\mathbf{x}} = \nabla^{-2}\phi\Gamma\mathbf{u}$	$\dot{\mathbf{u}} = -\Gamma(\mathbf{u} + \nabla^{-2}\phi\nabla\phi)$	$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma D_{\mathbf{y}}^{-1}\mathbf{y} = \mathbf{0}$ $\Gamma\mathbf{y} = \nabla\phi \;\; \Gamma D_{\mathbf{y}} = \nabla^2\phi$
MI4	$\dot{\mathbf{x}} = \nabla^2\phi\Gamma\mathbf{u}$	$\dot{\mathbf{u}} = -\Gamma(\mathbf{u} + \nabla^2\phi\nabla\phi)$	$\ddot{\mathbf{y}} + \Gamma\dot{\mathbf{y}} + \Gamma\nabla^2\phi\nabla\phi = \mathbf{0}$ $\dot{\mathbf{y}} = \nabla^2\phi\dot{\mathbf{x}}$
MI5	$\dot{\mathbf{x}} = \nabla^{-2}\phi\mathbf{u}$	$\dot{\mathbf{u}} = -(I - \Gamma + a\nabla^{-2}\phi)\mathbf{u}$ $-(\Gamma + b\nabla^{-2}\phi)\nabla\phi$	$\ddot{\mathbf{y}} + (I - \Gamma + a\nabla^{-2}\phi)\dot{\mathbf{y}}$ $+(\Gamma + b\nabla^{-2}\phi)\nabla\phi = \mathbf{0}$ $\dot{\mathbf{y}} = \nabla^2\phi\dot{\mathbf{x}}$
HBF	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\nabla\phi - \Gamma\mathbf{u}$	$\ddot{\mathbf{x}} + \Gamma\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$
DIN	$\dot{\mathbf{x}} = \mathbf{u}$	$\dot{\mathbf{u}} = -\nabla\phi - (a + b\nabla^2\phi)\mathbf{u}$	$\ddot{\mathbf{x}} + (a + b\nabla^2\phi)\dot{\mathbf{x}} + \nabla\phi = \mathbf{0}$

Tabela 3.4 Algoritmos contínuos de segunda ordem para achar mínimos de funções escalares derivados com a metodologia CLF/LOC

3.4.1 Simulações dos algoritmos contínuos de segunda ordem para achar mínimos de funções escalares

A seguir, serão realizadas simulações com os algoritmos apresentados na seção anterior. Foram escolhidas, a fim de poderem ser graficadas, funções de duas variáveis. As simulações foram realizadas com o aplicativo Simulink do programa MATLAB 6. A discretização foi realizada pelo método de Euler, com comprimento do passo fixo igual a $0.01s$. e foram executados durante 10s.

Em todos os casos, o valor inicial da variável de controle foi $\mathbf{u}_0 = [0\ 0]^T$. Os ganhos foram variáveis segundo a função e o algoritmo.

3.4.1.1 Simulações dos algoritmos de segunda ordem contínuos com a função de Camelback invertida

A função de Camelback (equação 3.8), foi invertida a fim de apresentar mínimos locais e globais.

Foram escolhidas como constantes $a = -2,\ b = 1.05,\ c = -\frac{1}{6},\ d = -1$ e $e = 0$. Com estas constantes a função apresenta um mínimo global em $\mathbf{x} = [0\ 0]^T$, dois mínimos locais em $\mathbf{x} = [-1.7475\ 0.8737]^T$ e $\mathbf{x} = [1.7475\ -0.8737]^T$, e duas selas em $\mathbf{x} = [-1.0705\ 0.5352]^T$ e $\mathbf{x} = [1.0705\ -0.5352]^T$.

Foram experimentados diversos ganhos e escolhidos aqueles que apresentaram melhores resultados, no sentido das trajetórias convergirem mais diretamente para um mínimo, apresentar menores oscilações, e, principalmente, a fim de conseguir ultrapassar os mínimos locais para se estacionarem no mínimo global. Estes foram

MI1: $\kappa = 1$

MI3: $\Gamma = 8I$

MI4: $\Gamma = 0.1I$

MI5: $\Gamma = 0.1I,\ a = b = 1$

HBF: $\Gamma = I$

DIN: $a = b = 1$

No algortimo MI2, o ganho é $(\nabla^2\phi(\mathbf{x}))^2$, razão pela qual este ganho não pode ser escolhido.

Os pontos iniciais testados foram $\mathbf{x}_0 = [-1\ -1.5]^T$ e $\mathbf{x}_0 = [2\ 1]^T$.

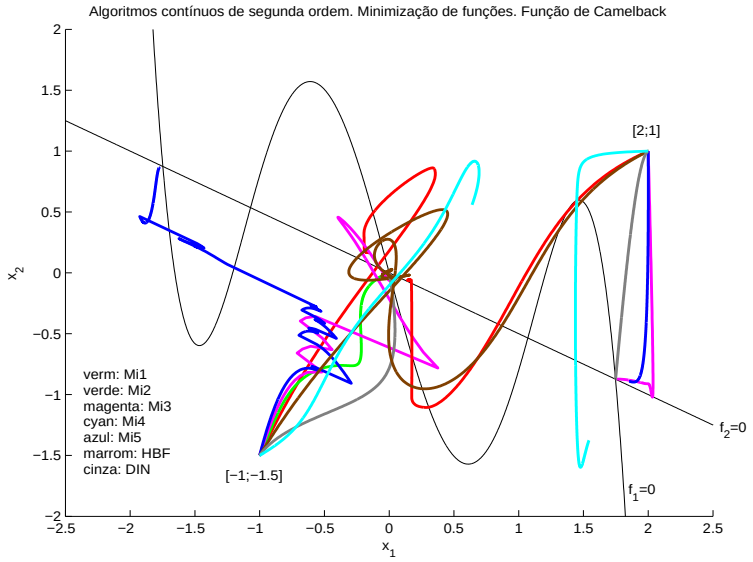

Figura 3.11: Simulações dos algoritmos contínuos para a função de Camelback invertida

A partir do ponto inicial $\mathbf{x}_0 = [-1; -1.5]^T$, todas as trajetórias convergem ao mínimo global, excetuando a gerada pelo MI5, que se desvia a um mínimo local. A trajetória gerada pelo MI4 apresenta um comportamento muito errante, sendo impossível determinar se esta convergiria a um mínimo caso a simulação tivesse continuado. As trajetórias geradas pelo MI2 e DIN, convergem de maneira direta para o mínimo global, ao tempo que as outras apresentam oscilações antes de atingir este. A partir do ponto inicial $\mathbf{x}_0 = [2\ 1]^T$, apenas as trajetórias geradas pelo MI1, MI3, MI5, HBF e DIN conseguiram convergir, e apenas o MI1 e o HBF ao mínimo global, ultrapassando o mínimo local mais próximo. A trajetória do MI4 também se mostra errante e não é possível afirmar se teria atingido um mínimo caso tivesse continuado. Um resultado interessante é que o algoritmo DIN só consegue ir para o mínimo global para valores muito baixos do ganho b, quando o algoritmo se aproxima do HBF, o que mostra que, apesar de convergir mais diretamente para um mínimo, evitando as oscilações espaciais características do HBF, o termo de amortecimento adicional do DIN anula a capacidade de ultrapassar mínimos locais, o que constitui a vantagem do HBF.

3.4.1.2 Simulações dos algoritmos de segunda ordem contínuos com a função de Rosenbrock

A função de Rosenbrock foi apresentada na equação (3.11).

As constantes escolhidas são $a = 0.5$ e $b = 1$. Com estas constantes, a função

apresenta um mínimo global em $x_1 = x_2 = 1$, que corresponde ao único zero do gradiente.

Os ganhos foram escolhidos:

MI1: $\kappa = 1.3$

MI3: $\Gamma = 5I$

MI4: $\Gamma = 0.3I$

MI5: $\Gamma = 0.1I,\ a = b = 1$

HBF: $\Gamma = I$

DIN: $a = b = 1$

Foram testados diversos pontos iniciais, mas serão mostradas as trajetórias que começam em $\mathbf{x}_0 = [0\ 3]^T$, por se mostrarem mais representativas.

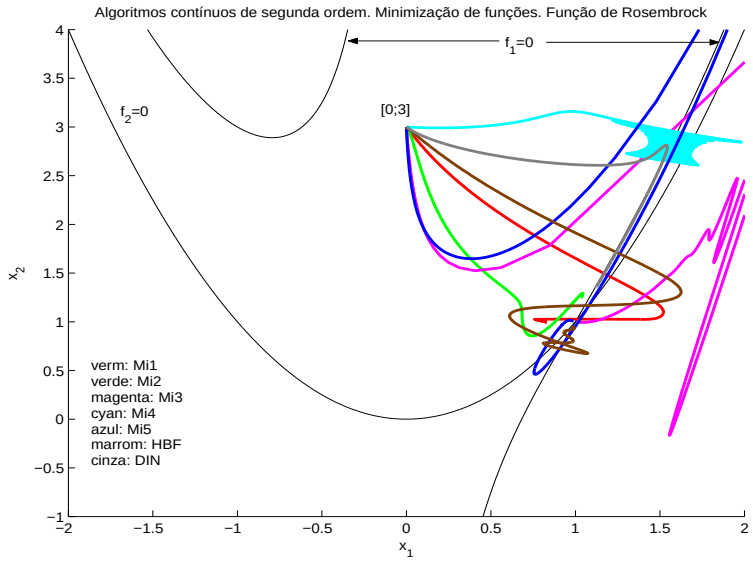

Figura 3.12: Simulações dos algoritmos contínuos para a função de Rosenbrock

Aqui, a trajetória gerada por MI4 também apresenta muitas oscilações, sendo impossível predizer se atingiria o mínimo caso a simulação tivesse continuado. As trajetórias geradas pelos outros algoritmos convergem ao mínimo. No caso da gerada pelo MI2, de maneira bastante direta, nos outros casos com algumas oscilações espaciais. Observe-se que os algoritmos MI3 e MI5, que utilizam na planta a inversa do hessiano, não estão definidos sobre o locus de $\det(\nabla^2\phi(\mathbf{x})) = x_2 - x_1^2 - 1 = 0$, onde este hessiano é singular. Porém, as trajetórias geradas por estes conseguem ultrapassar esta curva devido à discretização do algoritmo, tal como acontecia com o algoritmo de Newton. Note-se que estas trajetórias parecem acompanhar esta curva, até conseguirem

ultrapassa-la, para depois retornar, convergindo assim ao mínimo da função.

3.5 Discretização dos algoritmos contínuos de segunda ordem para achar mínimos de funções

Nesta seção, serão discretizados os algoritmos contínuos apresentados na seção anterior. Os algoritmos serão discretizados pelo método de Euler, com os comprimentos dos passos das variáveis de estado e de controle calculadas pelo método de controle ótimo de Liapunov.

Também aqui, o objetivo é que ambas variáveis atinjam os valores desejados, isto é, um mínimo da função para o caso da variável de estado, e zero para o caso da variável de controle, de maneira tal de estacionar a trajetória descrita pela variável de estado neste ponto. Para calcular os comprimentos dos passos de ambas variáveis, duas funções de Liapunov candidatas discretas serão escolhidas, e os comprimentos dos passos calculados de maneira tal de minimizar o decrescimento de ambas funções de Liapunov a cada iteração, segundo a metodologia de controle ótimo de Liapunov.

As leis de atualização de ambas variáveis, discretizando seus correspondentes contínuos por Euler, são dadas por:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \varphi_k \tag{3.40}$$

onde $\varphi_k = \dot{\mathbf{x}}$ dependerá da escolha da planta.

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \beta_k \psi_k \tag{3.41}$$

onde $\psi_k = \dot{\mathbf{u}}$ dependerá da escolha do controlador.

A variável de referência não pode ser escolhida como o valor desejado para a função, pois este valor, correspondente a um mínimo da função, é desconhecido. Embora isto foi o realizado no caso contínuo, aqui este valor será utilizado no cálculo do comprimento do passo para atualizar a variável de estado. Por tal razão, escolhemos como referência o valor ao qual se deseja que tenda o gradiente da função, neste caso zero, sendo a resposta da planta o valor deste gradiente, $\nabla\phi(\mathbf{x})$. O resíduo ou erro, assim, será a

referência menos a resposta a cada iteração, isto é

$$\mathbf{r}_k = -\nabla\phi(\mathbf{x}_k) \tag{3.42}$$

Na iteração seguinte, desenvolvendo o resíduo (3.42) por Taylor e aproximando até o termo de primeira ordem:

$$\mathbf{r}_{k+1} \simeq \mathbf{r}_k - \alpha_k \nabla^2\phi(\mathbf{x}_k)\varphi_k$$

Como função de Liapunov candidata discreta da variável de estado $\mathbf{x}_k$, será escolhida a norma quadrado do resíduo, isto é

$$V_{\mathbf{r}_k} = \mathbf{r}_k^T \mathbf{r}_k$$

$$\Rightarrow\ \Delta V_{\mathbf{r}} = V_{\mathbf{r}_{k+1}} - V_{\mathbf{r}_k} = \|\mathbf{r}_{k+1}\|^2 - \|\mathbf{r}_k\|^2 = 2\alpha_k \nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\varphi_k + \alpha_k^2 \varphi_k^T(\nabla\phi(\mathbf{x}_k))^2\varphi_k$$

e derivando com respeito a α_k e igualando a zero, segundo a metodologia LOC:

$$\frac{\partial \Delta V_{\mathbf{r}}}{\partial \alpha_k} = 2\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\varphi_k + 2\alpha_k\varphi_k^T(\nabla^2\phi(\mathbf{x}_k))^2\varphi_k = 0$$

Portanto

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\varphi_k}{\varphi_k^T(\nabla^2\phi(\mathbf{x}_k))^2\varphi_k} \tag{3.43}$$

Com este comprimento do passo

$$\Delta V_{\mathbf{r}} = -\frac{(\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\varphi_k)^2}{\varphi_k^T(\nabla^2\phi(\mathbf{x}_k))^2\varphi_k} \leq 0$$

o que garante o não crescimento desta função de Liapunov ao longo das trajetórias do sistema em malha fechada.

Observe-se que a escolha do resíduo (3.42), sendo o objetivo do sistema de controle em malha fechada diminuir a norma quadrado deste valor a cada iteração, implica que a trajetória poderia se estacionar em qualquer ponto de gradiente zero ($\mathbf{r}_k = \mathbf{0}$), inclusive em um máximo da função, o que não acontecia com o sistema contínuo. Uma possibilidade para evitar isto seria escolher como resíduo o valor da função menos um

mínimo estimado γ, de maneira tal que para todo $\mathbf{x} \in \text{dom}(\phi(\mathbf{x}))$, $\gamma < \phi(\mathbf{x})$.

Assim, $r_k = \gamma - \phi(\mathbf{x}_k) \in \mathbb{R}^1$. Como função de Liapunov candidata para a variável de estado $\mathbf{x}_k$, pode ser escolhida $V_{r_k} = r_k^2 = (\gamma - \phi(\mathbf{x}_k))^2$. A metodologia LOC teria conduzido ao seguinte comprimento do passo:

$$\alpha_k = -\frac{\phi(\mathbf{x}_k) - \gamma}{\nabla^T \phi(\mathbf{x}_k)\varphi_k}$$

e a variação da função de Liapunov candidata discreta da variável de estado seria

$$\Delta V_r = -(\gamma - \phi(\mathbf{x}_k))^2 < 0$$

Este comprimento do passo não foi testado.

No caso da variável de controle, deseja-se que esta tenda para zero, para as trajetórias se estacionarem. Por esta razão, a função de Liapunov candidata discreta para esta variável será escolhida como a norma quadrado da variável de controle, isto é:

$$V_{\mathbf{u}_k} = \mathbf{u}_k^T \mathbf{u}_k$$

$$\Rightarrow \Delta V_{\mathbf{u}} = V_{\mathbf{u}_{k+1}} - V_{\mathbf{u}_k} = \|\mathbf{u}_{k+1}\|^2 - \|\mathbf{u}_k\|^2 = 2\beta_k \mathbf{u}_k^T \psi_k + \beta_k^2 \psi_k^T \psi_k$$

e derivando com respeito a β_k e igulando a zero, segundo a metodologia LOC:

$$\frac{\partial \Delta V_{\mathbf{u}}}{\partial \beta_k} = 2\mathbf{u}_k^T \psi_k + 2\beta_k \psi_k^T \psi_k = 0$$

Portanto

$$\beta_k = -\frac{\mathbf{u}_k^T \psi_k}{\psi_k^T \psi_k} \tag{3.44}$$

Com este comprimento do passo:

$$\Delta V_{\mathbf{u}} = -\frac{(\mathbf{u}_k^T \psi_k)^2}{\psi_k^T \psi_k} \leq 0$$

o que garante o não crescimento desta função de Liapunov ao longo das trajetórias do sistema em malha fechada.

Aqui também, como no caso dos sistemas de segunda ordem para achar zeros de

funções vetoriais, pode se escrever:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \frac{\mathbf{u}_k^T \psi_k}{\psi_k^T \psi_k} \psi_k = \left(I - \frac{\psi_k \psi_k^T}{\psi_k^T \psi_k} \right) \mathbf{u}_k$$

isto é, $\mathbf{u}_{k+1}$ é a projeção de $\mathbf{u}_k$ sobre o plano normal a $\psi_\mathbf{k}$.

Observe-se que, aqui também, os comprimentos dos passos tanto para a atualização da variável de estado como para a atualização da variável de controle podem ser negativos. Entretanto, o não crescimento das respectivas funções de Liapunov discretas está garantido.

Em seguida, serão escolhidas as variáveis φ_k e ψ_k segundo as escolhas de planta e controlador para cada caso apresentado na seção anterior.

1) Primeira escolha (MI1)

$\varphi_k = \mathbf{u}_k \quad \psi_k = -\kappa \text{sgn}(\mathbf{u}_k) - \nabla\phi(\mathbf{x}_k) \quad \kappa > 0$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T \phi(\mathbf{x}_k) \nabla^2 \phi(\mathbf{x}_k) \mathbf{u}_k}{\mathbf{u}_k^T (\nabla^2 \phi(\mathbf{x}_k))^2 \mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T (\kappa \text{sgn}(\mathbf{u}_k) + \nabla\phi(\mathbf{x}_k))}{\|\kappa \text{sgn}(\mathbf{u}_k) + \nabla\phi(\mathbf{x}_k)\|^2} \tag{3.45}$$

Os cálculos dos comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, ou naqueles pontos onde $\mathbf{u}_k \in \mathcal{N}(\nabla^2\phi(\mathbf{x}_k))$, ou ainda onde $\mathbf{u}_k \in \mathcal{N}((\nabla^2\phi(\mathbf{x}_k))^2)$.

2) Segunda escolha (MI2)

$\varphi_k = \mathbf{u}_k \quad \psi_k = -(\nabla^2\phi(\mathbf{x}_k))^2 \mathbf{u}_k - \nabla\phi(\mathbf{x}_k)$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T \phi(\mathbf{x}_k) \nabla^2 \phi(\mathbf{x}_k) \mathbf{u}_k}{\mathbf{u}_k^T (\nabla^2 \phi(\mathbf{x}_k))^2 \mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T \left[(\nabla^2 \phi(\mathbf{x}_k))^2 \mathbf{u}_k + \nabla\phi(\mathbf{x}_k) \right]}{\|(\nabla^2 \phi(\mathbf{x}_k))^2 \mathbf{u}_k + \nabla\phi(\mathbf{x}_k)\|^2} \tag{3.46}$$

Também aqui, os cálculos dos comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, ou naqueles pontos onde $\mathbf{u}_k \in \mathcal{N}(\nabla^2\phi(\mathbf{x}_k))$, ou ainda onde $\mathbf{u}_k \in \mathcal{N}((\nabla^2\phi(\mathbf{x}_k))^2)$, e, no caso da variável de controle, se $(\nabla^2\phi(\mathbf{x}_k))^2 \mathbf{u}_k = -\nabla\phi(\mathbf{x}_k)$.

3) Terceira escolha (MI3)

$\varphi_k = \nabla^{-2}\phi(\mathbf{x}_k) \Gamma \mathbf{u}_k \quad \psi_k = -\Gamma(\mathbf{u}_k + \nabla^{-2}\phi(\mathbf{x}_k) \nabla\phi(\mathbf{x}_k)) \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\Gamma\mathbf{u}_k}{\mathbf{u}_k^T\Gamma^2\mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T\Gamma(\mathbf{u}_k + \nabla^{-2}\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))}{\|\Gamma(\mathbf{u}_k + \nabla^{-2}\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))\|^2} \tag{3.47}$$

Aqui, os comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, ou se $\mathbf{u}_k = -\nabla^{-2}\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)$.

4) Quarta escolha (MI4)

$\varphi_k = \nabla^2\phi(\mathbf{x}_k)\Gamma\mathbf{u}_k \quad \psi_k = -\Gamma(\mathbf{u}_k + \nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k) \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)(\nabla^2\phi(\mathbf{x}_k))^2\Gamma\mathbf{u}_k}{\mathbf{u}_k^T\Gamma(\nabla^2\phi(\mathbf{x}_k))^4\Gamma\mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T\Gamma(\mathbf{u}_k + \nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))}{\|\Gamma(\mathbf{u}_k + \nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k))\|^2} \tag{3.48}$$

Aqui, os comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, se $\mathbf{u}_k \in \mathcal{N}(\nabla^n\phi(\mathbf{x}_k)\Gamma)$ para $n \in \{1, .., 4\}$, ou ainda se $\mathbf{u}_k = -\nabla^2\phi(\mathbf{x}_k)\nabla\phi(\mathbf{x}_k)$.

5) Quinta escolha (MI5)

$\varphi_k = \nabla^{-2}\phi(\mathbf{x}_k)\mathbf{u}_k \quad \psi_k = -\Gamma\nabla\phi(\mathbf{x}_k) - (I-\Gamma)\mathbf{u}_k - \nabla^{-2}\phi(\mathbf{x}_k)(a\mathbf{u}_k + b\nabla\phi(\mathbf{x}_k)) \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\mathbf{u}_k}{\mathbf{u}_k^T\mathbf{u}_k} \quad \beta_k = -\frac{\mathbf{u}_k^T\left[-(\Gamma + b\nabla^{-2}\phi(\mathbf{x}_k))\nabla\phi(\mathbf{x}_k) - (I - \Gamma + a\nabla^{-2}\phi(\mathbf{x}_k))\mathbf{u}_k\right]}{\|(\Gamma + b\nabla^{-2}\phi(\mathbf{x}_k))\nabla\phi(\mathbf{x}_k) + (I - \Gamma + a\nabla^{-2}\phi(\mathbf{x}_k))\mathbf{u}_k\|^2} \tag{3.49}$$

Aqui, se $\mathbf{u}_k = \mathbf{0}$, ou se $(\Gamma + b\nabla^{-2}\phi(\mathbf{x}_k))\nabla\phi(\mathbf{x}_k) = -(I - \Gamma + a\nabla^{-2}\phi(\mathbf{x}_k))\mathbf{u}_k$, então os comprimentos dos passos se fazem indeterminados.

6) Sexta escolha (HBF)

$\varphi_k = \mathbf{u}_k \quad \psi_k = -\nabla\phi(\mathbf{x}_k) - \Gamma\mathbf{u}_k \quad \Gamma = \Gamma^T \succ 0$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi(\mathbf{x}_k))^2\mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T(\nabla\phi(\mathbf{x}_k) + \Gamma\mathbf{u}_k)}{\|\nabla\phi(\mathbf{x}_k) + \Gamma\mathbf{u}_k\|^2} \tag{3.50}$$

Aqui, os comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, se $\mathbf{u}_k \in$

$\mathcal{N}(\nabla^2\phi(\mathbf{x}_k))$, se $\mathbf{u}_k \in \mathcal{N}((\nabla^2\phi(\mathbf{x}_k))^2)$, ou se $\nabla\phi(\mathbf{x}_k) = -\Gamma\mathbf{u}_k$.

7) Sétima escolha (DIN)

$$\varphi_k = \mathbf{u}_k \quad \psi_k = -\nabla\phi(\mathbf{x}_k) - a\mathbf{u}_k - b\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k$$

E substituindo em (3.43) e (3.44):

$$\alpha_k = -\frac{\nabla^T\phi(\mathbf{x}_k)\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi(\mathbf{x}_k))^2\mathbf{u}_k} \quad \beta_k = \frac{\mathbf{u}_k^T(\nabla\phi(\mathbf{x}_k) + a\mathbf{u}_k + b\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k)}{\|\nabla\phi(\mathbf{x}_k) + a\mathbf{u}_k + b\nabla^2\phi(\mathbf{x}_k)\mathbf{u}_k\|^2} \tag{3.51}$$

Aqui, o cálculo dos comprimentos dos passos se fazem indeterminados se $\mathbf{u}_k = \mathbf{0}$, se $\mathbf{u}_k \in \mathcal{N}(\nabla^2\phi(\mathbf{x}_k))$, se $\mathbf{u}_k \in \mathcal{N}((\nabla^2\phi(\mathbf{x}_k))^2)$, ou se $(aI + b\nabla^2\phi(\mathbf{x}_k))\mathbf{u}_k = -\nabla\phi(\mathbf{x}_k)$.

Os algoritmos discretos serão apresentados resumidamente na seguinte tabela.

nome	α_k	β_k
MI1	$-\frac{\nabla^T\phi\nabla^2\phi\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi)^2\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T(\kappa\mathrm{sgn}(\mathbf{u}_k)+\nabla\phi)}{\|\kappa\mathrm{sgn}(\mathbf{u}_k)+\nabla\phi\|^2}$
MI2	$-\frac{\nabla^T\phi\nabla^2\phi\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi)^2\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T\left[(\nabla^2\phi)^2\mathbf{u}_k+\nabla\phi\right]}{\|(\nabla^2\phi)^2\mathbf{u}_k+\nabla\phi\|^2}$
MI3	$-\frac{\nabla^T\phi\Gamma\mathbf{u}_k}{\mathbf{u}_k^T\Gamma^2\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T\Gamma(\mathbf{u}_k+\nabla^{-2}\phi\nabla\phi)}{\|\Gamma(\mathbf{u}_k+\nabla^{-2}\phi\nabla\phi)\|^2}$
MI4	$-\frac{\nabla^T\phi(\nabla^2\phi)^2\Gamma\mathbf{u}_k}{\mathbf{u}_k^T\Gamma(\nabla^2\phi)^4\Gamma\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T\Gamma(\mathbf{u}_k+\nabla^2\phi\nabla\phi)}{\|\Gamma(\mathbf{u}_k+\nabla^2\phi\nabla\phi)\|^2}$
MI5	$-\frac{\nabla^T\phi\mathbf{u}_k}{\mathbf{u}_k^T\mathbf{u}_k}$	$-\frac{\mathbf{u}_k^T\left[-\Gamma\nabla\phi-(I-\Gamma)\mathbf{u}_k-\nabla^{-2}\phi(a\mathbf{u}_k+b\nabla\phi)\right]}{\|\Gamma\nabla\phi+(I-\Gamma)\mathbf{u}_k+\nabla^{-2}\phi(a\mathbf{u}_k+b\nabla\phi)\|^2}$
HBF	$-\frac{\nabla^T\phi\nabla^2\phi\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi)^2\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T(\nabla\phi+\Gamma\mathbf{u}_k)}{\|\nabla\phi+\Gamma\mathbf{u}_k\|^2}$
DIN	$-\frac{\nabla^T\phi\nabla^2\phi\mathbf{u}_k}{\mathbf{u}_k^T(\nabla^2\phi)^2\mathbf{u}_k}$	$\frac{\mathbf{u}_k^T(\nabla\phi+a\mathbf{u}_k+b\nabla^2\phi\mathbf{u}_k)}{\|\nabla\phi+a\mathbf{u}_k+b\nabla^2\phi\mathbf{u}_k\|^2}$

Tabela 3.5 Algoritmos discretos de segunda ordem para achar mínimos de funções deduzidos com a metodologia CLF/LOC onde $\nabla\phi = \nabla\phi(\mathbf{x}_k)$ e $\nabla^2\phi = \nabla^2\phi(\mathbf{x}_k)$

3.5.1 Simulações dos algoritmos de segunda ordem discretos para achar mínimos de funções escalares

Os algoritmos discretos apresentados na seção anterior foram testados com funções escalares de duas variáveis, a fim das trajetórias poderem ser graficadas.

As funções escolhidas foram a de Rosenbrock (3.11) e a de Camelback invertida (3.8). Foi adotado como critério de parada a iteração tal que $\|\nabla\phi(\mathbf{x}_k)\| < 0.01$, e foi considerada a trajetória divergente se o algoritmo não conseguiu atingir um mínimo em até 500 iterações.

Os valores iniciais da variável de controle $\mathbf{u}_0$ foi diferente para cada algoritmo, a fim de não indeterminar os cálculos dos comprimentos dos passos e de orientar a trajetória inicial de uma maneira conveniente. Este valor inicial foi $\mathbf{u}_0 = -\nabla^2\phi(\mathbf{x}_0)\nabla\phi(\mathbf{x}_0)$ para todos os algoritmos excetuando o MI4 (onde este valor zeraria o denominador de β_0), para o qual foi escolhido $\mathbf{u}_0 = -\Gamma^{-1}\nabla\phi(\mathbf{x}_0)$.

Os ganhos foram variáveis também para cada experiência, a fim de sintonizar os algoritmos de maneira tal de atingir um mínimo no menor número de iterações possível.

Os tempos de duração de cada experiência foram testados também, mas esses resultados foram inconclusivos pois diferentes execuções do mesmo algoritmo e a partir do mesmo ponto inicial demoraram diferentes durações, e sempre na ordem das décimas de segundo.

3.5.1.1 Simulações com a função de Camelback invertida

A função de Camelback (3.8) será utilizada com as mesmas constantes que no caso contínuo, e também invertida a fim dos máximos virarem mínimos. Os pontos iniciais testados foram $\mathbf{x}_0 = [2\ 1]^T$, $\mathbf{x}_0 = [-1\ -1.5]^T$ e $\mathbf{x}_0 = [0\ 1.5]^T$.

Os ganhos foram escolhidos:

MI1: $\kappa = 10$

MI3: $\Gamma = I$

MI4: $\Gamma = 0.01I$, exceto a partir do ponto inicial $\mathbf{x}_0 = [-1\ -1.5]^T$, onde foi escolhido $\Gamma = 0.001I$

MI5: $\Gamma = I$ $a = b = 10$, exceto do ponto inicial $\mathbf{x}_0 = [0\ 1.5]^T$, onde $a = b = 1$, mesmo Γ

HBF: $\Gamma = I$

DIN: $a = b = 1$

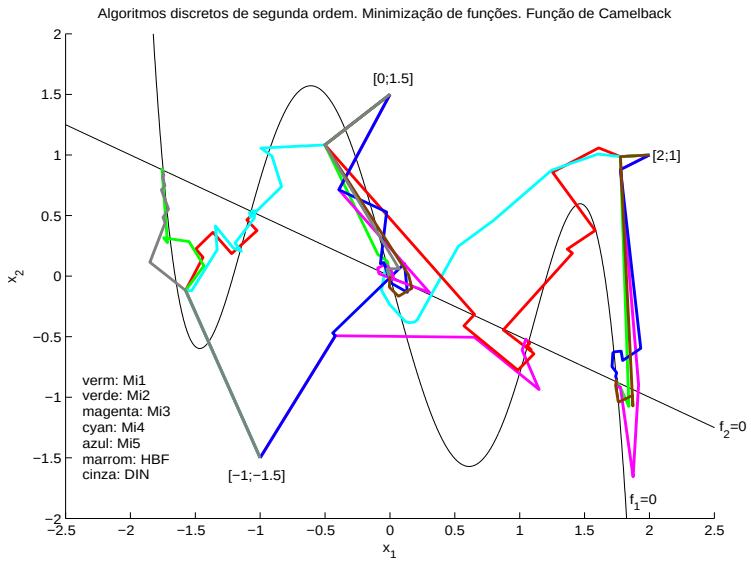

Figura 3.13: Simulações dos algoritmos discretos para a função de Camelback invertida

Observa-se aqui que todos os algoritmos conseguiram atingir um ponto de gradiente zero da função, exceto o HBF a partir do ponto inicial $\mathbf{x}_0 = [-1\ -1.5]^T$ e o DIN a partir do ponto inicial $\mathbf{x}_0 = [2\ 1]^T$. Note-se que, como já foi apontado, o objetivo de atingir pontos de gradiente zero pode implicar que as trajetórias se estacionem em selas, o que não acontecia com os sistemas contínuos, onde as trajetórias sempre convergiam a mínimos, mesmo que sejam locais.

3.5.1.2 Simulações com a função de Rosenbrock

A função de Rosenbrock (3.11), será utilizada com as mesmas constantes, com as quais apresenta um mínimo único em $\mathbf{x}^* = [1\ 1]^T$.

Os pontos iniciais testados foram $\mathbf{x}_0 = [-0.5\ 2]^T$ e $\mathbf{x}_0 = [1.5\ 0]^T$.

Os ganhos foram escolhidos:

MI1: $\kappa = 100$

MI3: $\Gamma = I$

MI4: $\Gamma = 0.01I$

MI5: $\Gamma = 10I$ $a = b = 1$, exceto do ponto inicial $\mathbf{x}_0 = [1.5\ 0]^T$, onde $\Gamma = I$, mesmos a e b

HBF: $\Gamma = I$

DIN: $a = b = 1$

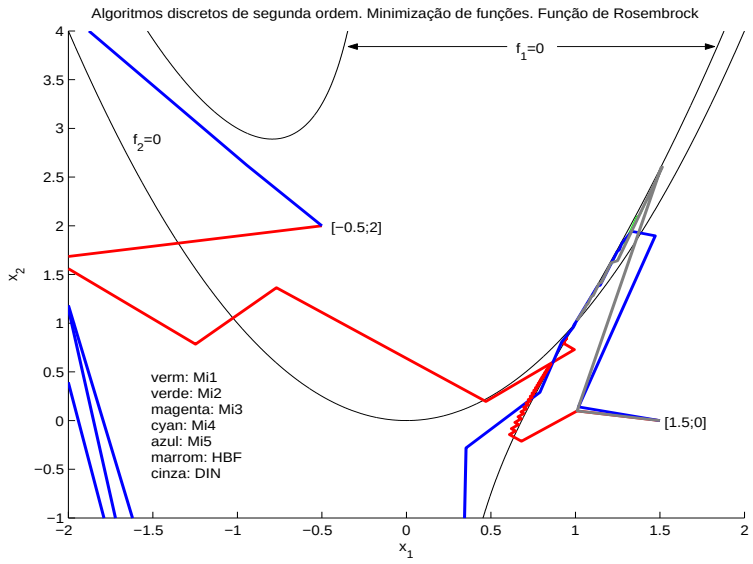

Figura 3.14: Simulações dos algoritmos discretos para a função de Rosenbrock

Observa-se aqui que os algoritmos apresentaram mais dificuldades para atingir o mínimo da função, devido à forma da função de Rosenbrock. A partir do ponto inicial $\mathbf{x}_0 = [-0.5\ 2]^T$, apenas o MI1 e o MI5 conseguiram convergir. A partir do ponto inicial $\mathbf{x}_0 = [1.5\ 0]^T$, mais próximo do mínimo, MI1, MI2, MI5 e DIN conseguiram convergir.

A seguir, serão apresentados os números de iterações dos algoritmos para ambas experiências na seguinte tabela.

função	$\mathbf{x}_0$	MI1	MI2	MI3	MI4	MI5	HBF	DIN
Camelback	$[2\ 1]^T$	14	6	9	18	13	11	-
	$[-1\ -1.5]^T$	12	8	7	10	7	-	13
	$[0\ 1.5]^T$	10	5	27	9	6	8	6
Rosenbrock	$[-0.5\ 2]^T$	49	-	-	-	12	-	-
	$[1.5\ 0]^T$	56	19	-	-	10	-	22

Tabela 3.6: Número de iterações realizadas pelos algoritmos discretos
onde - indica que o algoritmo não atingiu um mínimo em até 500 iterações

3.6 Conclusões deste capítulo

Os algoritmos de segunda ordem apresentados se mostraram capazes de atingir zeros de funções vetoriais e mínimos de funções escalares de maneira satisfatória.

Com respeito aos algoritmos para achar zeros de funções vetoriais, em comparação com os algoritmos de primeira ordem desenvolvidos no capítulo 2, notamos que aqui as excursões (número de iterações para os casos discretos), são maiores. Porém, para funções complexas, notadamente para o caso da função de Branin com $c = 1$, parecem ter mais facilidade para convergir a um zero. Também notamos que são mais sensíveis aos ganhos, o qual exige uma sintonização mais cuidadosa para atingir o objetivo.

Com respeito aos algoritmos para achar mínimos de funções escalares, observamos que a pretensão de ultrapassar mínimos locais para as trajetórias se estacionarem em um mínimo global, só é atingida em parte por alguns algoritmos e depois de uma cuidadosa sintonização dos ganhos. Porém, esta possibilidade representa uma vantagem evidente com respeito aos algoritmos equivalentes de primeira ordem, que sempre se estacionam em um mínimo que pode ser local caso a função objetivo seja não convexa. Aqui também notamos que as excursões parecem ser maiores que no caso das trajetórias geradas pelos algoritmos de primeira ordem.

No caso discreto, uma desvantagem que apresentam estos algoritmos é a possibilidade das trajetórias se estacionarem em qualquer ponto de gradiente zero. A escolha alternativa do resíduo, estimando um valor γ menor que o mínimo global, exigiria um critério de parada diferente de uma norma mínima do resíduo, pois evidentemente no seria atingido um ponto de resíduo zero.

Conclue-se assim que algoritmos de segunda ordem podem apresentar vantagens desejáveis, razão pela qual a pesquisa e desenvolvimento destes está justificada.

Capítulo 4

Algoritmo para otimização convexa baseado em sistema dinâmico gradiente projetado utilizando função de Liapunov de controle

4.1 Introdução

Otimização convexa é uma classe especial de problemas matemáticos de otimização, a qual inclui mínimos quadrados e problemas de programação linear. Para resolver estes problemas em particular, foi desenvolvida uma grande quantidade de teoria, e se utilizam em uma ampla variedade de aplicações diversas. A matemática de otimização convexa é estudada desde um século atrás, porém, várias descobertas recentes estimularam um novo interesse na área. A primeira é a descoberta que métodos de pontos interiores, desenvolvidos a partir de 1984 para a resolução de problemas de programação linear, podendo ser utilizados para resolver problemas de otimização convexa também. Estes novos métodos nos permitem resolver certas novas classes de problemas de otimização convexa, tais como programação semi-definida e programação em cones de segunda ordem, tão facilmente como programas lineares.

A segunda descoberta é que problemas de otimização convexa em geral, além dos casos particulares de programação linear e métodos de quadrados mínimos, são mais

comuns na prática do que se acreditava. Desde 1990 muitas aplicações foram descobertas em áreas tais como controle de sistemas automáticos, estimação e processamento de sinais, comunicações e redes, projeto de circuitos eletrônicos, análise de dados e modelagem, estatística e finanças. Otimização convexa encontrou também um amplo campo de aplicação em otimização combinatorial e otimização global, onde é utilizada para encontrar limites do valor ótimo, assim como soluções aproximadas.

Existem diversas vantagens em reconhecer ou formular um problema como um problema de otimização convexa, entre eles a existência de poderosos métodos de resolução, tanto contínuos quanto discretos, confiáveis e eficientes, que podem ser implementados em computadores ou em circuitos analógicos para resolução em tempo real ([17, prefácio]).

A metodologia de utilizar sistemas dinâmicos do tipo gradiente para resolver problemas de otimização tem uma longa história. Porém, a implementação desta à luz da teoria de controle é mais recente (ver [78]). Por exemplo, diversos algoritmos destinados a otimizar funções, tanto irrestritas como restritas a um domínio convexo, utilizando funções de controle de Liapunov (CLF) foram desenvolvidos e amplamente analisados em [10], [11] e [13].

O problema de achar o mínimo de uma função convexa C^1 irrestrita é equivalente a achar zeros do seu gradiente, e existem diversos algoritmos bem conhecidos destinados a tal fim. Algoritmos destinados a achar mínimos de funções convexas na presença de restrições convexas também são conhecidos. Por exemplo, em [54] os autores classificam diversos algoritmos apresentados em diferentes publicações destinados a achar mínimos de funções com e sem restrições. Os autores analisam apenas algoritmos contínuos e os dividem em duas classes: aqueles que apresentam um enfoque baseado em sistemas dinâmicos (ou ODE), e aqueles baseados em redes neurais. Os autores também propõem conciliar as vantagens de ambas aproximações em um esquema denominado otimização neurodinâmica. Nesse trabalho, embora seja apontada a inexistência de uma classificação consistente dos algoritmos existentes, não é abordada uma maneira genérica de derivação destes, como sim é feito em [13] utilizando a teoria de controle.

Para a resolução de problemas de otimização convexa, a utilização de sistemas de controle do tipo gradiente a estrutura variável, que apresentam trajetórias em modos deslizantes ao longo da fronteira do conjunto viável, também não é nova; ver, por

exemplo, o interessante trabalho de Zak ([86, s. 6.9]), assim como [32] e [27] para o caso de programação linear e quadrática.

Dentre os algoritmos apresentados na bibliografia, podemos mencionar como principais inconvenientes encontrados:

1) A utilização da projeção ortogonal, a qual é de difícil avaliação exceto se a fronteira do conjunto viável tiver uma forma simples (por exemplo, uma caixa ou uma esfera [48, p. 41]). Esta projeção é utilizada em [48], [67], [3], [54], [20][1].

2) A exigência do ponto inicial estar dentro do conjunto viável, o qual, dependendo do problema, pode ser de cara resolução do ponto de vista computacional ([48], [3], [67]).

3) Limitações com respeito à função objetivo ou às restrições. Por exemplo considerar apenas restrições de desigualdade (como [53] e [32]; [86] considera apenas restrições de desigualdade afins, [6] exemplifica apenas sobre o ortante não negativo, [15] e [47] consideram apenas limites superior e inferior das variáveis, assim como [63], embora estes últimos autores mencionam a possibilidade de estender a utilização do algoritmo apresentado a problemas convexos em geral) ou apenas de igualdade ([9]); ou considerar como objetivo funções quadráticas ([67]); ou considerar funções objetivos estritamente convexas ([5]). Em geral, as condições exigidas à função objetivo se estendem a todo o domínio da função, e não apenas ao conjunto viável (como [78]).

4) No caso dos algoritmos contínuos, não são apresentadas a discretização destes ([54], [86], [3], [5], [53], [32], [9], [20]); uma exceção é encontrada em [6].

5) No caso dos algoritmos discretos, a escolha do comprimento de passo é, em muitos casos, justificada experimentalmente, sem qualquer prova da sua qualidade e vantagens com respeito a outras escolhas ([48] utiliza a regra de Armijo sem mencionar a vantagem deste critério com respeito a outros existentes, [67] propõe a alternância entre dois critérios, sem provar a otimalidade destes nem do momento de alternância entre um e outro, [47] propõe o mais clássico secionamento do domínio ou biseção).

6) Os critérios de parada nos algoritmos discretos em muitos casos podem ser caros do ponto de vista computacional. Por exemplo conferir as condições KKT (alguns dos

[1]Embora nesta bibliografia não é utilizada explicitamente uma projeção ortogonal, exige-se que uma função $\Phi_0(\mathbf{x})$ que faz parte da ODE que representa a dinâmica do sistema, seja convexa com $\text{argmin}\Phi_0(\mathbf{x}) = \{\mathbf{x}$ tal que $\mathbf{x}$ esteja no conjunto viável$\}$. Em diversas situações esta função só pode ser calculada como função da distância entre o ponto $\mathbf{x}$ e sua projeção ortogonal sobre este conjunto.

algoritmos apresentados em [67]), ou conferir se o ponto achado em cada iteração é estacionário ([48]). [15] apenas limita o número de iterações.

Apresenta-se aqui um novo algoritmo para resolver problemas de otimização convexa baseado em um sistema de controle a estrutura variável, cuja lei de atualização das variáveis é projetada utilizando funções de controle de Liapunov (CLF), o qual não mostra os inconvenientes apontados. Este é discretizado calculando o comprimento de passo ótimo segundo a teoria de controle ótimo de Liapunov (LOC), resultando em um algoritmo iterativo simples de implementar independentemente da dificuldade do problema. Esta abordagem contrasta com aquela baseada na escolha do comprimento do passo de iteração por secionamento do domínio, cuja dificuldade é proporcional à dimensão do problema (número de variáveis, número de restrições, etc.). Uma versão preliminar dos resultados aqui apresentados foram publicados em [61].

4.2 Preliminares

Serão apresentadas nesta seção diversas definições e lemas necessários para o desenvolvimento do trabalho.

Definição 4.2.1 *Conjunto convexo ([17, p. 23])*

Um conjunto C é convexo se para todo $\mathbf{x}_1,\ \mathbf{x}_2 \in C$ e para todo escalar θ tal que $0 \leq \theta \leq 1$:

$$\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in C \tag{4.1}$$

Definição 4.2.2 *Função convexa ([17, p. 67])*

Uma função $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ é convexa se para todo $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom} f$ e para todo escalar θ tal que $0 \leq \theta \leq 1$

$$f(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2) \tag{4.2}$$

A função é estritamente convexa se a desigualdade é estrita para todo $\mathbf{x}_1 \neq \mathbf{x}_2$ e $0 < \theta < 1$.

Definição 4.2.3 *Função afim ([17, p. 67])*

Uma função é afim se a igualdade se mantém em (4.2) para todo $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom} f$ *e para todo* $\theta \in [0;1]$.

Lema 4.2.4 *Condição de primeira ordem das funções convexas.*

Uma função diferenciável $f : \mathbb{R}^n \to \mathbb{R}$ *é convexa se e somente se*

$$\forall \mathbf{x},\ \mathbf{y} \in \mathrm{dom} f \quad \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \leq f(\mathbf{y}) - f(\mathbf{x}) \tag{4.3}$$

A função e esritamente convexa se a desigualdade for estrita para todo $\mathbf{x} \neq \mathbf{y}$.

A função é afim se vale a igualdade em (4.3) (ver prova em [17, p. 70]).

Observe-se que, nas funções convexas,

$$\begin{aligned} \forall \mathbf{x},\ \mathbf{x}_1,\ \mathbf{x}_2 \in \mathrm{dom} f \qquad & \nabla^T f(\mathbf{x}_1)(\mathbf{x} - \mathbf{x}_1) \leq f(\mathbf{x}) - f(\mathbf{x}_1) \\ & \nabla^T f(\mathbf{x}_2)(\mathbf{x} - \mathbf{x}_2) \leq f(\mathbf{x}) - f(\mathbf{x}_2) \end{aligned}$$

Substituindo $\mathbf{x}$ por $\mathbf{x}_2$ na primeira desigualdade e $\mathbf{x}$ por $\mathbf{x}_1$ na segunda desigualdade e somando ambas, obtemos

$$\left[\nabla^T f(\mathbf{x}_1) - \nabla^T f(\mathbf{x}_2)\right](\mathbf{x}_1 - \mathbf{x}_2) \geq 0 \tag{4.4}$$

No caso da função ser estritamente convexa a desigualdade é estrita. No caso de ser afim, vale a igualdade em (4.4).

Lema 4.2.5 *Condição de segunda ordem das funções convexas.*

Uma função $f : \mathbb{R}^n \to \mathbb{R}$*, duas vezes diferenciável é convexa se e somente se*

$$\nabla^2 f(\mathbf{x}) \succeq 0 \tag{4.5}$$

A função é estritamente convexa se seu hessiano for positivo definido (ver prova em [17, p. 71]).

Definição 4.2.6 *Conjunto de nível de uma função ([17, p. 75]).*

O conjunto de nível α *de uma função* $f : \mathbb{R}^n \to \mathbb{R}$ *é definido como*

$$S_\alpha = \{\mathbf{x} \in \mathrm{dom} f \mid f(\mathbf{x}) \leq \alpha\} \tag{4.6}$$

Definição 4.2.7 *Funções quase-convexas ([17, p. 95]).*

Uma função $f : \mathbb{R}^n \to \mathbb{R}$ *é quase-convexa se seu domínio e todos os conjuntos de nível* S_α*, para todo* $\alpha \in \mathbb{R}$*, forem convexos.*

Esta definição implica que uma função, $f : \mathbb{R}^n \to \mathbb{R}$ *diferenciável é quase-convexa se e somente se ([20])*

$$\forall \mathbf{x},\, \mathbf{y} \in \mathrm{dom} f : \quad f(\mathbf{y}) \leq f(\mathbf{x}) \;\Rightarrow\; \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \leq 0 \tag{4.7}$$

A função $f : \mathbb{R}^n \to \mathbb{R}$ *diferenciável é estritamente quase-convexa se e somente se*

$$\forall \mathbf{x},\, \mathbf{y} \in \mathrm{dom} f : \quad f(\mathbf{y}) < f(\mathbf{x}) \;\Rightarrow\; \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) < 0 \tag{4.8}$$

Definição 4.2.8 *Funções pseudoconvexas ([46]).*

Uma função diferenciável $f : \mathbb{R}^n \to \mathbb{R}$ *é pseudoconvexa se para todo* $\mathbf{x},\ \mathbf{y} \in \mathrm{dom} f$ *e* $\mathbf{x} \neq \mathbf{y}$

$$\nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \geq 0 \;\Rightarrow\; f(\mathbf{y}) \geq f(\mathbf{x}) \tag{4.9}$$

A função f é estritamente pseudoconvexa se

$$\nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \geq 0 \;\Rightarrow\; f(\mathbf{y}) > f(\mathbf{x}) \tag{4.10}$$

e é fortemente pseudoconvexa se existe $\beta > 0$ *tal que*

$$\nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \geq 0 \;\Rightarrow\; f(\mathbf{y}) - f(\mathbf{x}) \geq \beta \|\mathbf{y} - \mathbf{x}\|^2 \tag{4.11}$$

Definição 4.2.9 *Projeção ortogonal sobre um conjunto convexo.*

Define-se como projeção ortogonal de um ponto $\mathbf{x}$ *sobre um conjunto fechado e convexo* Ω *a um ponto* $\mathbf{u} = \mathrm{Pr}_\Omega[\mathbf{x}] \in \Omega$ *tal que*

$$\mathbf{u} = \arg\min\{\|\mathbf{x} - \mathbf{u}\|_2,\ \forall \mathbf{u} \in \Omega\} \tag{4.12}$$

Note-se que se $\mathbf{x} \in \Omega$*, então* $\mathbf{x} = \mathrm{Pr}_\Omega[\mathbf{x}]$

4.3 Descrição do problema

Seguiremos a abordagem e nomenclatura utilizadas em [17].

Dado o problema de

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{f}(\mathbf{x}) \prec 0 \\ & \mathbf{h}(\mathbf{x}) = 0 \end{aligned} \tag{4.13}$$

onde $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})...f_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ e $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x})...h_p(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^p$ tal que para todo $i \in \{0, .., m\}$, $f_i(\mathbf{x})$ é convexa, contínua e com derivadas parciais contínuas, e para todo $i \in \{1, .., p\}$, $h_i(\mathbf{x})$ é afim, contínua e com derivadas parciais contínuas.

Define-se o conjunto viável

$$\chi = \{\mathbf{x} \,|\, \mathbf{f}(\mathbf{x}) \prec \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \tag{4.14}$$

O ponto $\mathbf{x}$ que minimiza a função dentro do conjunto viável será chamado de ponto ótimo $\mathbf{x}^*$, e o valor da função objetivo nesse ponto $p^* = f_0(\mathbf{x}^*)$.

Lema 4.3.1 *Seja* $\chi = \{\mathbf{x} \,|\, f_i(\mathbf{x}) \leq 0 \,\forall i \in \{1, .., m\},\ h_i(\mathbf{x}) = 0 \,\forall i \in \{1, .., p\}\}$. *Para toda* $f_i(\mathbf{x})$ *convexa,* $h_i(\mathbf{x})$ *afim, o conjunto* χ *é convexo.*

Prova:

Sejam $\mathbf{x}_1,\ \mathbf{x}_2 \in \chi :\ f_i(\mathbf{x}_1) \leq 0,\ f_i(\mathbf{x}_2) \leq 0,\ h_i(\mathbf{x}_1) = 0,\ h_i(\mathbf{x}_2) = 0$

$$\begin{aligned} \forall\, \theta \in [0, 1] :\quad & f_i(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f_i(\mathbf{x}_1) + (1-\theta) f_i(\mathbf{x}_2) \leq 0 \\ & h_i(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) = \theta h_i(\mathbf{x}_1) + (1-\theta) h_i(\mathbf{x}_2) = 0 \end{aligned}$$

Por definição de função convexa e função afim. Portanto $\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \chi$ *e* χ *é convexo.*

O ponto ótimo $\mathbf{x}^*$ observa a seguinte condição ([17, p. 139], [26, p. 13]):

$$\forall \mathbf{x} \in \chi \quad \nabla f_0^T(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0 \tag{4.15}$$

O problema (4.13) pode ter mais de uma solução. A continuação será estabelecida uma condição suficiente para garantir a unicidade da solução.

Suponha-se a existência de duas soluções $\mathbf{x}_1^*$ e $\mathbf{x}_2^*$. Portanto, de (4.15):

$$\begin{aligned} \forall \mathbf{x} \in \chi, \qquad & \nabla^T f_0(\mathbf{x}_1^*)(\mathbf{x} - \mathbf{x}_1^*) \geq 0 \\ & \nabla^T f_0(\mathbf{x}_2^*)(\mathbf{x} - \mathbf{x}_2^*) \geq 0 \end{aligned}$$

Considerando $\mathbf{x} = \mathbf{x}_2^*$ na primeira desigualdade e $\mathbf{x} = \mathbf{x}_1^*$ na segunda desigualdade e somando ambas, obtemos:

$$\left[\nabla^T f_0(\mathbf{x}_1^*) - \nabla^T f_0(\mathbf{x}_2^*)\right](\mathbf{x}_1^* - \mathbf{x}_2^*) \leq 0 \tag{4.16}$$

Portanto, convexidade estrita como mostrado na equação (4.4) para o caso da desigualdade estrita, é suficiente para garantir unicidade da solução.

No caso de existir mais de uma solução, o conjunto de pontos ótimos será denominado χ^*.

4.4 Função de energia

Os sistemas dinâmicos do tipo gradiente descendente mostram-se adequados na implementação de algoritmos para achar mínimos de funções irrestritas. Diante da presença de restrições, existe a estratégia de minimizar uma função irrestrita obtida a partir da função objetivo acrescentando termos de penalização, os quais incrementam seu valor quando alguma restrição não for observada ([56]). Esta função recebe o nome de função de "energia". Assim, procura-se que o mínimo da função de energia não restrita seja igual ao mínimo da função objetivo dentro do conjunto viável, e pode-se portanto utilizar um sistema dinâmico do tipo gradiente descendente a fim de achar o mínimo desta. Isto é:

$$\min_{\mathbf{x} \in \chi} E(\mathbf{x}) = \min_{\mathbf{x} \in \chi} f_0(\mathbf{x}) = f_0(\mathbf{x}^*)$$

Em [5] é mencionada uma função de energia utilizado um termo exponencial de penalização, o qual incrementa seu valor fora da região viável. Em [13] é apresentada

a seguinte função de energia:

$$E(\mathbf{x}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} f_i(\mathbf{x})\,\lambda_i\,\mathrm{uhsgn}(f_i(\mathbf{x})) + \sum_{i=1}^{p} h_i(\mathbf{x})\,\nu_i\,\mathrm{sgn}(h_i(\mathbf{x})) \tag{4.17}$$

onde $\lambda_i > 0$ para todo $i \in \{1, .., m\}$ e $\nu_i > 0$ para todo $i \in \{1, .., p\}$ e

$$\mathrm{uhsgn}(u) := \begin{cases} 1 & \forall u > 0 \\ 0 & \forall u < 0 \end{cases} \tag{4.18}$$

Os dois últimos termos em (4.17), constituem a penalização da função, e são chamados de penalização exata porque claramente se anulam para todo $\mathbf{x} \in \chi$, fazendo com que a função penalizada coincida exatamente com a função objetivo dentro do conjunto viável.

Em [13, p. 148], é mencionado que se λ_i e ν_i forem maiores que um determinado valor mínimo, o mínimo da função irrestrita $E(\mathbf{x})$ é igual ao mínimo da função restrita $f_0(\mathbf{x})$ dentro do conjunto viável. Este limite inferior de λ_i e ν_i é calculado em [78]. Porém, este cálculo exige estimativas do valor mínimo de vários gradientes, o que torna difícil a verificação das condições de convergência baseadas na função (4.17)[2].

Em [86] é considerado o problema de otimização de uma função convexa restringida por condições de desigualdade afins, apresentando a seguinte função de energia:

$$E(\mathbf{x}) = \sigma(\mathbf{x}) f_0(\mathbf{x}) + \mu \sum_{i=1}^{m} (\mathbf{a}_i\mathbf{x} - b_i)\mathrm{uhsgn}(\mathbf{a}_i\mathbf{x}_i - b_i) \tag{4.19}$$

onde μ é uma constante positiva, os termos $\mathbf{a}_i\mathbf{x} - b_i$ representam as restrições de desigualdade afins e

$$\sigma(\mathbf{x}) = \begin{cases} 1 & \forall \mathbf{x} \in \chi \\ 0 & \forall \mathbf{x} \notin \chi \end{cases} \tag{4.20}$$

[2]Os valores mínimos de λ_i e ν_i correspondem aos multiplicadores de Lagrange λ_i^* e ν_i^*, os quais zeram a primeira das condições KKT (ver [17, p. 243]).

O fator $\sigma(\mathbf{x})$ pode ser avaliado como

$$\sigma(\mathbf{x}) = \prod_{i=1}^{m}(1 - \text{uhsgn}(f_i(\mathbf{x}))) \prod_{i=1}^{p}(1 - |\text{sgn}(h_i(\mathbf{x})|))$$

comentendo o abuso matemático de assumir $\text{sgn}(0) = 0$ e $\text{uhsgn}(0) = 0$.

Em [32] é considerada a mesma função de energia porém estende a análise a restrições de desigualdade convexas em geral, sem entretanto permitir restrições de igualdade.

Neste capítulo serão estendidos e generalizados os resultados obtidos por [86] e [32] para o problema geral de otimização convexa, modificando a função de energia (4.19).

Define-se

$$E(\mathbf{x}) := \sigma(\mathbf{x}) f_0(\mathbf{x}) + \mathbf{f}^T(\mathbf{x})\, \Lambda\, \text{uhsgn}(\mathbf{f}(\mathbf{x})) \; + \; \mathbf{h}^T(\mathbf{x})\, \Gamma\, \text{sgn}(\mathbf{h}(\mathbf{x})) \tag{4.21}$$

sendo $\Lambda \in \mathbb{R}^{m\times m}$ tal que $\Lambda = diag(\lambda_i) \succ 0$ e $\Gamma \in \mathbb{R}^{p\times p}$ tal que $\Gamma = diag(\nu_i) \succ 0$.

Note-se que $E(\mathbf{x})$ não é diferenciável em todas as superfícies tal que $f_i(\mathbf{x}) = 0$ para todo $i \in \{1, .., m\}$, ou $h_i(\mathbf{x}) = 0$ para todo $i \in \{1, ..p\}$. Estas superfícies incluem a fronteira do conjunto viável, onde $E(\mathbf{x})$ também não é contínua.

Observa-se aqui que não há exigências com respeito ao valor mínimo dos ganhos, como em [78], apenas que sejam positivos, pois o fator $\sigma(\mathbf{x})$ anula a função objetivo fora do conjunto viável, e para todo $\mathbf{x} \notin \chi : \; E(\mathbf{x}) \in (0, \infty)$. Dentro do conjunto viável, são os termos de penalização que se anulam, fazendo que $\min E(\mathbf{x}) = f_0(\mathbf{x}^*) = p^*$.

Portanto:

$$E(\mathbf{x}) = \begin{cases} f_0(\mathbf{x}) & \forall \mathbf{x} \in \chi \\ \mathbf{f}^T(\mathbf{x})\, \Lambda\, \text{uhsgn}(\mathbf{f}) + \mathbf{h}^T(\mathbf{x})\, \Gamma\, \text{sgn}(\mathbf{h}) & \forall \mathbf{x} \notin \chi \end{cases}$$

A função (4.21) possui um lado direito que apresenta descontinuidades sobre a fronteira do conjunto viável. Funções com lado direito descontínuo foram inicialmente estudadas em [28] e utilizadas em abundante bibliografia (ver, por exemplo [78], [23] e [68]). Tais funções originam sistemas a estrutura variável, e o comportamento típico das trajetórias na dinâmica destes sistemas em malha fechada é o de modos deslizantes.

Em particular, se sobre uma superfície $h_i(\mathbf{x}) = 0$, as condições

$$\lim_{h_i \to 0^+} h_i(\mathbf{x}) < 0 \quad \text{e} \quad \lim_{h_i \to 0^-} h_i(\mathbf{x}) > 0$$

se mantêm, então um deslocamento em modo deslizante ocorre sobre esta superfície (ver [78]).

Seguindo a abordagem de funções de Liapunov de controle (CLF), escolhe-se como função de Liapunov candidata (sobre o método direto de Liapunov, ver [69, c. 3])

$$V(\mathbf{x}) = E(\mathbf{x}) - \sigma(\mathbf{x})p^* = E(\mathbf{x}) - \sigma(\mathbf{x})f_0(\mathbf{x}^*) \tag{4.22}$$

a qual será não negativa tanto dentro como fora do conjunto viável.

Para evitar no cálculo da lei de atualização das variáveis a avaliação dos gradientes generalizados dos termos descontínuos, opta-se por definir um sistema dinâmico que será analisado a seguir.

$$\dot{\mathbf{x}} = -\kappa \left[\sigma(\mathbf{x})\nabla f_0(\mathbf{x}) + D_{\mathbf{f}}^T(\mathbf{x})\,\Lambda\,\mathrm{uhsgn}(\mathbf{f}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\,\Gamma\,\mathrm{sgn}(\mathbf{h}(\mathbf{x}))\right] := \mathbf{m}(\mathbf{x}) \tag{4.23}$$

sendo κ um ganho escalar positivo. Observe-se que $\dot{\mathbf{x}}$ apresenta descontinuidades na fronteira do conjunto viável e pode apresentar descontinuidades fora deste também, sendo portanto o lado direito de (4.23), $\mathbf{m}(\mathbf{x})$, um mapeamento de ponto em conjunto (*set valued map*). Por tal motivo, (4.23) pode ser substituído para efeitos de análise pela seguinte inclusão diferencial:

$$\dot{\mathbf{x}} \in \mathcal{F}[\mathbf{m}](\mathbf{x}) \tag{4.24}$$

onde $\mathcal{F}[\mathbf{m}](\mathbf{x})$ denota a solução de Filippov de (4.23) (ver definição em [23, p. 22] e [68]). Note-se que, fora do conjunto viável, esta definição coincide com o gradiente generalizado de Clarke da função de energia por ser $E(\mathbf{x})$ Lipschitz contínua nesta região (ver definição em [23, p. 32] e [68]). Isto é:

$$\forall \mathbf{x} \notin \chi \quad \dot{\mathbf{x}} \in -\kappa \partial E(\mathbf{x})$$

Em seguida, será analisada a convergência do algoritmo tanto dentro como fora do conjunto viável.

4.5 Análise da convergência

4.5.1 Fase de alcançar o conjunto viável

Se

$$\mathbf{x} \notin \chi \Rightarrow \dot{\mathbf{x}} = -\kappa \left[D_{\mathbf{f}}^T(\mathbf{x}) \, \Lambda \, \mathrm{uhsgn}(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \, \Gamma \, \mathrm{sgn}(\mathbf{h}) \right] \tag{4.25}$$

Para um determinado ponto $\mathbf{x}$, define-se $I(\mathbf{x}) = \{i \,|\, f_i(\mathbf{x}) \leq 0\} \cup \{i \,|\, h_i(\mathbf{x}) = 0\}$, isto é, o conjunto de índices cujas restrições são satisfeitas no ponto $\mathbf{x}$, e $J(\mathbf{x}) = \{j \,|\, f_j(\mathbf{x}) > 0\} \cup \{j \,|\, h_j(\mathbf{x}) \neq 0\}$, ou conjunto de índices cujas restrições não são satisfeitas no ponto $\mathbf{x}$. Observe-se que para todo $\mathbf{x} \notin \chi$, o conjunto $J(\mathbf{x})$ é não vazio, e para todo $\mathbf{x} \in \chi$, $I(\mathbf{x}) = \{1, .., m + p\}$.

Dadas estas definições dos conjuntos de índices podemos reescrever

$$D_{\mathbf{f}}^T(\mathbf{x}) \, \Lambda \, \mathrm{uhsgn}(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \, \Gamma \, \mathrm{sgn}(\mathbf{h}) =$$

$$\textstyle\sum_{i=1}^{m} \nabla f_i(\mathbf{x}) \, \lambda_i \, \mathrm{uhsgn}(f_i) + \sum_{i=1}^{p} \nabla h_i(\mathbf{x}) \, \nu_i \, \mathrm{sgn}(h_i) =$$

$$\sum_{\forall j \in J(\mathbf{x})} \nabla f_j(\mathbf{x}) \lambda_j + \sum_{\forall j \in J(\mathbf{x})} \nabla h_j(\mathbf{x}) \, \nu_j \, \mathrm{sgn}(h_j) \qquad \lambda_j, \nu_j > 0 \tag{4.26}$$

Neste ponto, cabem algumas observações sobre as restrições:

a) Se χ é vazio o problema é irrealizável.

b) Se $\chi = \mathbb{R}^n$ por ser $f_i(\mathbf{x})$ convexa e $h_i(\mathbf{x})$ afim, para todo $\mathbf{x} \in \mathbb{R}^n : \; f_i(\mathbf{x}) \leq 0$, $h_i(\mathbf{x}) = 0$ e as restrições são redundantes. Neste caso, para todo $i \in \{1, .., m\}$, $f_i(\mathbf{x})$ é afim e $p^* = \min_{\mathbf{x}} f_0(\mathbf{x}) = f_0(\mathbf{x}^*)$.

c) Se χ é não vazio e $\chi \neq \mathbb{R}^n$; $\forall \mathbf{x} \notin \chi$, $\forall j \in J(\mathbf{x}) : \; \nabla f_j(\mathbf{x}) \neq 0$ e $\nabla h_j(\mathbf{x}) \neq 0$.

A seguir, serão analisados os percursos das trajetórias tanto nas superfícies de descontinuidade de $\dot{\mathbf{x}}$ como fora delas.

Caso 1)

Primeiramente, sera analisado o caso $\mathbf{x} \notin \chi$ tal que para todo $i \in I(\mathbf{x}):\ f_i(\mathbf{x}) < 0$ e não existe $i \in I(\mathbf{x})$ tal que $h_i(\mathbf{x}) = 0$, isto é, para pontos fora das superfícies de descontinuidade de (4.25). Neste caso, $\dot{\mathbf{x}}$ é Lipschitz contínua.

Escolhendo a função de Liapunov candidata (4.22) nesta fase e para este caso[3]

$$V_{rp}(\mathbf{x}) = E(\mathbf{x}) - \sigma(\mathbf{x})p^* = \mathbf{f}^T(\mathbf{x})\Lambda \text{uhsgn}(\mathbf{f}) + \mathbf{h}^T(\mathbf{x})\Gamma \text{sgn}(\mathbf{h}) > 0 \quad \forall \mathbf{x} \notin \chi \tag{4.27}$$

$$\begin{aligned} \dot{V}_{rp}(\mathbf{x}) &= [D_{\mathbf{f}}^T(\mathbf{x})\,\Lambda\,\text{uhsgn}(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\,\Gamma\,\text{sgn}(\mathbf{h})]^T\dot{\mathbf{x}} \\ &= -\kappa\|D_{\mathbf{f}}^T(\mathbf{x})\,\Lambda\,\text{uhsgn}(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\,\Gamma\,\text{sgn}(\mathbf{h})\|_2^2 \leq 0 \end{aligned}$$

Note-se que os termos que contêm o gradiente generalizado das funções descontínuas são iguais a zero.

A seguir, demonstraremos que a trajetória $\mathbf{x}(t)$ a partir de um ponto inicial $\mathbf{x}_0 = \mathbf{x}(0) \notin \chi$ não se estaciona em um ponto $\mathbf{x} \notin \chi$, isto é, que $\dot{V}_{rp} \neq 0$ nesta fase e para este caso, através do teorema:

Teorema 4.5.1 *Para todo $\mathbf{x} \notin \chi$, não existe $\lambda_i, \nu_i > 0$ tais que (4.26) seja igual a zero, sendo portanto $\dot{V}_{rp} < 0$ para todo $\mathbf{x} \notin \chi$.*

Prova:

$\forall \mathbf{x}_1 \notin \chi,\ \forall \mathbf{x}_2 \in \chi,\ \forall j \in J(\mathbf{x}_1):\ f_j(\mathbf{x}_1) > 0,\ f_j(\mathbf{x}_2) \leq 0,\ h_j(\mathbf{x}_1) \neq 0,\ h_j(\mathbf{x}_2) = 0,\ \nabla f_j(\mathbf{x}_1) \neq 0,\ \nabla h_j(\mathbf{x}_1) \neq 0$

Assumindo primeiro $h_j(\mathbf{x}_1) > 0$

Por condição de primeira ordem das funções convexas [17, c. 3]:

$$\nabla^T f_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) \leq f_j(\mathbf{x}_2) - f_j(\mathbf{x}_1) < 0 \tag{4.28}$$

$$\nabla^T h_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) = h_j(\mathbf{x}_2) - h_j(\mathbf{x}_1) < 0 \tag{4.29}$$

Por ser χ convexo (lema 4.3.1) e pelo teorema do hiperplano separador

$\exists\, \mathbf{c} \in \mathbb{R}^n,\ d \in \mathbb{R}$ *tal que* $\forall \mathbf{x}_1 \notin \chi,\ \mathbf{x}_2 \in \chi:\ \mathbf{c}^T\mathbf{x}_1 - d > 0 \quad \mathbf{c}^T\mathbf{x}_2 - d < 0$

$$\Rightarrow\ \mathbf{c}^T(\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{c}^T\mathbf{x}_2 - d - \mathbf{c}^T\mathbf{x}_1 + d < 0 \tag{4.30}$$

[3]Em [11] e [13], observa-se que esta é uma função de Liapunov tipo Persidskii, explicitando $E(\mathbf{x}) = \sum_{i=1}^m \max\{0, f_i\} + \sum_{i=1}^p |h_i| = \sum_{i=1}^m \int_0^{f_i} \text{uhsgn}(\tau)\partial\tau + \sum_{i=1}^p \int_0^{h_i} \text{sgn}(\tau)\partial\tau$

O vetor $\mathbf{c}$ *pode ser qualquer vetor* $\in \mathbb{R}^n$ *que satisfaça (4.30). Em particular, pode ser escolhido um hiperplano separador tal que* $\mathbf{c}^T(\mathbf{x}_2 - \mathbf{x}_1) = -1$

$$\Rightarrow \ \mathbf{c} = -\frac{(\mathbf{x}_2 - \mathbf{x}_1)}{\|\mathbf{x}_2 - \mathbf{x}_1\|_2^2}$$

$$\Rightarrow \nabla^T f_j(\mathbf{x}_1)\mathbf{c} = -\frac{\nabla^T f_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1)}{\|\mathbf{x}_2 - \mathbf{x}_1\|_2^2} > 0 \tag{4.31}$$

Idem para $h_j(\mathbf{x}_1)$.

No caso de existir $j \in J(\mathbf{x}_1)$ *tal que* $h_j(\mathbf{x}_1) < 0$

$$\nabla^T h_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) = h_j(\mathbf{x}_2) - h_j(\mathbf{x}_1) > 0 \Rightarrow \ -\nabla^T h_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) < 0$$

$$\Rightarrow \ -\nabla^T h_j(\mathbf{x}_1)\mathbf{c} = \frac{\nabla^T h_j(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1)}{\|\mathbf{x}_2 - \mathbf{x}_1\|_2^2} > 0 \tag{4.32}$$

De (4.31)(para o caso $h_j(\mathbf{x}_1) > 0$*) e (4.32), conclue-se*

$$sgn(h_j(\mathbf{x}_1))\nabla^T h_j(\mathbf{x}_1)\mathbf{c} > 0 \tag{4.33}$$

As equacões (4.31) e (4.33) indicam que todos os vetores $\nabla f_j(\mathbf{x}_1)$ *e* $sgn(h_j(\mathbf{x}_1))\nabla h_j(\mathbf{x}_1)$, $\forall j \in J(\mathbf{x}_1)$ *estão do mesmo lado do hiperplano separador* $\mathbf{c}^T\mathbf{x} = d$. *Portanto não existe combinação linear positiva que zere (4.26).*

$$\Rightarrow D_{\mathbf{f}}^T(\mathbf{x})\,\Lambda\,\mathrm{uhsgn}(\mathbf{f}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\,\Gamma\,\mathrm{sgn}(\mathbf{h}(\mathbf{x})) < 0$$

$$\dot{V}_{rp} = -\kappa\|\nabla V_{rp}\|_2^2 < 0$$

cometendo um abuso de notação pois $\nabla V_{rp}(\mathbf{x})$ pode não estar definido para todo $\mathbf{x} \notin \chi$. Por ser um sistema dinâmico gradiente, a convergência assintótica ao conjunto viável é garantida (ver [44]).

Caso 2)

Neste caso, assumiremos que existe $i \in I(\mathbf{x})$ tal que $f_i(\mathbf{x}) = 0$ ou $h_i(\mathbf{x}) = 0$, isto é, que o ponto $\mathbf{x} \notin \chi$ está sobre uma superfície de descontinuidade.

Para facilitar a demonstração, escolheremos uma função de Liapunov candidata

alternativa:

$$V_{rp2}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|^2 \quad \text{para algum } \mathbf{x}^* \in \chi^* \tag{4.34}$$

Esta função de Liapunov candidata é positiva para todo $\mathbf{x} \notin \chi$ e só será zero para $\mathbf{x} = \mathbf{x}^*$. $V_{rp2}(\mathbf{x})$ é Lipschitz contínua e continuamente diferenciável.

A derivada de Lie de (4.34) ao longo das trajetórias definidas por $\dot{\mathbf{x}}$ é dada por ([23, p. 42] e [68]):

$$\dot{V}_{rp2}(\mathbf{x}) \in \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{rp2}(\mathbf{x}) = 2(\mathbf{x} - \mathbf{x}^*)^T \dot{\mathbf{x}} \tag{4.35}$$

A seguir, demonstraremos que a trajetória $\mathbf{x}(t)$ a partir de um ponto inicial $\mathbf{x}_0 = \mathbf{x}(0) \notin \chi$ não se estaciona em um ponto $\mathbf{x} \notin \chi$, isto é, que $\dot{V}_{rp2} \neq 0$ nesta fase e para este caso, através do teorema:

Teorema 4.5.2 *Para todo $\mathbf{x} \notin \chi$ tal que existe $i \in I(\mathbf{x})$ tal que $f_i(\mathbf{x}) = 0$ ou $h_i(\mathbf{x}) = 0$, $\dot{V}_{rp2}(\mathbf{x}) < 0$, isto é, a trajetória não se estaciona em nenhum ponto $\mathbf{x} \notin \chi$.*
Prova:
Supondo inicialmente $f_i(\mathbf{x}) = 0$, para algum $i \in I(\mathbf{x})$, o vetor $\dot{\mathbf{x}}$ nessa descontinuidade pertence a $-\kappa$ vezes o gradiente generalizado de $E(\mathbf{x})$, o qual se define da seguinte maneira ([23, p. 32], [68]):

$$\begin{aligned} \dot{\mathbf{x}} \in -\kappa \partial E(\mathbf{x}) = co\{ \quad & -\kappa \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) \right] , \\ & -\kappa \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) + \lambda_i \nabla f_i(\mathbf{x}) \right] \} \end{aligned} \tag{4.36}$$

onde co denota o fecho convexo entre vetores (ver [68]).

A derivada de Lie expressada em (4.35), resulta:

$$\begin{aligned} \dot{V}_{rp2}(\mathbf{x}) \in \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{rp2}(\mathbf{x}) \quad & = 2 \ co \ \{ -\kappa (\mathbf{x} - \mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) \right] , \\ & \quad -\kappa (\mathbf{x} - \mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) + \lambda_i \nabla f_i(\mathbf{x}) \right] \} \end{aligned}$$

Para todo $\mathbf{x}^* \in \chi^*$, $\mathbf{x} \notin \chi$, $j \in J(\mathbf{x})$: $f_j(\mathbf{x}^*) \leq 0$ e $f_j(\mathbf{x}) > 0$. Por condição de primeira ordem das funções convexas:

$$\nabla^T f_j(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) \leq f_j(\mathbf{x}^*) - f_j(\mathbf{x}) < 0 \;\Rightarrow\; \nabla^T f_j(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) > 0 \tag{4.37}$$

Para todo $\mathbf{x}^* \in \chi^*$, $\mathbf{x} \notin \chi$, $j \in J(\mathbf{x})$: $h_j(\mathbf{x}^*) = 0$, $sgn(h_j(\mathbf{x}))h_j(\mathbf{x}) > 0$. *Por ser* $h_j(\mathbf{x})$ *afim:*

$$sgn(h_j(\mathbf{x}))\nabla^T h_j(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) = sgn(h_j(\mathbf{x}))(h_j(\mathbf{x}) - h_j(\mathbf{x}^*)) > 0 \tag{4.38}$$

De (4.26), (4.37) e (4.38), conclue-se:

$$\forall \mathbf{x}^* \in \chi^*,\ \forall \mathbf{x} \notin \chi, \quad (\mathbf{x} - \mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) \right] > 0 \tag{4.39}$$

Supondo que existe $i \in I(\mathbf{x})$ *tal que* $f_i(\mathbf{x}) = 0$, *para todo* $\mathbf{x}^* \in \chi^*$, $f_i(\mathbf{x}^*) \leq 0$. *Por ser* $f_i(\mathbf{x})$ *convexa e por condição de primeira ordem das funções convexas:*

$$\nabla^T f_i(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) \leq f_i(\mathbf{x}^*) - f_i(\mathbf{x}) \leq 0 \;\Rightarrow\; (\mathbf{x} - \mathbf{x}^*)^T \nabla f_i(\mathbf{x}) \geq 0 \tag{4.40}$$

De (4.39) e (4.40), conclue-se:

$$(\mathbf{x} - \mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) + \lambda_i \nabla f_i(\mathbf{x}) \right] > 0 \tag{4.41}$$

Portanto

$$\max\ \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{rp2}(\mathbf{x}) < 0$$

e as trajetórias não se estacionam em nenhum ponto $\mathbf{x} \notin \chi$ *([23, teorema 1]).*

Para o caso de existir $i \in I(\mathbf{x})$ *tal que* $h_i(\mathbf{x}) = 0$, *o vetor* $\dot{\mathbf{x}}$ *é expressado como:*

$$\begin{aligned} \dot{\mathbf{x}} \in -\kappa \partial E(\mathbf{x}) = co\ \{ \quad & -\kappa \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) + \nu_i \nabla h_i(\mathbf{x}) \right] , \\ & -\kappa \left[D_{\mathbf{f}}^T(\mathbf{x}) \Lambda uhsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x}) \Gamma sgn(\mathbf{h}) - \nu_i \nabla h_i(\mathbf{x}) \right] \} \end{aligned} \tag{4.42}$$

e a derivada de Lie expressada em (4.35) resulta:

$$\dot{V}_{rp2}(\mathbf{x}) \in \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{rp2}(\mathbf{x}) =$$
$$2\ co\ \{\ \ -\kappa(\mathbf{x}-\mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x})\Lambda u hsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma sgn(\mathbf{h}) + \nu_i \nabla h_i(\mathbf{x})\right]\ ,$$
$$-\kappa(\mathbf{x}-\mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x})\Lambda u hsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma sgn(\mathbf{h}) - \nu_i \nabla h_i(\mathbf{x})\right]\}$$

Para todo $\mathbf{x}^* \in \chi^*$, $h_i(\mathbf{x}^*) = 0$. *Por ser* h_i *afim*

$$\nabla^T h_i(\mathbf{x})(\mathbf{x}^* - \mathbf{x}) = h_i(\mathbf{x}^*) - h_i(\mathbf{x}) = 0 \tag{4.43}$$

De (4.39) e (4.43), conclue-se

$$(\mathbf{x}-\mathbf{x}^*)^T \left[D_{\mathbf{f}}^T(\mathbf{x})\Lambda u hsgn(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma sgn(\mathbf{h}) \pm \nu_i \nabla h_i(\mathbf{x})\right] > 0 \tag{4.44}$$

Portanto:

$$\max\ \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{rp2}(\mathbf{x}) < 0$$

e as trajetórias não se estacionam em nenhum ponto $\mathbf{x} \notin \chi$ *([23, teorema 1]).*

Note-se que na demostração do teorema 4.5.2 está incluida a do 4.5.1, porque para todo $\mathbf{x} \notin \chi$ tal que para todo $i \in I(\mathbf{x})$, $f_i(\mathbf{x}) < 0$, e não existe $i \in I(\mathbf{x})$ tal que $h_i(\mathbf{x}) = 0$, isto é, fora das superfícies de descontinuidade de $\dot{\mathbf{x}}$ (caso 1), $\dot{\mathbf{x}} \in \{-\kappa \left[D_{\mathbf{f}}^T(\mathbf{x})\Lambda \mathrm{uhsgn}(\mathbf{f}) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma \mathrm{sgn}(\mathbf{h})\right]\}$, e a equação (4.39) demonstra que $\dot{V}_{rp2}(\mathbf{x}) < 0$ neste ponto.

Estes teoremas demonstram que as trajetórias não se estacionam em qualquer $\mathbf{x} \notin \chi$, mesmo atravessando superfícies de descontinuidade de $\dot{\mathbf{x}}$.

A pesar de (4.25) constituir um sistema a estrutura variável, a convergência em tempo finito não pode ser garantida.

Exemplo 4.5.3 *Sejam* $m = 1$, $p = 0$, $\kappa = 1$ *e*

$$f_1(\mathbf{x}) = x_1^2 + x_2^2\ :\ \mathbb{R}^2 \to \mathbb{R} \quad \Rightarrow \chi = \{\mathbf{0}\}$$

$$\forall \mathbf{x} \notin \chi \quad \dot{\mathbf{x}} = -2\lambda_1 \mathbf{x} \Rightarrow \mathbf{x}(t) = \mathbf{x}(0)e^{-2\lambda_1 t}$$

4.5.2 Fase de convergência

Utilizando a lei de atualização das variáveis (4.23) e a função de Liapunov candidata (4.22):

$$\forall \mathbf{x} \in \chi : \quad \dot{\mathbf{x}} = -\kappa \nabla f_0(\mathbf{x})$$

Nesta fase $\dot{\mathbf{x}}$ é Lipschitz contínua.

$$V_{cp} = E(\mathbf{x}) - f_0(\mathbf{x}^*) \text{ sendo } \mathbf{x}^* \in \chi^* \text{ um ponto ótimo}$$

$$\Rightarrow V_{cp} = f_0(\mathbf{x}) - f_0(\mathbf{x}^*) \geq 0$$

$$\Rightarrow \dot{V}_{cp} = \nabla^T f_0(\mathbf{x})\dot{\mathbf{x}} = -\kappa \|\nabla f_0(\mathbf{x})\|_2^2 = -\kappa \|\nabla V_{cp}\|_2^2 \leq 0$$

Pelo lema de Barbalat ([69, s. 4.5]), por ser $\dot{V}_{cp}(\mathbf{x})$ contínua nesta fase, $\dot{V}_{cp}(\mathbf{x})$ é decrescente, e tenderia a zero quando t tende a ∞, se estacionando só no mínimo global de $f_0(\mathbf{x})$, caso este exista.

No caso trivial, sendo $f_0(\mathbf{x})$ estritamente convexa e $\min f_0(\mathbf{x}) \in \chi$, a convergência das trajetórias ao mínimo global efetivamente se produz nesta fase. No caso geral, as trajetórias convergem à fronteira do conjunto viável.

4.5.3 Fase sobre a fronteira do conjunto viável

Sobre a fronteira do conjunto viável, denominada $\partial\chi$, a lei de atualização das variáveis (4.23) é descontínua (embora também apresente descontinuidades fora do conjunto viável).

Por ser $\dot{\mathbf{x}}$ contínua por partes, a solução de Filippov adota uma expressão simples (ver [23, p. 22]). Definindo:

$$\forall \mathbf{x} \in \partial\chi : \quad \mathbf{g}(\mathbf{x}) = \lim_{\substack{\mathbf{x}_i \to \mathbf{x} \\ \mathbf{x}_i \notin \chi}} \left[D_{\mathbf{f}}^T(\mathbf{x}_i)\Lambda \text{uhsgn}(\mathbf{f}(\mathbf{x}_i)) + D_{\mathbf{h}}^T(\mathbf{x}_i)\Gamma \text{sgn}(\mathbf{h}(\mathbf{x}_i)) \right]$$

a lei de atualização das variáveis (4.23) sobre a fronteira do conjunto viável resulta

$$\forall \mathbf{x} \in \partial\chi : \quad \dot{\mathbf{x}} \in \mathcal{F}[\mathbf{m}](\mathbf{x}) = \text{co}\{-\kappa \nabla f_0(\mathbf{x}) \ , \ -\kappa \mathbf{g}(\mathbf{x})\} \tag{4.45}$$

onde co denota o fecho convexo entre vetores (ver [68]).

Observe-se que esta definição de solução de Filippov refere-se apenas à fronteira do conjunto viável, e não a todo o espaço $\mathbb{R}^n$ porque existem descontinuidades nos gradientes dos termos de penalização fora do conjunto viável também.

O vetor $\mathbf{g}(\mathbf{x})$ é perpendicular à fronteira do conjunto viável para quase todo $\mathbf{x} \in \partial\chi$. Nas quinas do conjunto viável $\mathbf{g}(\mathbf{x}) \in \mathcal{N}(\mathbf{x}, \chi)$, sendo $\mathcal{N}(\mathbf{x}, \chi)$ o cone normal ao conjunto χ no ponto $\mathbf{x}$ ([26]).

Existem duas possibilidades de percurso da trajetória uma vez alcançada a fronteira do conjunto viável. Nos casos

i) Para todo $\mathbf{x} \in \partial\chi$, se $\nabla^T f_0(\mathbf{x})\mathbf{g}(\mathbf{x}) > 0$ a trajetória atravessa a fronteira do conjunto viável, entrando na fase de convergência.

ii) Para todo $\mathbf{x} \in \partial\chi$, se $\nabla^T f_0(\mathbf{x})\mathbf{g}(\mathbf{x}) \leq 0$ se produz uma trajetória em modo deslizante a partir do ponto de contato com a fronteira do conjunto viável.

A seguinte figura ilustra ambos casos.

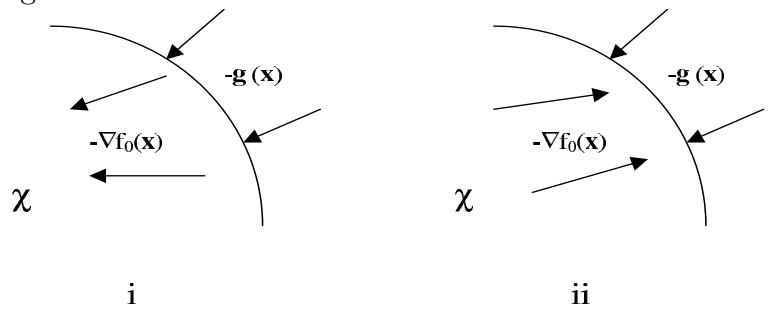

Figura 4.1: Trajetórias atravessando a fronteira do conjunto viável
onde em i) não se produz modo deslizante e em ii) se produz modo deslizante

Destacamos que trajetórias em modo deslizante também poderiam se produzir fora do conjunto viável, não sendo afetada a análise de convergência realizada na seção **Fase de alcançar o conjunto viável**.

No caso ii), por ser $\dot{\mathbf{x}}$ mensurável e localmente essencialmente limitada, existe pelo menos uma solução de Filippov para $\dot{\mathbf{x}} \in \mathcal{F}[\mathbf{m}]$ (ver [23, prop. 4]).

Também neste caso, por serem, para todo $\mathbf{x} \in \partial\chi$, $\nabla f_0(\mathbf{x})$, $\mathbf{g}(\mathbf{x})$[4], e $\nabla f_0(\mathbf{x}) - \mathbf{g}(\mathbf{x})$

[4]Note-se que para todo $\mathbf{x} \in \partial\chi$, $\mathbf{g}(\mathbf{x})$ está definido como o limite quando $\mathbf{x}_i \to \mathbf{x}$, mas sem alcançá-lo. Eis a razão pela qual $\mathbf{g}(\mathbf{x})$ é diferenciável.

continuamente diferenciáveis sobre $\partial\chi$, e por $-\kappa\mathbf{g}(\mathbf{x})$ apontar para χ, e $-\kappa\nabla f_0(\mathbf{x})$ apontar para fora do conjunto viável, então a solução de Filippov é única ([23, prop. 6]).

Escolhendo nesta fase a seguinte função de Liapunov candidata

$$\forall \mathbf{x} \in \partial\chi,\ \forall \mathbf{x}^* \in \chi^*, \quad V_{bp}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|^2 \tag{4.46}$$

Conferimos:

i) Evidentemente, esta função é positiva para todo $\mathbf{x} \in \partial\chi \backslash \chi^*$, e $V_{bp}(\mathbf{x}^*) = 0$ para todo $\mathbf{x}^* \in \chi^*$. Portanto é positiva definida.

ii) $\mathbf{0} \in \mathcal{F}[\mathbf{m}](\mathbf{x}^*)$. Portanto $\mathbf{x}^*$ é ponto de equilíbrio de (4.45). Se $\mathbf{x}^*$ estiver em uma quina do conjunto viável, dependerá do caminho do limite na definição de $\mathbf{g}(\mathbf{x})$ para $\mathbf{0} \in \mathcal{F}[\mathbf{m}](\mathbf{x}^*)$.

iii) No caso não trivial $\chi^* \subset \partial\chi$. No caso trivial, as trajetórias alcançam o ponto ótimo durante a fase de convergência.

iv) $V_{bp}(\mathbf{x})$ é Lipschitz contínua para todo $\mathbf{x} \in \mathbb{R}^n$.

v) $\mathcal{F}[\mathbf{m}](\mathbf{x})$ é superiormente semi-contínua para todo $\mathbf{x} \in \partial\chi$ e localmente limitada.

vi) Da equação (4.4), por ser $f_0(\mathbf{x})$ convexa:

$$\begin{aligned} &\forall \mathbf{x}_1,\ \mathbf{x}_2 \in \mathbb{R}^n, \quad \left[\nabla f_0(\mathbf{x}_1) - \nabla f_0(\mathbf{x}_2)\right]^T (\mathbf{x}_1 - \mathbf{x}_2) \geq 0 \\ \Rightarrow\ &\forall \mathbf{x} \in \chi, \quad \nabla^T f_0(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) \geq \nabla^T f_0(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0 \end{aligned} \tag{4.47}$$

Por, para todo $\mathbf{x} \in \partial\chi,\ \mathbf{g}(\mathbf{x})$ ser perpendicular à fronteira do conjunto viável, ou, caso o ponto $\mathbf{x}$ estar em uma quina deste, $\mathbf{g}(\mathbf{x}) \in \mathcal{N}(\mathbf{x}, \chi)$, e ser χ convexo:

$$\begin{aligned} &\forall \mathbf{y} \in \chi,\ \forall \mathbf{x} \in \partial\chi, \quad \mathbf{g}^T(\mathbf{x})(\mathbf{y} - \mathbf{x}) \leq 0 \\ &\Rightarrow\ \forall \mathbf{x} \in \partial\chi, \quad \mathbf{g}^T(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) \geq 0 \end{aligned} \tag{4.48}$$

A derivada de Lie ao longo da fronteira onde se produz a trajetória por modo deslizante está definida ([23]):

$$\dot{V}_{bp}(\mathbf{x}) \in \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{bp}(\mathbf{x}) = 2(\mathbf{x} - \mathbf{x}^*)^T \dot{\mathbf{x}} = 2\ \mathrm{co}\{-\kappa\nabla^T f_0(\mathbf{x})(\mathbf{x} - \mathbf{x}^*), -\kappa\mathbf{g}^T(\mathbf{x})(\mathbf{x} - \mathbf{x}^*)\}$$

e de (4.47) e (4.48), concluimos:

$$\max \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V_{bp}(\mathbf{x}) \leq 0$$

e portanto o sistema (4.45), para todo $\mathbf{x} \in \partial\chi$ é estável ([23, teorema 1]).

Observe-se que, para todo $\mathbf{x} \in \partial\chi$, $(\mathbf{x} - \mathbf{x}^*)^T \nabla f_0(\mathbf{x}) = 0$ se e somente se

i) $f_0(\mathbf{x})$ é afim

ii) a restrição violada em $B(\mathbf{x}, \delta) \cap (\mathbb{R}^n \backslash \chi)$ é afim,

iii) Neste caso, para todo $\mathbf{x}' \in \chi$, $(\mathbf{x}' - \mathbf{x})^T f_0(\mathbf{x}) \geq 0$ e portanto $\mathbf{x} \in \chi^*$.

Portanto,

$$\forall \mathbf{x} \in \partial\chi,\ \mathbf{x} \notin \chi^*, \quad \nabla^T f_0(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) > 0 \tag{4.49}$$

No caso $(\mathbf{x} - \mathbf{x}^*)^T \mathbf{g}(\mathbf{x}) = 0$, para algum $\mathbf{x} \in \partial\chi$, a restrição violada em $B(\mathbf{x}, \delta) \cap (\mathbb{R}^n \backslash \chi)$ é afim e $\mathbf{g}(\mathbf{x})$ é perpendicular à fronteira do conjunto viável neste ponto. Se $\mathbf{x} \notin \chi^*$, pela equação (4.49), $\nabla f_0(\mathbf{x})$ não é perpendicular à fronteira neste ponto, e portanto $\mathbf{0} \notin \mathcal{F}[\mathbf{m}](\mathbf{x})$, o que implica que $\dot{\mathbf{x}} \neq \mathbf{0}$ e as trajetórias não se estacionam neste ponto, não sendo portanto um ponto de equilíbrio.

A seguinte figura ilustra este caso.

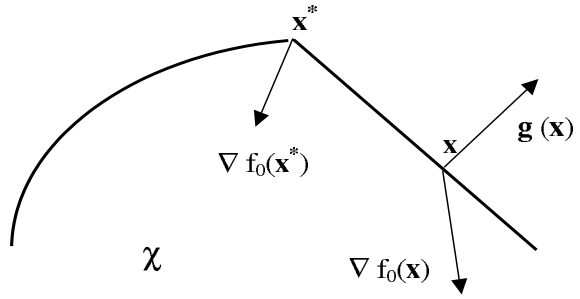

Figura 4.2: Vetores $\nabla f_0(\mathbf{x})$ e $\mathbf{g}(\mathbf{x})$ para um ponto $\mathbf{x} \in \partial\chi$ tal que a restrição violada em um entorno de $\mathbf{x}$ é afim

Concluimos então que as trajetórias geradas pelo sistema (4.13)-(4.45), para todo $\mathbf{x} \in \partial\chi$, convergem assintoticamente a um elemento do conjunto solução $\mathbf{x}^* \in \chi^*$.

Enunciamos, assim, o seguinte teorema.

Teorema 4.5.4 *Para toda função objetivo convexa sujeita a restrições de desigualdade convexas e de igualdade afins, todas diferenciáveis, a trajetória determinada pelo sis-*

tema (4.13)-(4.23), para todo $\mathbf{x}_0 \in \mathbb{R}^n$ *é globalmente assintoticamente convergente ao conjunto solução.*

Em [86, s. 6.9] é apresentada uma análise detalhada da convergência das trajetórias durante o deslizamento, sendo aplicável aqui uma análise semelhante.

Observe-se que, a diferença da maioria das bibliografias, incluindo [78], neste trabalho a exigência da função objetivo f_0 ser convexa refere-se apenas ao conjunto viável, admitindo o algoritmo qualquer forma para esta função fora de χ, pois o fator $\sigma(\mathbf{x})$ a anula nesta região. Inclusive, dentro do conjunto viável a função objetivo pode ser estritamente quase-convexa, pois não existem pontos de gradiente zero neste tipo de funções, a não ser eventualmente um mínimo global.

4.6 Simulação do algoritmo

A título de ilustração, será simulado o seguinte exemplo apresentado em [32] e utilizado em [53].

$$\begin{aligned} \min \quad & \tfrac{1}{2}\mathbf{x}^T \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x} + [-12 \ \ -10]\,\mathbf{x} \\ \text{s.t.} \quad & x_1^2 + (x_2 - 4)^2 - 64 \leq 0 \\ & (x_1 + 3)^2 + x_2^2 - 36 \leq 0 \\ & (x_1 - 3)^2 + x_2^2 - 36 \leq 0 \end{aligned}$$

O algoritmo contínuo será implementado utilizando a função ODE23 do programa MATLAB 6, com ganho $\kappa = 1$; os ganhos da função de penalização λ_i e ν_i também foram escolhidos unitários. As tolerâncias foram escolhidas **AbsTol**= $1e-6$ e **RelTol**= $1e-3$. Nas seguintes figuras representam-se trajetórias a partir de diversos pontos iniciais (tanto dentro como fora do conjunto viável) no espaço e a norma dos erros de trajetória em função do tempo a partir dos mesmos pontos iniciais.

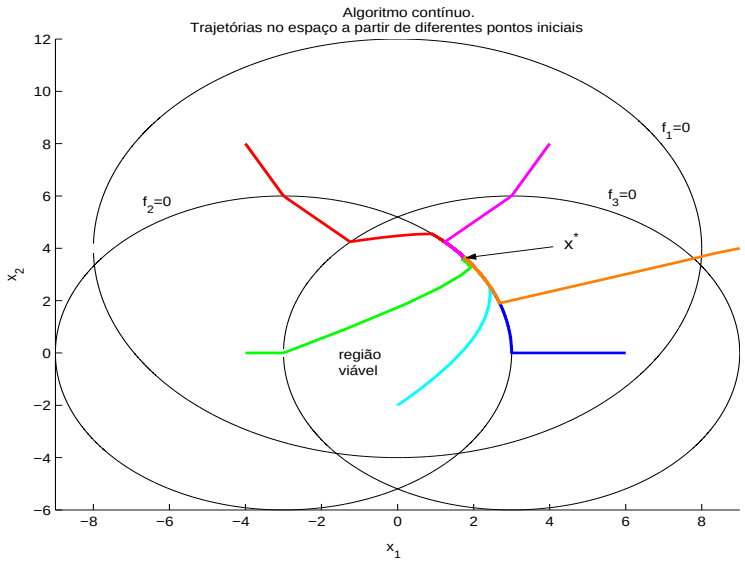

Figura 4.3: Implementação do algoritmo contínuo para o exemplo, mostrando convergência das trajetórias no espaço de estados a partir de diversos pontos iniciais.

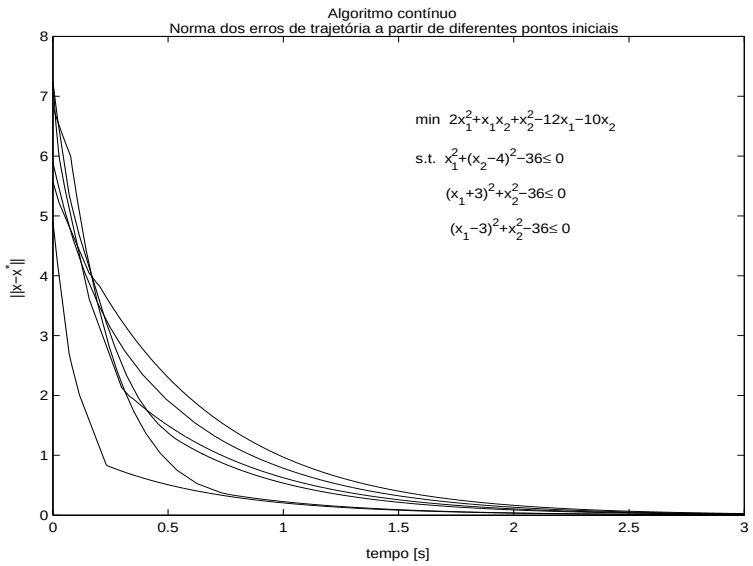

Figura 4.4: Implementação do algoritmo contínuo para o exemplo, mostrando a convergência da norma dos erros de trajetória em função do tempo a partir dos mesmos pontos iniciais.

Na figura 4.5, apresenta-se o diagrama de blocos representante da dinâmica do sistema em malha fechada.

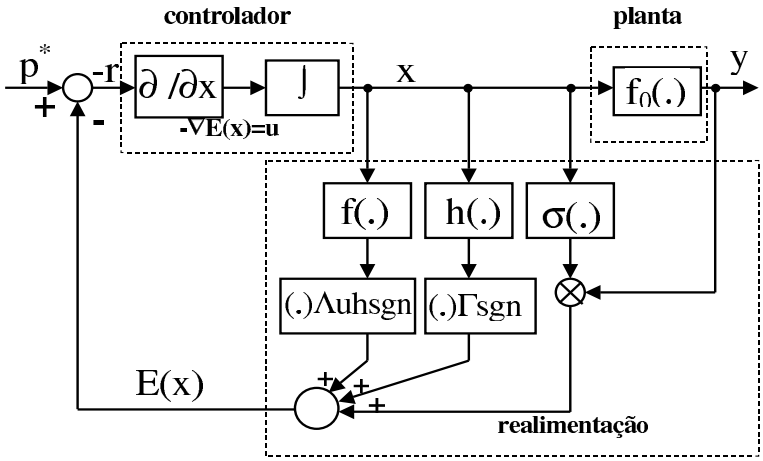

Figura 4.5: Diagrama de blocos do sistema contínuo

4.7 Discretização do algoritmo

A diferença dos algoritmos apresentados na bibliografia que se limitam a tratar o caso contínuo, discretizaremos aqui o algoritmo (4.23) calculando o comprimento do passo ótimo, ao qual daremos o nome de algoritmo gradiente a estrutura variável. Utilizando a nomenclatura da teoria de controle, define-se o resíduo ou erro como em (4.22):

$$r_k := E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\, p^* \geq 0$$

onde o $\min_{\mathbf{x}\in\chi}\ f_0(\mathbf{x}) = p^*$ pode ser substituído por um valor γ estimado[5] tal que $\gamma < p^*$.

$$\begin{aligned} r_{k+1} = \quad & r_k + \nabla^T r(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + h.o.t. \simeq r_k + \nabla^T r(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = \\ & r_k + \nabla^T E(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) \end{aligned}$$

Esta aproximação é válida sempre que $\sigma(\mathbf{x}_k) = \sigma(\mathbf{x}_{k+1})$. Aqui desconsideramos também o gradiente generalizado dos termos descontínuos observando que, por ser a fronteira do conjunto viável parte deste, para toda iteração k : $\mathbf{x}_k \in \chi$ ou $\mathbf{x}_k \notin \chi$, portanto, sem perda de generalidade, consideramos, para todo $\mathbf{x}_k$: $\nabla\sigma(\mathbf{x}_k) = 0$.

Discretizando a lei de atualização (4.23) por Euler (ver [4])

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla E(\mathbf{x}_k) \tag{4.50}$$

[5]Existe a possibilidade, não pesquisada no presente trabalho, de utilizar um mínimo estimado adaptativo, isto é, um valor variável a cada iteração tal que $\gamma_k = f(\mathbf{f}(\mathbf{x}_k))$.

$$\Rightarrow r_{k+1} \simeq r_k - \alpha_k \|\nabla E(\mathbf{x}_k)\|_2^2$$

sendo α_k o comprimento do passo.

Escolhe-se como função de Liapunov candidata discreta

$$V_k = r_k^2 = (E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\gamma)^2 > 0 \tag{4.51}$$

$$\begin{aligned} \Rightarrow \Delta V_k &\simeq V_{k+1} - V_k = r_{k+1}^2 - r_k^2 = (r_k - \alpha_k \|\nabla E(\mathbf{x}_k)\|_2^2)^2 - r_k^2 = \\ &= r_k^2 - 2r_k\alpha_k \|\nabla E(\mathbf{x}_k)\|_2^2 + \alpha_k^2 \|\nabla E(\mathbf{x}_k)\|_2^4 - r_k^2 = \\ &= -2\alpha_k (E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\,\gamma) \|\nabla E(\mathbf{x}_k)\|_2^2 + \alpha_k^2 \|\nabla E(\mathbf{x}_k)\|_2^4 \end{aligned} \tag{4.52}$$

Seguindo a metodologia de controle ótimo de Liapunov, otimiza-se o comprimento do passo α_k

$$\frac{\partial \Delta V_k}{\partial \alpha_k} = -2(E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\,\gamma) \|\nabla E(\mathbf{x}_k)\|_2^2 + 2\alpha_k \|\nabla E(\mathbf{x}_k)\|_2^4 = 0$$

Portanto:

$$\alpha_k = \frac{E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\,\gamma}{\|\nabla E(\mathbf{x}_k)\|_2^2} > 0 \tag{4.53}$$

Substituindo em (4.52)

$$\Rightarrow \Delta V_k \simeq -(E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)\gamma)^2 = -r_k^2 = -V_k < 0$$

sendo, portanto, o algoritmo gradiente a estrutura variável globalmente assintoticamente convergente.

Observe-se que esta variação da função de Liapunov implica que $V_{k+1} = 0$, o qual efetivamente aconteceria caso a função de energia seja linear e portanto a aproximação por Taylor para o cálculo de $\mathbf{r}_{k+1}$, exata. Por exemplo, este seria o caso se o ponto $\mathbf{x}_k \notin \chi$ violasse apenas uma restrição afim.

Destacamos que o comprimento do passo ótimo aqui calculado é diferente daqueles mencionados em [48] e suas referências.

No caso de não ser possível estimar um valor γ tal que $\gamma < p^*$, condição exigida

para o cálculo do comprimento do passo (4.53), é possível realizar uma outra escolha de comprimento do passo. Observa-se que este novo passo é ótimo em relação a uma nova escolha da função de energia, que acarretará uma nova função de Liapunov candidata discreta, a qual será definida a seguir:

$$E(\mathbf{x}) := \|\nabla f_0(\mathbf{x})\|\sigma(\mathbf{x}) + \mathbf{f}^T(\mathbf{x})\Lambda \text{uhsgn}(\mathbf{f}(\mathbf{x})) + \mathbf{h}^T(\mathbf{x})\Gamma \text{sgn}(\mathbf{h}(\mathbf{x})) \tag{4.54}$$

embora mantendo como lei de atualização das variáveis

$$\mathbf{u}_k = -\left[\nabla f_0(\mathbf{x}_k)\sigma(\mathbf{x}_k) + D^T_{\mathbf{f}}(\mathbf{x}_k)\Lambda \text{uhsgn}(\mathbf{f}(\mathbf{x}_k)) + D^T_{\mathbf{h}}(\mathbf{x}_k)\Gamma \text{sgn}(\mathbf{h}(\mathbf{x}_k))\right] \tag{4.55}$$

de maneira tal que, discretizando (4.23) por Euler:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k$$

Esta função de energia (4.54), que permanece igual a (4.21) para todo $\mathbf{x} \notin \chi$, será igual à norma do gradiente da função objetivo para todo $\mathbf{x} \in \chi$; esta escolha justifica-se porque, na medida que a trajetória aproxima-se do mínimo, a função de energia é não crescente.

Como resíduo, portanto, pode se escolher esta função de energia, a qual pretende-se minimizar.

$$r_k := E(\mathbf{x}_k)$$

e portanto

$$r_{k+1} = E(\mathbf{x}_{k+1}) \simeq E(\mathbf{x}_k) + \alpha_k \nabla^T E(\mathbf{x}_k)\mathbf{u}_k$$

Escolhendo a mesma função de Liapunov candidata discreta (4.51)

$$V_k = r_k^2$$

e aplicando as mesmas equações com as que se chegou a (4.53), chega-se a:

$$\alpha_k = -\frac{E(\mathbf{x}_k)}{\nabla^T E(\mathbf{x}_k)\mathbf{u}_k} \tag{4.56}$$

comprimento do passo que, dentro de cada região, será igual às expressões:

$$\alpha_k = \begin{cases} \frac{\mathbf{f}^T(\mathbf{x}_k)\Lambda \mathbf{u} \mathrm{hsgn}(\mathbf{f}(\mathbf{x}_k)) + \mathbf{h}^T(\mathbf{x}_k)\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}_k))}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\Lambda \mathbf{u} \mathrm{hsgn}(\mathbf{f}(\mathbf{x}_k)) + D_{\mathbf{h}}^T(\mathbf{x}_k)\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}_k))\|^2} > 0 & \forall \mathbf{x}_k \notin \chi \\ \frac{\|\nabla f_0(\mathbf{x}_k)\|^2}{\nabla^T f_0(\mathbf{x}_k) \nabla^2 f_0(\mathbf{x}_k) \nabla f_0(\mathbf{x}_k)} > 0 & \forall \mathbf{x}_k \in \chi \end{cases}$$

sendo $\nabla^2 f_0(\mathbf{x})$ o hessiano da função objetivo. Este comprimento do passo conduz a uma variação da função de Liapunov discreta:

$$\Delta V_k \simeq -E^2(\mathbf{x}_k) = -V_k < 0$$

a qual decresce a cada iteração, sendo portanto o algoritmo convergente.

4.7.1 Critério de parada

Por ser ignorado *a priori* o valor ótimo da função objetivo p^*, evidentemente não se pode utilizar $r = 0$ como regra de parada do algoritmo. Conferir as condições KKT ou se um ponto é estacionário pode significar um alto custo computacional dependendo da complexidade do problema abordado[6].

Escolhe-se aqui, portanto, um outro critério de parada simples.

Por ser $\Delta V_k < 0$, sempre que a trajetória entrar no conjunto viável o fará com um valor da função objetivo não crescente, exceto no caso em que as trajetórias já tenham ultrapassado o valor mínimo dentro deste conjunto. Assim, avalia-se o valor de $f_0(\mathbf{x}_k)$ sempre que a trajetória cumpra as restrições, e se esse valor aumentar (ou for igual) significa que esse mínimo foi ultrapassado (ou se estacionou neste ponto), parando o algoritmo e escolhendo como ponto ótimo, o último ponto $\mathbf{x}_k$, calculado antes do aumento de $f_0(\mathbf{x}_k)$, dentro do conjunto viável.

Tal método é simples e eficiente, mas possui a desvantagem de desperdiçar algumas iterações depois de encontrar o ponto ótimo.

4.7.2 Implementação do algoritmo gradiente EV

Em seguida será apresentada a forma de implementar o algoritmo passo a passo.

[6]Define-se $\mathbf{x} \in \chi$ como ponto estacionário para o problema (4.13) se $\forall \mathbf{u} \in \chi : (\mathbf{u} - \mathbf{x})^T \nabla f_0(\mathbf{x}) \geq 0$

Passo 0) Escolher $\mathbf{x}_0$ e estimar γ tal que $\gamma < p^*$. Atribuir $f_0 = \infty$

Passo 1) Calcular $\sigma(\mathbf{x}_k)$

Passo 2) Se $\sigma(\mathbf{x}_k) = 1$: calcular $f_0(\mathbf{x}_k)$

se $f_0(\mathbf{x}_k) \geq f_0$: ir para passo 6

se não: $f_0 := f_0(\mathbf{x}_k), \quad \mathbf{x}^* := \mathbf{x}_k$

Passo 3) Calcular $E(\mathbf{x}_k)$ e $\nabla E(\mathbf{x}_k)$

Passo 4) Calcular α_k

Passo 5) Calcular $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla E(\mathbf{x}_k)$

voltar passo 1.

Passo 6) ponto ótimo $\mathbf{x}^*$

valor ótimo f_0

Na figura 4.6 é apresentado o diagrama de blocos representante do algoritmo gradiente EV discreto.

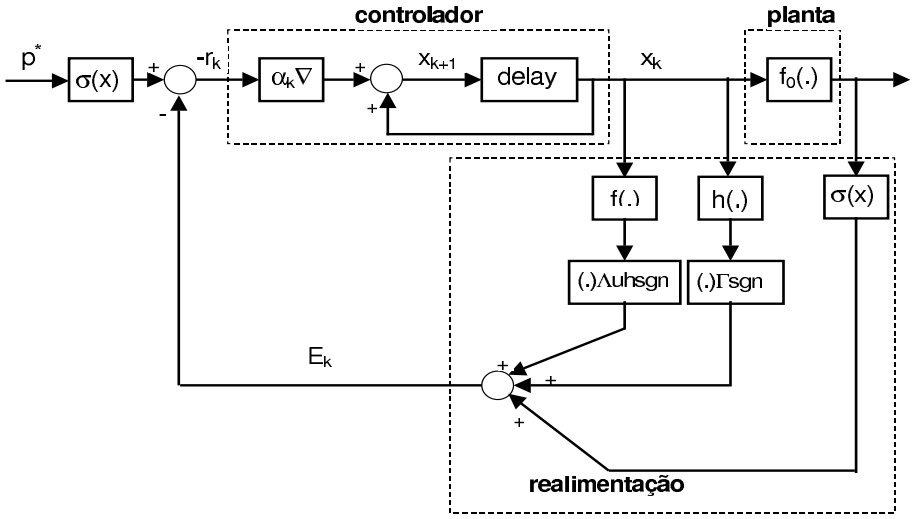

Figura 4.6: Diagrama de blocos do sistema discreto

4.8 Execução do algoritmo gradiente EV

O algoritmo foi implementado com o programa MATLAB 6, e testado com 5 problemas diferentes, descritos a seguir:

Problema 1

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 + x_2 + 1 \leq 0 \\ & (x_1 + 0.5)^2 + x_2^2 - 1 \leq 0 \\ & 2x_1 - x_2 + 1 = 0 \end{aligned}$$

Problema 2

$$\begin{aligned} \min \quad & (x_1 - 1)^2 - x_2 + 2 \\ \text{s.t.} \quad & x_1 + x_2^2 + 1 \leq 0 \\ & (x_1 + 1)^2 + x_2^2 - 1 \leq 0 \\ & \tfrac{3}{2}x_1 - x_2 + 2 = 0 \end{aligned}$$

Problema 3

$$\begin{aligned} \min \quad & (x_1 - 1)^2 - x_2 + 2 \\ \text{s.t.} \quad & x_1 + x_2^2 + 1 \leq 0 \\ & (x_1 + 1)^2 + x_2^2 - 0.5 \leq 0 \\ & -2x_1 + x_2 - 3 \leq 0 \\ & 11x_1 - 5x_2 + 13 \leq 0 \\ & -x_2 - 1 \leq 0 \end{aligned}$$

Neste último problema, a quinta condição é redundante, entretanto, o desempenho do algoritmo não é afetado.

Problema 4

Este problema é o terceiro problema de teste descrito em [64, pp. 353-354], e modeliza a geração de energia por parte de três geradores para satisfazer uma determinada demanda durante um determinado período. A função objetivo é quadrática estritamente convexa de 15 variáveis. Possui 65 restrições de desigualdade afins, sendo 36 delas condições de limite das variáveis, e nenhuma restrição de igualdade.

Problema 5

O modelo matemático que descreve este problema pode ser achado em [16], com o índice s383. A função objetivo possui 14 variáveis e está restrita por 28 condições de desigualdade (que correspondem aos limites superior e inferior das variáveis) e uma

restrição de igualdade afim.

$$\begin{array}{lll} \min & \sum_{i=1}^{14} a_i/x_i & \\ \text{s.t.} & -x_i \leq 0 & \forall i \in \{1,..,14\} \\ & x_i - 0.04 \leq 0 & \forall i \in \{1,..,5\} \\ & x_i - 0.03 \leq 0 & \forall i \in \{6,..,14\} \\ & \sum_{i=1}^{14} c_i x_i - 1 = 0 & \end{array}$$

sendo c_i e a_i constantes especificadas na página mencionada.

Os problemas foram também implementados com a função **fmincon**, do programa MATLAB 6, conferindo-se os resultados apresentados e comparados com os resultados do algoritmo gradiente EV. Em todos os casos o ponto ótimo e o valor mínimo da função objetivo restrita foram achados pelo algoritmo proposto, com comprimento do passo (4.53), evidenciando assim convergência global.

No problema 1, o ponto ótimo é $\mathbf{x}^* = [-0.6667 \ \ -0.3333]^T$ e o valor mínimo da função $p^* = 0.5556$; foi utilizado um valor estimado como mínimo da função retrita $\gamma = 0.5$.

No problema 2, o ponto ótimo é $\mathbf{x}^* = [-1.1111 \ 0.3333]^T$ e o valor mínimo da função $p^* = 6.1235$; foi utilizado um mínimo estimado $\gamma = 5$.

No problema 3, o ponto ótimo é $\mathbf{x}^* = [-1.0679 \ 0.2506]^T$ e o valor ótimo $p^* = 6.0258$, tendo sido utilizado o mesmo mínimo estimado γ do problema anterior.

No problema 4, o ponto ótimo é $\mathbf{x}^* = [8 \ 57 \ 3 \ 1 \ 64 \ 0 \ 0 \ 71 \ 0 \ 1 \ 78 \ 6 \ 3 \ 85 \ 12]^T$ e o valor ótimo $p^* = 78.6979$; foi utilizado um mínimo estimado $\gamma = 50$.

No problema 5, o ponto ótimo é
$\mathbf{x}^* = 10^{-2}[4 \ 3.32 \ 3.23 \ 3.10 \ 3.09 \ 2.78 \ 2.67 \ 2.49 \ 2.36 \ 2.21 \ 2.04 \ 1.96 \ 2.08 \ 2.58]^T$ e o valor mínimo da função restrita é $p^* = 7.293\,10^5$; foi utilizado um mínimo estimado $\gamma = 5\,10^5$.

Em todos os problemas, os ganhos λ_i e ν_i utilizados foram unitários.

As trajetórias dos três primeiros problemas são apresentados nos seguintes gráficos, onde a partir de diversos pontos iniciais, são colocados o número total de iterações, e o número de iterações até o ponto ótimo.

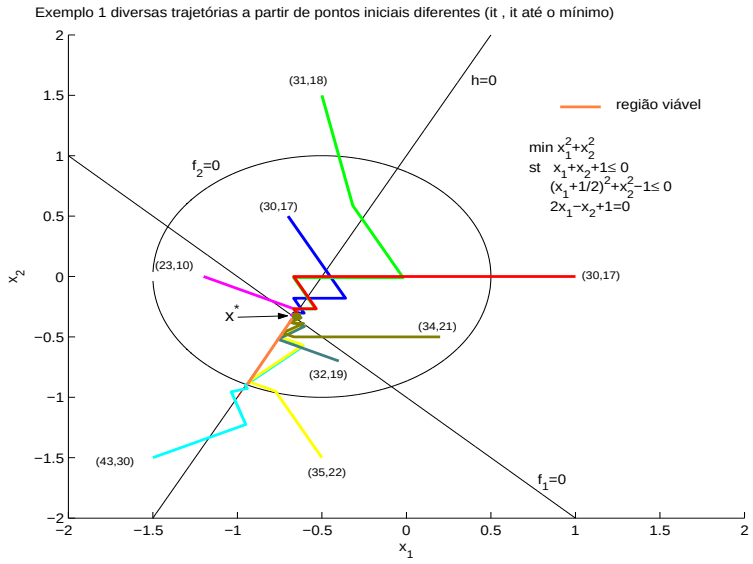

Figura 4.7: Implementação do algoritmo gradiente EV para o problema 1 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

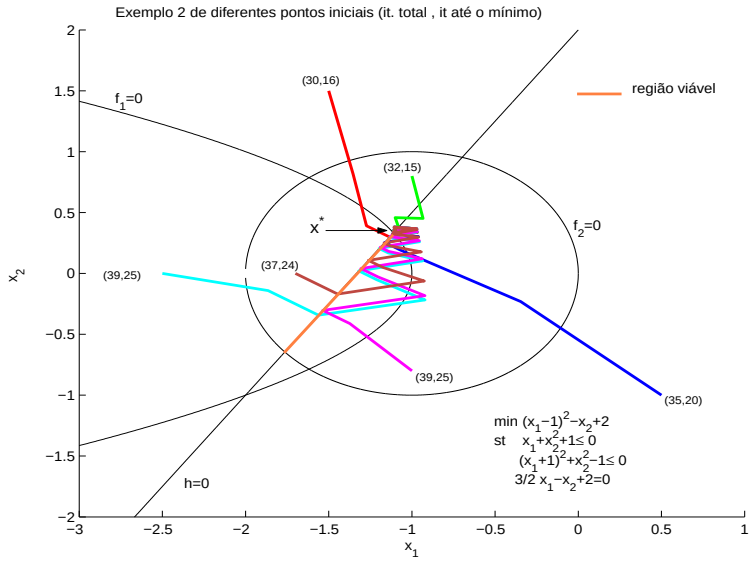

Figura 4.8: Implementação do algoritmo gradiente EV para o problema 2 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

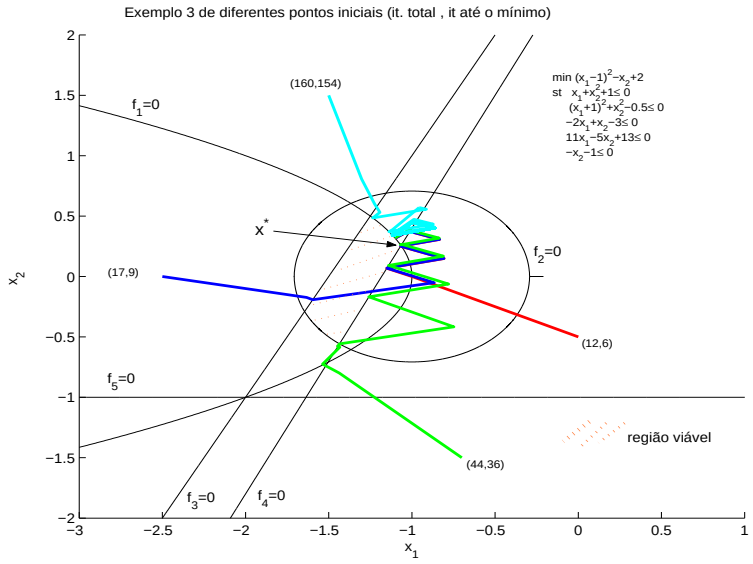

Figura 4.9: Implementação do algoritmo gradiente EV para o problema 3 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

O problema 4 foi testado a partir de dois pontos iniciais e foi medido o tempo de execução do algoritmo. O primeiro ponto inicial foi $\mathbf{x}_0 = [20\,55\,15\,20\,60\,20\,20\,60\,20\,20\,60\,20\,20\,60\,20]^T$, tendo achado o ponto ótimo em 753 iterações e concluido em 838 iterações. O tempo de execução foi 14.28 s. O segundo ponto inicial foi $\mathbf{x}_0 = [10\,50\,2\,1\,55\,0\,0\,60\,5\,5\,70\,12\,7\,70\,16]^T$, tendo achado o ponto ótimo em 282 iterações e concluido em 360 iterações. O tempo de execução foi de 6.21 s.

O problema 5 também foi testado a partir de dois pontos iniciais, e medido o tempo de execução do algoritmo. O primeiro ponto inicial foi $x_{0_i} = 0.01 \quad \forall i \in \{1, .., 14\}$, tendo achado o ponto ótimo em 7416 iterações e concluido em 8003 iterações. O tempo de execução do algoritmo foi de 63.55 s. O segundo ponto inicial foi $x_{0_i} = 0.02 \quad \forall i \in \{1, .., 14\}$, tendo achado o ponto ótimo em 6425 iterações e concluido em 7012 iterações. O tempo de execução foi de 55.75 s.

Cabe destacar que a função **fmincon** do programa MATLAB 6 não foi capaz de resolver o problema 5 a partir de nenhum dos pontos iniciais testados, apontando um valor ótimo da função igual a infinito após duas iterações.

4.9 Algoritmo de gradiente projetado

Neste novo algoritmo, quando a trajetória entra no conjunto viável, e o gradiente da função de energia faz a trajetória sair, escolhe-se como lei de atualização a projeção do vetor gradiente sobre a fronteira do conjunto viável (ou da restrição que estaria sendo violada). Sabe-se que a matriz $P_{\mathbf{y}} = I - \frac{\mathbf{y}\mathbf{y}^T}{\|\mathbf{y}\|_2^2}$ projeta um vetor sobre o hiperplano normal a $\mathbf{y}$. Assim, definindo o algoritmo de projeção

$$\text{Proj}(\nabla E(\mathbf{x}), \mathbf{y}) \begin{cases} \nabla E(\mathbf{x}) & \text{se } \sigma(\mathbf{x})(1 - \sigma(\mathbf{x} - \beta\nabla E(\mathbf{x}))) = 0 \\ P_{\mathbf{y}}\nabla E(\mathbf{x}) & \text{se } \sigma(\mathbf{x}) = 1 \text{ e } \sigma(\mathbf{x} - \beta\nabla E(\mathbf{x})) = 0 \end{cases} \tag{4.57}$$

sendo β um escalar suficientemente pequeno.

Este método de projeção foi utilizado em [45]. Assim, uma vez que a trajetória entra no conjunto viável, só se afastará deste se, no ponto de saída, a fronteira for estritamente convexa, e não mais sairá se a restrição violada for afim [7].

O algoritmo tem a seguinte propriedade: $\|\text{Proj}(\nabla E(\mathbf{x}), \mathbf{y})\| \leq \|\nabla E(\mathbf{x})\|$ (ver [45]).

Assim, chamando $\bar{\mathbf{x}}_k = \mathbf{x}_k - \beta\nabla E(\mathbf{x}_k)$, se $\sigma(\mathbf{x}_k) = 1$ e $\sigma(\bar{\mathbf{x}}_k) = 0$, então existe j tal que $f_j(\mathbf{x}_k) \leq 0$ e $f_j(\bar{\mathbf{x}}_k) > 0$, ou $h_j(\mathbf{x}_k) = 0$ e $h_j(\bar{\mathbf{x}}_k) \neq 0$. A projeção será feita sobre o gradiente desta função, e o escalar β é escolhido de maneira tal que $f_j(\bar{\mathbf{x}}_k)$ ou $h_j(\bar{\mathbf{x}}_k)$ estejam arbitrariamente próximos de zero.

Portanto:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \left(\nabla E(\mathbf{x}_k) - \sigma(\mathbf{x}_k)(1 - \sigma(\bar{\mathbf{x}}_k)) \frac{\nabla^T E(\mathbf{x}_k)\nabla E(\bar{\mathbf{x}}_k)\nabla E(\bar{\mathbf{x}}_k)}{\nabla^T E(\bar{\mathbf{x}}_k)\nabla E(\bar{\mathbf{x}}_k)} \right) \tag{4.58}$$

onde α_k é calculado como em (4.53).

Observe-se que este método de projeção é fundamentalmente diferente do método de projeção ortogonal (4.12), utilizado em diversas bibliografias. Esta projeção pode resultar de difícil avaliação, ou cara do ponto de vista computacional dependendo da forma da fronteira do conjunto viável. Em [48] afirma-se que a avaliação desta projeção só é simples no caso da fronteira ser de forma simples, por exemplo um hipercubo ou

[7] Evidentemente, esta projeção não faz sentido no caso do algoritmo contínuo, porque neste já acontece da trajetória ficar confinada ao conjunto viável uma vez dentro deste.

uma hiperesfera, para os outros casos, o problema de determinar a projeção ortogonal de um ponto $\mathbf{x}$ sobre um conjunto convexo χ é em si mesmo um problema de programação não linear que pode ser descrito como

$$\begin{aligned} \min_{\mathbf{y}} \quad & \|\mathbf{y} - \mathbf{x}\| \\ \text{s.t.} \quad & \mathbf{y} \in \chi \end{aligned}$$

Em contraste, a projeção aqui proposta consiste em apenas produtos escalares de vetores.

Em seguida, serão apresentados os gráficos das trajetórias percorridas por este algoritmo, utilizando os mesmos problemas descritos na seção anterior e a partir dos mesmos pontos iniciais, mantendo os mesmos ganhos, o comprimento do passo (4.53) e mínimos estimados. Em todos os casos o ponto ótimo e o valor mínimo da função restrita foram achados.

Nestes gráficos pode-se observar que, em geral, o número de iterações no algoritmo de gradiente projetado é menor que no caso anterior. Porém, esta característica não se verifica em todos os problemas e a partir de todos os pontos iniciais. Entretanto, a previsibilidade da trajetória (sempre beirando a fronteira do conjunto viável), faz este algoritmo atrativo.

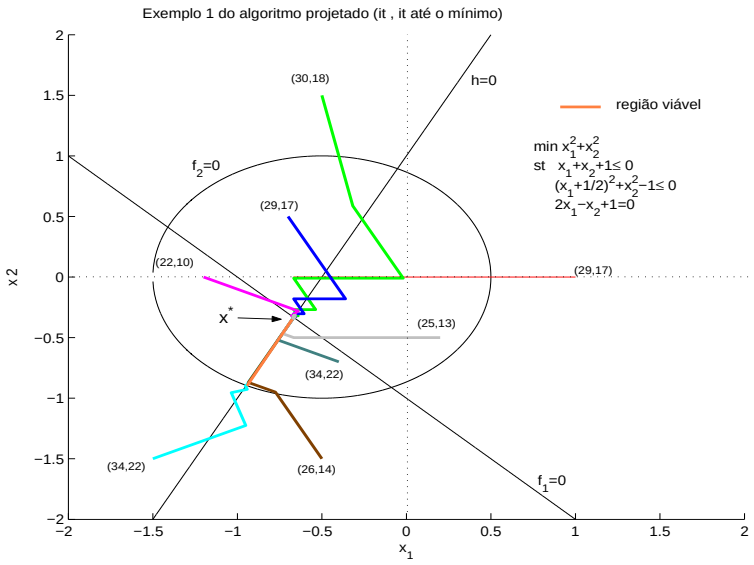

Figura 4.10: Implementação do algoritmo gradiente EV para o problema 1 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

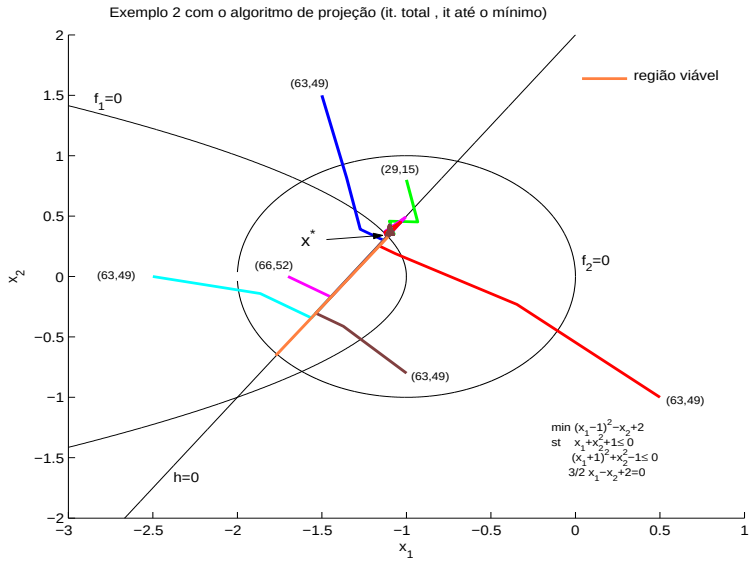

Figura 4.11: Implementação do algoritmo gradiente EV para o problema 2 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

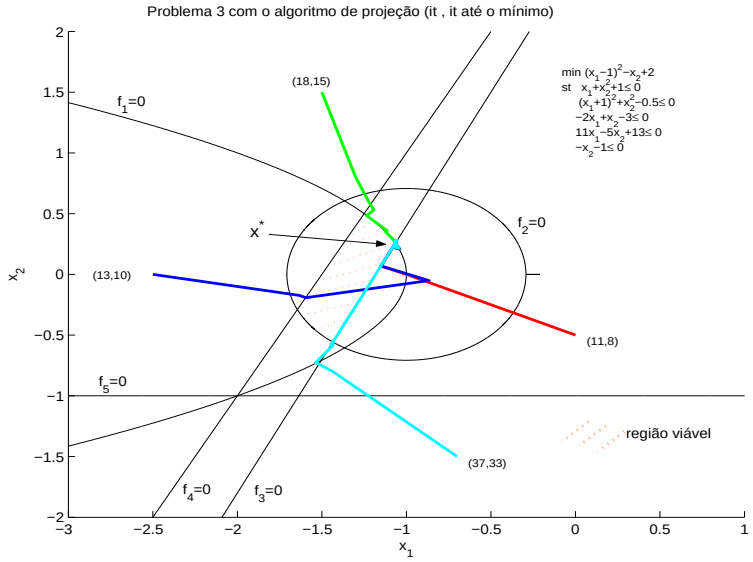

Figura 4.12: Implementação do algoritmo gradiente EV para o problema 3 a partir de diversos pontos iniciais, mostrando convergência das trajetórias

O problema 4 foi testado a partir dos mesmos dois pontos iniciais. A quantidade de iterações para o primeiro ponto inicial foi de 244 até o ponto ótimo e 289 até a parada do algoritmo. O tempo de execução foi de 15.98 s. A partir do segundo ponto inicial, o algoritmo empregou 218 iterações para achar o ponto ótimo, e 331 até se deter. O tempo de execução foi de 17.96 s.

O problema 5 foi testado a partir dos mesmos pontos iniciais também. Para o primeiro ponto inicial, o algoritmo achou o ponto ótimo em 103271 iterações, se detendo em 103880 iterações, com um tempo de execução de 2122.7 s. A partir do segundo ponto inicial, o algoritmo demorou 104708 iterações para achar o ponto ótimo e 105317 até se deter, demorando 2172.7 s para sua execução.

A seguinte tabela resume os resultados obtidos nos diferentes problemas a partir de todos os pontos iniciais testados. No caso dos problemas 1, 2 e 3, o tempo de demora dos algoritmos é um dado inconclusivo, pois diferentes execuções forneceram resultados diferentes, e em todos os casos na ordem das centésimas de segundo. Por tal razão esses dados não foram incluídos.

	$\mathbf{x}_0$	gradiente EV			gradiente projetado			fmincon			$\mathbf{x}^*$	p^*
		it.	it.m.	t[s]	it.	it.m.	t [s]	it.	n°a.f	t[s]		
1	[-1.5;1.5]	43	30	N.T.	34	22	N.T.	3	15	N.T.	[-0.67; -0.33]	0.56
	[-0.5;1.5]	35	22	N.T.	26	14	N.T.	3	15	N.T.		
	[-0.4;-0.7]	32	19	N.T.	34	22	N.T.	2	11	N.T.		
	[0.2;-0.5]	34	2	N.T.	25	13	N.T.	2	11	N.T.		
	[1;0]	30	17	N.T.	29	17	N.T.	2	11	N.T.		
	[-0.5;1.5]	31	18	N.T.	30	18	N.T.	3	15	N.T.		
	[-0.7;0.5]	30	17	N.T.	29	17	N.T.	2	11	N.T.		
	[-1.2;0]	23	10	N.T.	22	10	N.T.	2	11	N.T.		
2	[-1;-0.8]	39	25	N.T.	63	49	N.T.	6	28	N.T.	[-1.11; 0.33]	6.12
	[0.5;-1]	35	20	N.T.	63	49	N.T.	6	27	N.T.		
	[-1;0.8]	32	15	N.T.	29	15	N.T.	5	23	N.T.		
	[-1.5;1.5]	30	16	N.T.	63	49	N.T.	6	27	N.T.		
	[-1.7;0]	37	24	N.T.	66	52	N.T.	4	19	N.T.		
	[-2.5;0]	39	25	N.T.	63	49	N.T.	5	23	N.T.		
3	[-0.7;-1.5]	44	36	N.T.	37	33	N.T.	7	31	N.T.	[-1.07; 0.25]	6.02
	[0;-0.5]	12	6	N.T.	11	8	N.T.	6	27	N.T.		
	[-1.5;1.5]	160	154	N.T.	18	15	N.T.	6	27	N.T.		
	[-2.5;0]	17	9	N.T.	13	10	N.T.	6	27	N.T.		
4	$\mathbf{x}_{0_1}$	838	753	14.28	289	244	15.98	9	169	3.84	$\mathbf{x}_4^*$	78.70
	$\mathbf{x}_{0_2}$	360	282	6.21	331	218	17.96	7	135	2.14		
5	$x_{0_i} = 0.01$ $i \in \{1..14\}$	8003	7416	63.55	103880	103271	2122.7	-	-	-	$\mathbf{x}_5^*$	7.29E5
	$x_{0_i} = 0.02$ $i \in \{1..14\}$	7012	6425	55.75	105317	104708	2172.7	-	-	-		

Tabela 4.1: Resultados das implementações dos diferentes algoritmos, onde
it.=iterações, it.m.=iterações até o ponto mínimo,
t=tempo, n°a.f.=número de avaliações da função, N.T.=não testado.

$\mathbf{x}_{0_1} = [20\ 55\ 15\ 20\ 60\ 20\ 20\ 60\ 20\ 20\ 60\ 20\ 20\ 60\ 20]^T$

$\mathbf{x}_{0_2} = [10\ 50\ 2\ 1\ 55\ 0\ 0\ 60\ 5\ 5\ 70\ 12\ 7\ 70\ 16]^T$

$\mathbf{x}_4^* = [8\ 57\ 3\ 1\ 64\ 0\ 0\ 71\ 0\ 1\ 78\ 6\ 3\ 85\ 12]^T$

$\mathbf{x}_5^* = 10^{-2}[4\ 3.32\ 3.23\ 3.10\ 3.09\ 2.78\ 2.67\ 2.49\ 2.36\ 2.21\ 2.04\ 1.96\ 2.08\ 2.58]^T$

4.10 Conclusões deste capítulo

Os algoritmos apresentados demonstraram ser eficientes na busca do valor mínimo de uma função convexa (ou estritamente quase-convexa dentro do conjunto viável) restrita por condições de desigualdade convexas e de igualdade afins, a partir de qualquer ponto inicial, achando este valor assim como o ponto ótimo nos cinco problemas apresentados a modo de exemplo.

Porém, algumas observações e propostas de trabalho futuro merecem ser mencionadas.

1) Os problemas também foram implementados utilizando a função **fmincon** do programa MATLAB 6 e a partir dos mesmos pontos iniciais. Esta função utiliza um método SQP, quase-Newton, busca em linha. O número de iterações nos quatro primeiros problemas apresentados foi muito menor que os dos nossos algoritmos discretos (4.50) e (4.58) (o maior número de iterações foi 9, no problema 4 a partir do primeiro ponto inicial), assim como o tempo de execução no caso do problema 4. Porém, cada iteração de um método SQP realiza cálculos envolvendo estimativas do jacobiano e do hessiano. Estes cálculos requerem um número de operações de ponto flutuante que cresce como o cubo do número de variáveis. Os algoritmos discretos propostos (4.50) e (4.58), com comprimento de passo (4.53), por outro lado, utilizam apenas o cálculo de gradientes e produtos escalares. Desta forma, espera-se que, para problemas maiores (i.e., com maior número de variáveis), haja ganhos dos algoritmos propostos em relação a algoritmos que utilizem informação de jacobianos e hessianos, ainda que estes realizem um número reduzido de iterações. Esclarecemos que o comprimento do passo alternativo (4.56), proposto quando não é possível estimar o mínimo γ, e que não foi utilizado nas implementações dos algoritmos, também exige a utilização do hessiano da função objetivo. Além disso, como já foi apontado, a função **fmincon** mostrou-se incapaz de achar o valor ótimo no caso do quinto problema apresentado, ao passo que os algoritmos (4.50) e (4.58), embora empregando um número considerável de iterações e tempo de execução, achou estos valores a partir dos pontos iniciais

testados.

2) Mostrou-se a necessidade de implementar um critério de parada para os algoritmos discretos mais eficiente, pois o utilizado realiza algumas iterações a mais.

3) No caso da função objetivo não ser convexa, e apresentar mínimos locais e um mínimo global, existe a possibilidade, não pesquisada no presente trabalho, de utilizar outras funções diferentes do gradiente conjugado, mas seguindo a filosofia de implementação de um sistema de controle em malha fechada a estrutura variável. Alguns destes algoritmos a serem implementados podem ser o HBF (*heavy ball with friction*) e o DIN, apresentados em [3].

4) Mostrou-se também a necessidade de realizar uma avaliação quantitativa dos erros produzidos pelos algoritmos (4.50) e (4.58), isto é, quantificar o erro na determinação do ponto ótimo achado e o nível de satisfação das restrições neste ponto.

Dadas estas considerações, achamos que o conceito de chaveamento pode render futuras pesquisas que resultem em algoritmos mais eficientes.

Capítulo 5

Algoritmo para desigualdades variacionais baseado em sistema dinâmico gradiente projetado utilizando uma função de Liapunov de controle

5.1 Introdução

O problema de desigualdades variacionais tem muitas aplicações importantes em uma ampla variedade de campos da ciência e da engenharia, os quais incluem economia de redes, engenharia de transportes, teoria de jogos, controle, processamento de sinais, sistemas de identificação, projetos de filtros, análise de regressão, projetos de estruturas, projetos mecânicos, planeamento de redes elétricas, entre outros (ver [26], [30] e suas referências). Também problemas de otimização tais como minimização de funções convexas restritas a um domínio convexo ou não, programação linear e quadrática, programação não linear, assim como problemas de minimax podem ser formulados como desigualdades variacionais.

No capítulo referente a otimização convexa, foi apresentado um algoritmo contínuo destinado a resolver problemas de otimização convexa baseado em um sistema dinâmico

gradiente a estrutura variável projetado utilizando uma função de Liapunov de controle (CLF). Tal algoritmo é discretizado e seu comprimento do passo otimizado utilizando controle ótimo de Liapunov (LOC).

Algoritmos específicos para a resolução de problemas de desigualdade variacional são menos abundantes na bibliografia. Porém, dentre os apresentados, podemos mencionar como principais inconvenientes encontrados:

1) A utilização da projeção ortogonal, a qual é de difícil avaliação exceto se a fronteira do conjunto viável tiver uma forma simples (por exemplo, um hipercubo ou uma hiperesfera [48, p. 41]). Esta projeção é utilizada em [38], [37], [74], [83], [80], [82], [65], [71], [81] e [46].

2) A exigência do ponto inicial estar dentro do conjunto viável, o qual, dependendo do problema, pode ser de costosa resolução do ponto de vista computacional ([38], [74], [37], [7]).

3) Limitações com respeito às restrições que determinam o conjunto viável ou à função objetivo. Por exemplo, [53] considera apenas a existência de restrições de desigualdade, [81] considera como conjunto viável um hipercubo, assim como [31] considera restrições de igualdade dentro do ortante não negativo. Em [46] é exigida uma função objetivo pseudomonôtona no conjunto viável, em [30], [37], [75], [80], [85], [31], [81] e [7] que a função objetivo seja monôtona em todo seu domínio ou no conjunto viável, em [38] e [83], que seja fracamente co-coerciva.

4) No caso dos algoritmos contínuos, não são apresentadas as discretizações destes ([30], [53], [81], [46]).

5) No caso dos algoritmos discretos, a escolha do comprimento de passo é justificada experimentalmente, sem qualquer prova da sua qualidade e vantagens com respeito a outras escolhas ([38], [74], [85], [7], [83], [37], [71], [75], [31]).

6) O incremento da dimensão do problema. Em [30] e [53] os algoritmos possuem uma dimensão igual ao número de variáveis mais o número de restrições que conformam o conjunto viável; em [37], a dimensão do problema incrementa-se com o número de restrições de igualdade; em [31], é de duas vezes o número de variáveis mais o número de restrições.

Seguindo idêntica metodologia utilizada no capítulo de otimização convexa, apresenta-

se aqui um novo algoritmo contínuo para resolver problemas de desigualdades variacionais baseado em um sistema de controle em malha fechada a estrutura variável, o qual apresenta uma trajetória em modo deslizante ao longo da fronteira do conjunto viável, e não mostra os inconvenientes apontados. A lei de atualização das variáveis é projetada utilizando uma função de Liapunov de controle (CLF). Este algoritmo é discretizado otimizando o comprimento de passo segundo a teoria de controle ótimo de Liapunov (LOC), resultando em um algoritmo iterativo simples de implementar independentemente da dificuldade do problema. Esta abordagem contrasta com aquela baseada na escolha do comprimento do passo de iteração por secionamento do domínio, cuja dificuldade é proporcional à dimensão do problema (número de variáveis, número de restrições, etc.).

O capítulo está organizado como segue. Na seção 2 são apresentados preliminares necessários para o entendimento do problema. Na seção 3 é abordada a descrição do problema. Na seção 4 são apresentados os problemas de teste que serão utilizados. Na seção 5 são analisados outros algoritmos apresentados na bibliografia. Na seção 6 é desenvolvido o algoritmo contínuo para a resolução do problema de desigualdades variacionais. Na seção 7 o algoritmo contínuo é simulado. Na seção 8 o algoritmo é discretizado e sua convergência analisada. Na seção 9 é apresentada a versão projetada do algoritmo discreto. Na seção 10 os algoritmos são testados. Na seção 11 é discretizado o algoritmo apresentado em [30] com comprimento de passo otimizado por LOC. Na seção 12 outros algoritmos discretos são apresentados sem qualquer prova de convergência. Finalmente, na seção 13 são apresentadas as conclusões deste capítulo.

5.2 Preliminares

Nesta seção serão apresentadas diversas definições, lemas e teoremas necessários para o entendimento do problema.

Definição 5.2.1 *Ponto fixo ([50, p. 7]).*

Muitos problemas de análise não linear podem ser resolvidos por meio de teoremas de ponto fixo. Dado um mapeamento de um conjunto $\Omega \subseteq \mathbb{R}^n$ *nele mesmo,* $\mathbf{f} : \Omega \to \Omega$,

um ponto $\mathbf{x} \in \Omega$ *é chamado de ponto fixo de* $\mathbf{f}$ *se*

$$\mathbf{f}(\mathbf{x}) = \mathbf{x} \tag{5.1}$$

Definição 5.2.2 *Um mapeamento* $\mathbf{f} : \Omega \to \Omega$ *é um mapeamento de contração se*

$$\forall \mathbf{x}, \mathbf{y} \in \Omega, \quad 0 \leq \alpha < 1 : \qquad \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\| \leq \alpha \|\mathbf{x} - \mathbf{y}\|$$

Quando $\alpha = 1$ *o mapeamento chama-se não expansivo.*

Definição 5.2.3 *Um mapeamento* $\mathbf{f}(\mathbf{x}) : \Omega \to \Omega$ *é Lipschitz contínuo se existe uma constante* $L > 0$ *tal que*

$$\forall \mathbf{x},\ \mathbf{y} \in \Omega \quad \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|$$

Teorema 5.2.4 *Seja* S *espaço métrico completo e* $\mathbf{f} : S \to S$ *um mapeamento de contração, então existe um único ponto fixo de* $\mathbf{f}$ *(ver prova em [50, p. 8]).*

Teorema 5.2.5 *Teorema de Brouwer.*

Seja $\mathbf{f}$ *um mapeamento contínuo sobre um conjunto* $\Omega \subset \mathbb{R}^n$ *convexo compacto, nele mesmo, então* $\mathbf{f} : \Omega \to \Omega$ *admite pelo menos um ponto fixo (ver prova em [50, p. 8]).*

Definição 5.2.6 *Projeção ortogonal sobre um conjunto convexo.*

Seja Ω *um subconjunto fechado e convexo de um espaço de Hilbert real* H*, para todo* $\mathbf{x} \in H$*, define-se como a projeção ortogonal de* $\mathbf{x}$ *em* Ω*:*

$$\mathrm{Pr}_{\Omega}[\mathbf{x}] = \arg\min_{\mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\| \qquad \forall \mathbf{y} \in \Omega \tag{5.2}$$

Lema 5.2.7 *Seja* Ω *um subconjunto fechado e convexo de um espaço de Hilbert real* H*, então para cada* $\mathbf{x} \in H$ *existe um único ponto* $\mathbf{y} \in \Omega$ *tal que* $\mathbf{y} = \mathrm{Pr}_{\Omega}[\mathbf{x}]$ *(ver prova em ([50, p. 8] e [26, p. 77]).*

Note-se que para todo $\mathbf{x} \in \Omega$: $\mathrm{Pr}_{\Omega}[\mathbf{x}] = \mathbf{x}$.

Teorema 5.2.8 *Seja Ω um subconjunto fechado e convexo de um espaço de Hilbert H, então $\mathbf{y} = \mathrm{Pr}_\Omega[\mathbf{x}]$ se e somente se*

$$\forall \mathbf{x}' \in \Omega : \quad \mathbf{y}^T(\mathbf{x}' - \mathbf{y}) \geq \mathbf{x}^T(\mathbf{x}' - \mathbf{y})$$

(ver prova em [50, pp. 9-10] e [26, p. 77]).

Corolário 5.2.9 *Para todo $\mathbf{x} \in H$*

$$(\mathbf{x} - \mathrm{Pr}_\Omega[\mathbf{x}])^T(\mathbf{y} - \mathrm{Pr}_\Omega[\mathbf{x}]) \leq 0 \quad \forall \mathbf{y} \in \Omega$$

A prova é imediata a partir do teorema (5.2.8).

Corolário 5.2.10 *O operador Pr_Ω é não expansivo, isto é*

$$\forall \mathbf{x},\ \mathbf{x}' \in H : \quad \|\mathrm{Pr}_\Omega[\mathbf{x}] - \mathrm{Pr}_\Omega[\mathbf{x}']\| \leq \|\mathbf{x} - \mathbf{x}'\|$$

Prova ([50, p. 10], [26, p. 77]):
Sejam $\mathbf{y} = \mathrm{Pr}_\Omega[\mathbf{x}]$ e $\mathbf{y}' = \mathrm{Pr}_\Omega[\mathbf{x}']$, então, por teorema (5.2.8)

$$\mathbf{y}^T(\mathbf{z} - \mathbf{y}) \geq \mathbf{x}^T(\mathbf{z} - \mathbf{y}) \quad \forall \mathbf{z} \in \Omega$$
$$\mathbf{y}'^T(\mathbf{z} - \mathbf{y}') \geq \mathbf{x}'^T(\mathbf{z} - \mathbf{y}') \quad \forall \mathbf{z} \in \Omega$$

Escolhendo $\mathbf{z} = \mathbf{y}'$ na primeira desigualdade e $\mathbf{z} = \mathbf{y}$ na segunda desigualdade e somando ambas, obtemos:

$$\|\mathbf{y} - \mathbf{y}'\|^2 \leq (\mathbf{x} - \mathbf{x}')^T(\mathbf{y} - \mathbf{y}') \leq \|\mathbf{x} - \mathbf{x}'\|\|\mathbf{y} - \mathbf{y}'\|$$
$$\Rightarrow \quad \|\mathbf{y} - \mathbf{y}'\| \leq \|\mathbf{x} - \mathbf{x}'\|$$

Portanto $\mathrm{Pr}_\Omega[\mathbf{x}]$ é um operador globalmente Lipschitz contínuo sobre $\mathbb{R}^n$.

Mais ainda, [83, Lema 2.2] afirma, sem demonstração

$$\forall \mathbf{x},\ \mathbf{x}' \in H : \quad \|\mathrm{Pr}_\Omega[\mathbf{x}] - \mathrm{Pr}_\Omega[\mathbf{x}']\|^2 \leq \|\mathbf{x} - \mathbf{x}'\|^2 - \|(\mathrm{Pr}_\Omega[\mathbf{x}] - \mathbf{x}) - (\mathrm{Pr}_\Omega[\mathbf{x}'] - \mathbf{x}')\|^2 \quad (5.3)$$

Lema 5.2.11 *A função*

$$\rho(\mathbf{x}) = \frac{1}{2}\|\mathbf{x} - \mathrm{Pr}_\Omega[\mathbf{x}]\|_2^2 \quad \forall \mathbf{x} \in \mathbb{R}^n$$

é continuamente diferenciável em $\mathbf{x}$*. Mais ainda*

$$\nabla\rho(\mathbf{x}) = \mathbf{x} - \mathrm{Pr}_\Omega[\mathbf{x}]$$

(ver prova em [26, p. 78]).

O operador de projeção ortogonal nem sempre pode ser calculado analiticamente, ou seu cálculo pode resultar costoso do ponto de vista computacional. Tal cálculo apenas é simples quando $\mathbf{x} \in \Omega$ ou quando a fronteira do conjunto convexo Ω tem uma forma simples, por exemplo um hipercubo ou uma hiperesfera ([48]). Por exemplo, para o caso $\Omega = \{\mathbf{x} \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i,\ \forall i = 1, .., n\}$:

$$\mathrm{Pr}_\Omega[x_i] = \begin{cases} l_i, & x_i < l_i \\ x_i, & l_i \leq x_i \leq u_i \\ u_i, & x_i > u_i \end{cases} \tag{5.4}$$

Note-se que u_i poderia ser $+\infty$ e l_i poderia ser $-\infty$. Para o caso $\Omega = \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{c}\| \leq r,\ r > 0\}$, onde $\mathbf{c} \in \mathbb{R}^n$ e $r \in \mathbb{R}$ são duas constantes:

$$\mathrm{Pr}_\Omega[\mathbf{x}] = \begin{cases} \mathbf{x}, & \|\mathbf{x} - \mathbf{c}\| \leq r \\ \mathbf{c} + \frac{\mathbf{x}-\mathbf{c}}{\|\mathbf{x}-\mathbf{c}\|} r, & \|\mathbf{x} - \mathbf{c}\| > r \end{cases} \tag{5.5}$$

Para os outros casos, o problema de determinar a projeção ortogonal de um ponto $\mathbf{x} \in H$ sobre um conjunto convexo $\Omega \subseteq H$ é em si mesmo um problema de programação não linear que pode ser descrito como

$$\begin{aligned} \min_{\mathbf{y}} \quad & \|\mathbf{y} - \mathbf{x}\| \\ \text{s.t.} \quad & \mathbf{y} \in \Omega \end{aligned}$$

5.3 Descrição do problema

Dado um conjunto $\chi \subseteq \mathbb{R}^n$ convexo, chamado de conjunto viável e definido por

$$\chi = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{g}(\mathbf{x}) \preceq \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \tag{5.6}$$

onde $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), .., g_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $\forall i \in \{1, .., m\} : g_i(\mathbf{x})$ convexa, contínua com derivadas parciais contínuas, e $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), .., h_p(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^p$ tal que $\forall i \in \{1, .., p\} : h_i(\mathbf{x})$ afim, contínua, com derivadas parciais contínuas. Seja uma função $\mathbf{f} : \chi \to \mathbb{R}^n$ um mapeamento Lipschitz contínuo, chamada função objetivo, define-se como o problema de desigualdade variacional VI($\mathbf{f}, \chi$), a encontrar um ponto $\mathbf{x}^* \in \chi$ tal que

$$(\mathbf{x} - \mathbf{x}^*)^T \mathbf{f}(\mathbf{x}^*) \geq \mathbf{0} \quad \forall \mathbf{x} \in \chi \tag{5.7}$$

O problema (5.7) nem sempre admite solução. Por exemplo, se $\chi = \mathbb{R}$, $f(x)(y-x) \geq 0$ para todo $y \in \mathbb{R}$, não admite solução para $f(x) = e^x$.

Teorema 5.3.1 *Seja $\chi \subset \mathbb{R}^n$ convexo fechado e $\mathbf{f} : \chi \to \mathbb{R}^n$ contínuo, uma condição necessária e suficiente para a existência de solução do problema (5.7) é que exista $R > 0$ tal que uma solução $\mathbf{x}_R \in \chi \cap B(\mathbf{0}, R)$ (bola fechada de centro $\mathbf{0} \in \mathbb{R}^n$ e raio R) de (5.7) satisfaça:*

$$\|\mathbf{x}_R\| < R$$

(ver prova em [50, p. 13]).

A partir deste teorema podem ser derivadas muitas condições suficientes de existência. Por exemplo

Corolário 5.3.2 *Se existe $\mathbf{f} : \chi \to \mathbb{R}^n$ que satisfaz*

$$\frac{(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_0))^T (\mathbf{x} - \mathbf{x}_0)}{\|\mathbf{x} - \mathbf{x}_0\|} \to \infty \ \text{ quando } \ \|\mathbf{x}\| \to \infty \quad \forall \mathbf{x} \in \chi$$

para algum $\mathbf{x}_0 \in \chi$, então existe solução de (5.7).

A solução de um problema de desigualdade variacional em geral não é única. Chamaremos ao conjunto solução de χ^*. Porém, existe uma condição muito natural

para garantir unicidade. Suponha-se que $\mathbf{x}_1^*$, $\mathbf{x}_2^* \in \chi^*$ são duas soluções do problema (5.7). Portanto

$$\mathbf{f}(\mathbf{x}_1^*)^T(\mathbf{y} - \mathbf{x}_1^*) \geq \mathbf{0} \quad \forall \mathbf{y} \in \chi$$
$$\mathbf{f}(\mathbf{x}_2^*)^T(\mathbf{y} - \mathbf{x}_2^*) \geq \mathbf{0} \quad \forall \mathbf{y} \in \chi$$

Escolhendo $\mathbf{y} = \mathbf{x}_2^*$ na primeira desigualdade e $\mathbf{y} = \mathbf{x}_1^*$ na segunda desigualdade e somando ambas, obtemos:

$$(\mathbf{f}(\mathbf{x}_1^*) - \mathbf{f}(\mathbf{x}_2^*))^T(\mathbf{x}_1^* - \mathbf{x}_2^*) \leq \mathbf{0}$$

Portanto uma condição natural suficiente para garantir unicidade seria ([50, p. 14])

$$(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}'))^T(\mathbf{x} - \mathbf{x}') > 0 \quad \forall \mathbf{x}, \mathbf{x}' \in \chi \text{ e } \mathbf{x} \neq \mathbf{x}' \tag{5.8}$$

Definição 5.3.3 *([50, p. 15]) Um mapeamento* $\mathbf{f}(\mathbf{x}) : \chi \to \mathbb{R}^n$ *é* monôtono *em* χ *se e somente se*

$$\forall \mathbf{x}, \mathbf{x}' \in \chi \quad (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}'))^T(\mathbf{x} - \mathbf{x}') \geq 0 \tag{5.9}$$

O mapeamento é estritamente monôtono *em* χ *se a igualdade se mantém apenas para* $\mathbf{x} = \mathbf{x}'$*, isto é, quando (5.8) é válida, e é* fortemente monôtono *se existe uma constante* $\beta > 0$ *tal que*

$$\forall \mathbf{x}, \mathbf{x}' \in \chi \quad (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}'))^T(\mathbf{x} - \mathbf{x}') \geq \beta\|\mathbf{x} - \mathbf{x}'\|^2 \tag{5.10}$$

Claramente, as definições estão ordenadas do condicionamento mais fraco ao mais restrito, isto é, monotonicidade forte implica monotonicidade estrita que por sua vez implica monotonicidade.

Em [30] os autores esclarecem que a definição acima refere-se a mapeamentos (estritamente, fortemente) monôtonos em χ, mas um mapeamento poderia ser (estritamente, fortemente) monôtono apenas para um ponto do conjunto viável, isto é, se para algum $\mathbf{y} \in \chi$ tal que

$$\forall \mathbf{x} \in \chi \text{ e } \mathbf{x} \neq \mathbf{y} \quad (\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq 0$$

então o mapeamento é monôtono (estritamente para o caso da desigualdade ser estrita,

fortemente para o caso da expressão do lado esquerdo da desigualdade ser maior ou igual a $\beta\|\mathbf{x}-\mathbf{y}\|^2$ para algum $\beta > 0$) apenas com respeito a um ponto $\mathbf{y} \in \chi$. Que um mapeamento $\mathbf{f} : \chi \to \mathbb{R}^n$ seja (estritamente, fortemente) monôtono para um ponto $\mathbf{y} \in \chi$ não implica que este seja (estritamente, fortemente) monôtono para todo o conjunto convexo χ.

Definição 5.3.4 *Um mapeamento* $\mathbf{f}(\mathbf{x}) : \chi \to \mathbb{R}^n$ *é* pseudomonôtono *em* χ *se ([74])*

$$\forall \mathbf{x},\ \mathbf{y} \in \chi,\ \mathbf{x} \neq \mathbf{y} \quad \mathbf{f}(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geq 0 \quad \Rightarrow \quad \mathbf{f}(\mathbf{x})^T(\mathbf{x}-\mathbf{y}) \geq 0 \tag{5.11}$$

O mapeamento é estritamente pseudomonôtono *em* χ *se ([46])*

$$\forall \mathbf{x},\ \mathbf{y} \in \chi,\ \mathbf{x} \neq \mathbf{y} \quad \mathbf{f}(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geq 0 \quad \Rightarrow \quad \mathbf{f}(\mathbf{x})^T(\mathbf{x}-\mathbf{y}) > 0 \tag{5.12}$$

O mapeamento é fortemente pseudomonôtono *em* χ *se existe* $\beta > 0$ *tal que ([46])*

$$\forall \mathbf{x},\ \mathbf{y} \in \chi,\ \mathbf{x} \neq \mathbf{y} \quad \mathbf{f}(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geq 0 \quad \Rightarrow \quad \mathbf{f}(\mathbf{x})^T(\mathbf{x}-\mathbf{y}) > \beta\|\mathbf{x}-\mathbf{y}\|^2 \tag{5.13}$$

Aqui também as definições estão ordenadas do condicionamento mais fraco ao mais forte.

Observe-se que monotonicidade (estrita, forte) em χ implica pseudomonotonicidade (estrita, forte) em χ, sendo portanto estas últimas condições mais fracas.

Lema 5.3.5 *Seja* $\mathbf{f}(\mathbf{x}) : \chi \to \mathbb{R}^n$ *diferenciável, então*

(a) $\mathbf{f}(\mathbf{x})$ *é monôtona em* χ *se e somente se* $D_{\mathbf{f}}(\mathbf{x}) \succeq 0$*, para todo* $\mathbf{x} \in \chi$*.*

(b) $\mathbf{f}(\mathbf{x})$ *é estritamente monôtona em* χ *se e somente se* $D_{\mathbf{f}}(\mathbf{x}) \succ 0$*, para todo* $\mathbf{x} \in \chi$*.*

(c) $\mathbf{f}(\mathbf{x})$ *é fortemente monôtona em* χ *se e somente se* $D_{\mathbf{f}}(\mathbf{x})$ *é uniformemente positiva definida, isto é, existe uma constante* $\beta > 0$ *tal que, para todo* $\mathbf{x} \in \chi$*:*

$$\mathbf{y}^T D_{\mathbf{f}}(\mathbf{x})\mathbf{y} \geq \beta\|\mathbf{y}\|^2 \quad \forall \mathbf{y} \in \mathbb{R}^n$$

(ver prova em [26, p. 156]).

Seja $\mathbf{x}^*$ um elemento do conjunto solução de VI$(\mathbf{f}(\mathbf{x}), \chi)$, uma condição do mapeamento $\mathbf{f}(\mathbf{x})$ que será exigida posteriormente é

$$\forall \mathbf{x} \in \chi \backslash \chi^* \quad \mathbf{f}(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^*) > 0 \tag{5.14}$$

A condição (5.14) evidentemente é observada se o mapeamento for estritamente monôtono com respeito a $\mathbf{x}^*$. Mais ainda, Solodov ([74] e [71]) nota que a condição se observa se o mapeamento for estritamente pseudomonôtono, e que inclusive não é difícil achar exemplos onde (5.14) é observada mesmo para mapeamentos nem estritamente pseudomonôtonos nem estritamente monôtonos, sendo portanto esta condição mais fraca que as apontadas.

A seguinte figura ilustra a condição (5.14)

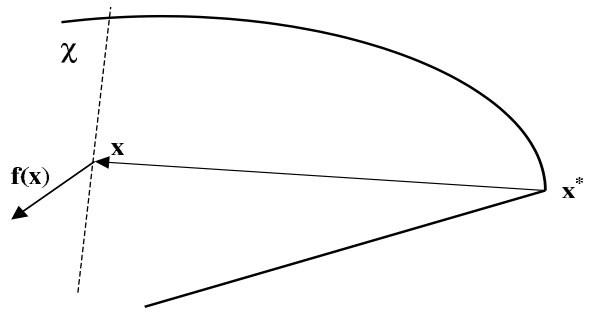

Figura 5.1: Representação da condição (5.14) exigida

Lema 5.3.6 *Seja* $\mathbf{f} : \chi_1 \to \mathbb{R}^n$ *um mapeamento contínuo estritamente monôtono de um conjunto fechado convexo* $\chi_1 \subset \mathbb{R}^n$. *Seja* $\chi_2 \subset \chi_1$ *fechado convexo. Suponha-se que existem soluções dos problemas*

$$\exists \mathbf{x}_j \in \chi_j \text{ tal que } \quad \mathbf{f}^T(\mathbf{x}_j)(\mathbf{y} - \mathbf{x}_j) \geq 0 \quad \forall \mathbf{y} \in \chi_j, \;\; j = 1, 2$$

(i) Se $\mathbf{f}(\mathbf{x}_2) = \mathbf{0}$, *então* $\mathbf{x}_1 = \mathbf{x}_2$.
(ii) Se $\mathbf{f}(\mathbf{x}_2) \neq \mathbf{0}$ *e* $\mathbf{x}_1 \neq \mathbf{x}_2$, *então o hiperplano* $\mathbf{f}^T(\mathbf{x}_2)(\mathbf{y} - \mathbf{x}_2) = 0$ *separa* $\mathbf{x}_1$ *de* χ_2.

Ver prova em [50, prop. 4.6].

5.3.1 Função de erro

A resolução do problema (5.7), VI($\mathbf{f}, \chi$), supondo a existência única de solução, pode ser descrito também como o problema de encontrar $\mathbf{x}^* \in \chi$ tal que

$$\mathbf{x}^* = \mathrm{Pr}_\chi[\mathbf{x}^* - \mathbf{f}(\mathbf{x}^*)] \tag{5.15}$$

ou, equivalentemente, a achar o ponto zero da função de resíduo ou erro definida em [53], [54] e [72], entre outros, como

$$\mathbf{e}(\mathbf{x}) := \mathbf{x} - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})] \tag{5.16}$$

Em [38] e [74] esse erro é definido como

$$\mathbf{e}(\mathbf{x}, \beta) := \mathbf{x} - \mathrm{Pr}_\chi[\mathbf{x} - \beta\mathbf{f}(\mathbf{x})] \tag{5.17}$$

para algum $\beta > 0$, sendo portanto $\mathbf{e}(\mathbf{x}) = \mathbf{e}(\mathbf{x}, 1)$. Este resíduo ou erro pode ser visto como uma distância entre um ponto $\mathbf{x}$ e a solução única do problema $\mathbf{x}^*$. Por tal motivo, podem ser utilizados como critérios de parada de um algoritmo contínuo destinado a resolver (5.7)

$$\|\mathbf{e}(\mathbf{x})\|_\infty \leq \epsilon \quad \text{ou} \quad \frac{\|\mathbf{e}(\mathbf{x})\|_\infty}{\|\mathbf{e}(\mathbf{x}(0))\|_\infty} \leq \epsilon$$

para alguma constante $\epsilon > 0$ arbitrariamente pequena. Em [54, Lema 3] e [26, p. 78], demonstra-se que o erro assim definido é Lipschitz contínuo no conjunto viável se também o for a função objetivo.

5.3.2 Propriedades da função de erro

As seguintes 4 desigualdades foram adaptadas de [53].

a) Desde que $\mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})] \in \chi \Rightarrow$

$$\mathbf{f}^T(\mathbf{x}^*)(\mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})] - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \tag{5.18}$$

b) A partir do corolário (5.2.9):

$$\begin{aligned}
&(\mathbf{x} - \mathrm{Pr}_\chi[\mathbf{x}])^T(\mathbf{y} - \mathrm{Pr}_\chi[\mathbf{x}]) \leq 0 \quad \forall \mathbf{y} \in \chi \\
&\Rightarrow \ (\mathbf{x} - \mathbf{f}(\mathbf{x}) - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})])^T(\mathbf{x}^* - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})]) \leq 0 \\
&\Rightarrow \ (\mathbf{e}(\mathbf{x}) - \mathbf{f}(\mathbf{x}))^T(\mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})] - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n
\end{aligned} \tag{5.19}$$

c) A partir de (5.9), considerando $\mathbf{f}(\mathbf{x})$ monôtono com respeito a $\mathbf{x}^*$:

$$\left\{\mathbf{f}(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right]) - \mathbf{f}(\mathbf{x}^*)\right\}^T \left(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right] - \mathbf{x}^*\right) \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n \tag{5.20}$$

d) Definindo:

$$\mathbf{d}(\mathbf{x}) := \mathbf{e}(\mathbf{x}) - \left\{\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right])\right\}$$

e somando (5.18), (5.19) e (5.20):

$$\begin{aligned}
&\Rightarrow \ \left\{\mathbf{f}(\mathbf{x}^*) + \mathbf{e}(\mathbf{x}) - \mathbf{f}(\mathbf{x}) + \left(\mathbf{f}(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right]) - \mathbf{f}(\mathbf{x}^*)\right)\right\}^T \left(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right] - \mathbf{x}^*\right) \geq 0 \\
&\Rightarrow \ \mathbf{d}(\mathbf{x})^T(\mathrm{Pr}_\chi\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right] - \mathbf{x}^*) \geq 0 \\
&\Rightarrow \ \mathbf{d}(\mathbf{x})^T(\mathbf{x} - \mathbf{x}^* - \mathbf{e}(\mathbf{x})) \geq 0
\end{aligned} \tag{5.21}$$

As seguintes desigualdades serão apresentadas sem demonstração.

e) $\forall \mathbf{x} \in \chi$, $\forall i \in \{1, .., m\}$ tal que $\nabla g_i(\mathbf{x} - \mathbf{f}(\mathbf{x})) \neq \mathbf{0}$, $\forall j \in \{1, .., p\}$ tal que $\nabla h_j(\mathbf{x} - \mathbf{f}(\mathbf{x})) \neq \mathbf{0}$,

$$\|\mathbf{x} - \mathbf{f}(\mathbf{x}) - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})]\| \geq \frac{g_i(\mathbf{x} - \mathbf{f}(\mathbf{x}))}{\|\nabla g_i(\mathbf{x} - \mathbf{f}(\mathbf{x}))\|} \tag{5.22}$$

$$\text{e} \quad \|\mathbf{x} - \mathbf{f}(\mathbf{x}) - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})]\| \geq \frac{|h_j(\mathbf{x} - \mathbf{f}(\mathbf{x}))|}{\|\nabla h_j(\mathbf{x} - \mathbf{f}(\mathbf{x}))\|} \tag{5.23}$$

Observe-se que as desigualdades são válidas tanto para as restrições violadas como para aquelas observadas, independentemente do ponto $\mathbf{x} - \mathbf{f}(\mathbf{x})$ estar ou não no conjunto viável. No caso de $g_i(\mathbf{x} - \mathbf{f}(\mathbf{x}))$ ser a única restrição violada nesse ponto e esta ser afim, vale a igualdade em (5.22). Idem se $h_j(\mathbf{x} - \mathbf{f}(\mathbf{x}))$ for a única restrição violada em (5.23).

f) $\forall \mathbf{x} \in \chi$

$$\left[\mathbf{x} - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})]\right]^T \left[\mathbf{x} - \mathbf{f}(\mathbf{x}) - \mathrm{Pr}_\chi[\mathbf{x} - \mathbf{f}(\mathbf{x})]\right] \leq 0 \tag{5.24}$$

g) $\forall \mathbf{x} \in \chi$

$$\|\mathbf{x} - \mathrm{Pr}_{\chi}[\mathbf{x} - \mathbf{f}(\mathbf{x})]\| \leq \|\mathbf{f}(\mathbf{x})\| \tag{5.25}$$

h) $\forall \mathbf{x} \in \chi$

$$\mathbf{f}^{T}(\mathbf{x})\left[\mathbf{x} - \mathbf{f}(\mathbf{x}) - \mathrm{Pr}_{\chi}[\mathbf{x} - \mathbf{f}(\mathbf{x})]\right] \leq 0 \tag{5.26}$$

i) $\forall \mathbf{x} \in \chi$

$$\left[\mathbf{x} - \mathrm{Pr}_{\chi}[\mathbf{x} - \mathbf{f}(\mathbf{x})]\right]^{T} \mathbf{f}(\mathbf{x}) \geq 0 \tag{5.27}$$

De (5.24), (5.26) e (5.27) é fácil concluir que

$$0 \leq \|\mathbf{e}(\mathbf{x})\|^{2} \leq \mathbf{e}^{T}(\mathbf{x})\mathbf{f}(\mathbf{x}) \leq \|\mathbf{f}(\mathbf{x})\|^{2} \tag{5.28}$$

Lema 5.3.7 *([38] e [37]) Para todo* $\mathbf{x} \in \chi$ *e dois escalares* $0 < \alpha_1 < \alpha_2$*, tomando como função de erro (5.17), temos:*

$$\|\mathbf{e}(\mathbf{x}, \alpha_1)\| \leq \|\mathbf{e}(\mathbf{x}, \alpha_2)\| \tag{5.29}$$

$$\frac{\|\mathbf{e}(\mathbf{x}, \alpha_1)\|}{\alpha_1} \geq \frac{\|\mathbf{e}(\mathbf{x}, \alpha_2)\|}{\alpha_2} \tag{5.30}$$

Lema 5.3.8 *([38] e [37]) Seja* $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ *um mapeamento Lipschitz contínuo estritamente monôtono em um conjunto fechado convexo* $\chi \subset \mathbb{R}^n$*, e seja* $\mathbf{x}^*$ *solução única do problema* VI$(\mathbf{f}, \chi)$*, então para todo* $\mathbf{x} \in \mathbb{R}^n$*, existe uma constante* C *tal que*

$$\|\mathbf{x} - \mathbf{x}^*\| \leq C\|\mathbf{e}(\mathbf{x})\| \tag{5.31}$$

Lema 5.3.9 *([85]) Seja* $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ *um mapeamento contínuo fortemente monôtono com respeito ao conjunto viável convexo* $\chi \subset \mathbb{R}^n$ *com módulo* $\beta > 0$*, e* $\mathbf{x}^*$ *solução única do problema* VI$(\mathbf{f}, \chi)$*, então, para todo* $\nu > 1/\beta$

$$\|\mathbf{x} - \mathbf{x}^*\|^{2} \leq 2\nu\mathbf{e}(\mathbf{x}, \nu)^{T}\mathbf{f}(\mathbf{x}) - \|\mathbf{e}(\mathbf{x}, \nu)\|^{2} \quad \forall \mathbf{x} \in \chi \tag{5.32}$$

5.3.3 O problema de otimização convexa como um caso particular de desigualdade variacional

Dado o problema geral de otimização convexa:

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \preceq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{5.33}$$

onde $f_0(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, convexa, contínua e com derivadas parciais contínuas, $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})..g_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $\forall i \in \{1,..,m\} : \ g_i(\mathbf{x})$ convexa, contínua e com derivadas parciais contínuas e $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x})..h_p(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^p$ tal que $\forall i \in \{1,..,p\}$: $h_i(\mathbf{x})$ afim, contínua e com derivadas parciais contínuas. Definindo o conjunto viável

$$\chi = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$$

é sabido ([17, p. 139]) que o ponto ótimo $\mathbf{x}^*$ que minimiza $f_0(\mathbf{x})$ dentro do conjunto viável χ, seja esta solução única ou não, está sujeito à condição

$$\nabla^T f_0(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) \geq 0 \quad \forall \mathbf{x} \in \chi \tag{5.34}$$

e este é exatamente um problema de desigualdade variacional definido para o mesmo conjunto viável χ e com função definida como um mapeamento gradiente $\mathbf{f}(\mathbf{x}) = \nabla f_0(\mathbf{x})$. Podemos definir, portanto, o seguinte teorema

Teorema 5.3.10 *Um ponto $\mathbf{x}^* \in \chi$, com $\chi \subseteq \mathbb{R}^n$ convexo é o ponto ótimo do problema (5.33) se e somente se este ponto é solução do problema (5.7) definido como* $\mathrm{VI}(\nabla f_0, \chi)$.

É sabido ([46] e suas referências) que uma função objetivo $f_0(\mathbf{x})$ diferenciável é (estritamente) convexa se e somente se seu gradiente for (estritamente) monôtono. Mais ainda, a função é (estritamente) pseudoconvexa se e somente se seu gradiente for (estritamente) pseudomonôtono. Se o gradiente for fortemente pseudomonôtono, então a função objetivo escalar é fortemente pseudoconvexa, mas a recíproca desta afirmação não é verdadeira.

Nos casos em que a função $\mathbf{f}(\mathbf{x})$ não corresponder ao gradiente de uma função escalar, o mapeamento chama-se assimétrico em χ.

Observe-se que, assim como o problema VI$(\mathbf{f}, \chi)$, o problema de otimização convexa (5.33) pode admitir mais de uma solução. Nos casos:
i) $f_0(\mathbf{x})$ afim, por condição de primeira ordem das funções convexas ([17, p. 69])

$$\begin{aligned} \forall \mathbf{y},\ \mathbf{x} \in \chi \text{ e } \mathbf{y} \neq \mathbf{x}: \quad & \nabla^T f_0(\mathbf{x})(\mathbf{y}-\mathbf{x}) = f_0(\mathbf{y}) - f_0(\mathbf{x}) \\ & \nabla^T f_0(\mathbf{y})(\mathbf{x}-\mathbf{y}) = f_0(\mathbf{x}) - f_0(\mathbf{y}) \\ \Rightarrow \quad & (\nabla^T f_0(\mathbf{x}) - \nabla^T f_0(\mathbf{y}))(\mathbf{y}-\mathbf{x}) = 0 \\ \Rightarrow \quad & (\nabla f_0(\mathbf{x}) - \nabla f_0(\mathbf{y}))^T(\mathbf{x}-\mathbf{y}) = 0 \end{aligned}$$

Portanto ∇f_0 é monôtono mas não estritamente e a solução pode não ser única.
ii) $f_0(\mathbf{x})$ estritamente convexa, por condição de primeira ordem das funções convexas

$$\begin{aligned} \forall \mathbf{y},\ \mathbf{x} \in \chi \text{ e } \mathbf{y} \neq \mathbf{x}: \quad & \nabla^T f_0(\mathbf{x})(\mathbf{y}-\mathbf{x}) < f_0(\mathbf{y}) - f_0(\mathbf{x}) \\ & \nabla^T f_0(\mathbf{y})(\mathbf{x}-\mathbf{y}) < f_0(\mathbf{x}) - f_0(\mathbf{y}) \\ \Rightarrow \quad & (\nabla^T f_0(\mathbf{x}) - \nabla^T f_0(\mathbf{y}))(\mathbf{y}-\mathbf{x}) < 0 \\ \Rightarrow \quad & (\nabla f_0(\mathbf{x}) - \nabla f_0(\mathbf{y}))^T(\mathbf{x}-\mathbf{y}) > 0 \end{aligned}$$

Portanto ∇f_0 é estritamente monôtona e a solução é única.

A recíproca destas afirmações não é verdadeira, isto é, nem todo operador (estritamente) monôtono é um mapeamento gradiente de uma função (estritamente) convexa (ver um contraexemplo em [50, pp. 16-17]).

Observe-se também que o problema VI$(\mathbf{f}, \chi)$ com $\chi \subseteq \mathbb{R}^n$ convexo e $\mathbf{f}(\mathbf{x}) = \nabla f_0(\mathbf{x}) : \chi \to \mathbb{R}^n$ pode não corresponder a um problema de minimização convexa como descrito em (5.33), pois f_0 pode não ser convexa.

Efetivamente, dada uma função $f_0(\mathbf{x}) : \chi \to \mathbb{R}$ não necessariamente convexa, se $\mathbf{x}^*$ é o ponto ótimo de $f_0(\mathbf{x})$ (o ponto que minimiza $f_0(\mathbf{x})$ dentro de χ), então para todo $\mathbf{x} \in \chi:\ \nabla^T f_0(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) \geq 0$, sendo portanto $\mathbf{x}^*$ solução do problema VI$(\nabla f_0(\mathbf{x}), \chi)$.

A recíproca desta afirmação não é verdadeira, isto é, nem todo ponto $\bar{\mathbf{x}} \in \chi$ tal que para todo $\mathbf{x} \in \chi:\ \nabla^T f_0(\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}}) \geq 0$, é ponto ótimo (minimizador) de $f_0(\mathbf{x})$ dentro de χ.

Portanto, nem todo problema de desigualdade variacional onde a função é um mapeamento gradiente corresponde a um problema equivalente de otimização convexa.

5.3.4 Problemas equivalentes

O problema de desigualdade variacional VI($\mathbf{f}(\mathbf{x}), \chi$) é, na maioria dos casos, não trivial e de difícil solução, tanto analítica como achada através de métodos numéricos ou algoritmos.

Uma estratégia para encontrar uma solução consiste em formular o problema de uma maneira que, sob certas condições, seja equivalente, isto é, apresente igual solução do problema original.

Já foi mencionado que o valor que zera a função de erro $\mathbf{e}(\mathbf{x}) = \mathbf{x} - \mathrm{Pr}_{\chi}\left[\mathbf{x} - \mathbf{f}(\mathbf{x})\right]$ é solução do problema de desigualdade variacional. Porém, a dificuldade deste problema é, na maioria dos casos, similar.

Solodov e Tseng ([75] e suas referências), mencionam outros problemas equivalentes, que serão apresentados a continuação.

Define-se como função *gap*

$$g(\mathbf{x}) := \max_{\mathbf{y} \in \chi} \mathbf{f}^T(\mathbf{x})(\mathbf{x} - \mathbf{y}) \tag{5.35}$$

O valor $\mathbf{x} \in \chi$ que minimiza esta função dentro do conjunto viável é solução do problema VI($\mathbf{f}, \chi$).

A dificuldade que encontram os algoritmos baseados em métodos de decrescimento da função (5.35) consiste em que esta não é diferenciável.

Uma solução a este problema é a utilização da chamada função *gap* regularizada:

$$\begin{aligned} g_{\alpha}(\mathbf{x}) &:= \max_{\mathbf{y} \in \chi} \mathbf{f}^T(\mathbf{x})(\mathbf{x} - \mathbf{y}) - \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{y}\|^2 \\ &= \mathbf{f}^T(\mathbf{x})\left(\mathbf{x} - \mathrm{Pr}_{\chi}\left[\mathbf{x} - \alpha\mathbf{f}(\mathbf{x})\right]\right) - \frac{1}{2\alpha}\|\mathbf{x} - \mathrm{Pr}_{\chi}\left[\mathbf{x} - \alpha\mathbf{f}(\mathbf{x})\right]\|^2 \end{aligned}$$

onde α é uma constante positiva. Esta função é contínuamente diferenciável, e está demonstrado ([75] e suas referências) que se $\mathbf{x}^*$ é ponto estacionário do problema de minimizar $g_{\alpha}(\mathbf{x})$ sujeito a $\mathbf{x} \in \chi$, e o jacobiano da função objetivo nesse ponto, $D_{\mathbf{f}}(\mathbf{x}^*)$ for positivo definido, então $\mathbf{x}^*$ é solução do problema VI($\mathbf{f}, \chi$).

Solodov e Tseng ([75]) utilizam a chamada função *D-gap*

$$h_{\alpha,\beta}(\mathbf{x}) := g_\alpha(\mathbf{x}) - g_\beta(\mathbf{x}) \quad \text{onde } \alpha > \beta > 0 \tag{5.36}$$

A função (5.36) é continuamente diferenciável, e está demonstrado ([75] e suas referências) que se $\mathbf{x}^*$ é ponto estacionário do problema de minimizar $h_{\alpha,\beta}(\mathbf{x})$, sujeito a $\mathbf{x} \in \chi$, e se o jacobiano da função objetivo nesse ponto, $D_{\mathbf{f}}(\mathbf{x}^*)$, for positivo definido, então $\mathbf{x}^*$ é solução do problema VI($\mathbf{f}, \chi$).

5.3.5 Desigualdades variacionais generalizadas

Muitos problemas de otimização podem ser planteados como problemas de resolução de desigualdades variacionais generalizadas. Este conceito é mais geral que o de desigualdades variacionais, pois abrange este assim como outros problemas de otimização.

Definição 5.3.11 *([81])*

Dados dois mapeamentos $\mathbf{g}$, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, *e um conjunto convexo* $\chi \subseteq \mathbb{R}^n$, *define-se como problema de desigualdade variacional generalizada (GVI) ao problema de encontrar* $\mathbf{x}^* \in \chi$, *tal que* $\mathbf{g}(\mathbf{x}^*) \in \chi$ *e*

$$\forall \mathbf{x} \in \chi \quad (\mathbf{x} - \mathbf{g}(\mathbf{x}^*))^T \mathbf{f}(\mathbf{x}^*) \geq 0 \tag{5.37}$$

Segundo esta definição, pode-se observar que o problema de desigualdade variacional VI($\mathbf{f}, \chi$) pode ser planteado como um problema GVI no conjunto χ de mapeamento $\mathbf{f}(\mathbf{x})$, onde $\mathbf{g}(\mathbf{x}) = \mathbf{x}$. Portanto, este problema é um caso particular de GVI[1].

Definição 5.3.12 *G-monotonicidade ([81])*

Um mapeamento $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ *é* G-monôtono *em* $\mathbf{x}^*$ *se*

$$\forall \mathbf{x} \in \mathbb{R}^n \quad [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}^*)]^T [\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)] \geq 0 \tag{5.38}$$

[1]Esta definição difere daquela apresentada em [82] e [65], que definem o problema como encontrar $\mathbf{x}^*$ tal que $\forall \mathbf{x} \in \chi \ [\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)]^T \mathbf{f}(\mathbf{x}^*) \geq 0$. Solodov em [73], define o problema como encontrar $\mathbf{x}^*$ tal que $\forall \mathbf{x} \in \chi \ [\mathbf{x} - \mathbf{g}(\mathbf{x}^*)]^T \mathbf{f}(\mathbf{x}^*) \geq f[\mathbf{g}(\mathbf{x}^*)] - f(\mathbf{x})$, onde $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ é o indicador do conjunto viável: $f(\mathbf{x}) = 0 \ \forall \mathbf{x} \in \chi, \ \ f(\mathbf{x}) = +\infty$ para outros valores.

O mapeamento $\mathbf{f}$ *é* estritamente G-monôtono *em* $\mathbf{x}^*$ *se a desigualdade for estrita para todo* $\mathbf{x} \neq \mathbf{x}^*$*, e* fortemente G-monôtono *em* $\mathbf{x}^*$ *se existe uma constante* $\beta > 0$ *tal que*

$$\forall \mathbf{x} \in \mathbb{R}^n \quad [\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}^*)]^T [\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)] \geq \beta \|\mathbf{x} - \mathbf{x}^*\|^2 \tag{5.39}$$

Observe-se que estas definições correspondem às de monotonicidade, monotonicidade estrita e monotonicidade forte, respectivamente, quando $\mathbf{g}(\mathbf{x}) = \mathbf{x}$.

A expressão

$$\mathbf{g}(\mathbf{x}) - \mathrm{Pr}_{\chi} [\mathbf{g}(\mathbf{x}) - \mathbf{f}(\mathbf{x})] \tag{5.40}$$

é igual a zero se e somente se $\mathbf{x} = \mathbf{x}^* \in \chi$. Portanto, resolver o problema de GVI é equivalente a achar o zero de (5.40), o qual pode ser utilizado como uma medida de erro.

5.4 Problemas de teste

Os algoritmos desenvolvidos neste capítulo serão testados com os problemas descritos a continuação.

Problema 1

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 + x_2 + 1 \leq 0 \\ & (x_1 + 0.5)^2 + x_2^2 - 1 \leq 0 \\ & 2x_1 - x_2 + 1 = 0 \end{aligned}$$

Este é o primeiro problema utilizado no capítulo de otimização convexa.

Aqui, o ponto ótimo está em $\mathbf{x}^* = [-0.6667 \ -0.3333]^T$.

Problema 2

$$\begin{aligned} \min \quad & (x_1 - 1)^2 - x_2 + 2 \\ \text{s.t.} \quad & x_1 + x_2^2 + 1 \leq 0 \\ & (x_1 + 1)^2 + x_2^2 - 0.5 \leq 0 \\ & -2x_1 + x_2 - 3 \leq 0 \\ & 11x_1 - 5x_2 + 13 \leq 0 \\ & -x_2 - 1 \leq 0 \end{aligned}$$

Este é o terceiro problema utilizado no capítulo de otimização convexa.

O ponto ótimo está em $\mathbf{x}^* = [-1.0679\ 0.2506]^T$.

Problema 3

Este é o terceiro problema de teste descrito em [64, pp. 353-354], e o quarto utilizado no capítulo de otimização convexa, e modeliza a geração de energia por parte de três geradores para satisfazer uma determinada demanda durante um determinado período. A função objetivo é quadrática estritamente convexa de 15 variáveis. Possui 65 restrições de desigualdade afins, sendo 36 delas condições de limite das variáveis, e nenhuma restrição de igualdade. Seu ponto ótimo é $\mathbf{x}^* = [8\ 57\ 3\ 1\ 64\ 0\ 0\ 71\ 0\ 1\ 78\ 6\ 3\ 85\ 12]^T$.

Problema 4

O modelo matemático que descreve este problema pode ser encontrado em [16] com o índice s383, e é o quinto problema utilizado no capítulo de otimização convexa. A função objetivo possui 14 variáveis e está restringida por 28 restrições de desigualdade (que correspondem aos limites superior e inferior das variáveis), e uma restrição de igualdade afim.

$$\begin{array}{lll} \min & \sum_{i=1}^{14} a_i|x_i & \\ \text{s.t.} & -x_i \leq 0 & \forall i \in \{1,..,14\} \\ & x_i - 0.04 \leq 0 & \forall i \in \{1,..,5\} \\ & x_i - 0.03 \leq 0 & \forall i \in \{6,..,14\} \\ & \sum_{i=1}^{14} c_i x_i - 1 = 0 & \end{array}$$

onde c_i e a_i são constantes especificadas em [16]. O ponto ótimo está em $\mathbf{x}^* = 10^{-2}[4\ 3.32\ 3.23\ 3.10\ 3.09\ 2.78\ 2.67\ 2.49\ 2.36\ 2.21\ 2.04\ 1.96\ 2.08\ 2.58]^T$.

Problema 5

$$\begin{array}{rl} \mathbf{f}(\mathbf{x}) = & \left[\begin{array}{c} 2x_1 + 2x_2 + 0.004x_1^3 - 8 \\ 2x_2 + x_3 + 0.007x_2^3 - 6 \\ 2x_1 + x_3 + 0.005x_3^3 - 4 \end{array}\right] \\ \text{s.t} & x_1 - x_2 - x_3^2 \leq 0 \\ & 2 - x_1^2 - x_2 \leq 0 \end{array}$$

Este é o exemplo 1 apresentado em [30].

O ponto ótimo está em $\mathbf{x}^* = [1.083 \;\; 0.826 \;\; 0.507]^T$.

Problema 6

$$\begin{aligned} \mathbf{f}(\mathbf{x}) = & \begin{bmatrix} 4x_1 - 3x_2 - x_3 + 1 \\ -x_1 + 4x_2 - 3x_3 + 1 \\ -3x_1 - x_2 + 4x_3 + 1 \end{bmatrix} \\ \text{s.t} \quad & -2x_1 + x_2^2 + x_3^2 \leq 0 \\ & x_1 + x_2 + x_3 - 2 = 0 \end{aligned}$$

Este é o exemplo 2 de [30].

O ponto ótimo está em $\mathbf{x}^* = [2/3 \;\; 2/3 \;\; 2/3]^T$.

Problema 7

$$\begin{aligned} \min \; f_0(\mathbf{x}) = \quad & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ & +10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\ \text{s.t.} \quad & 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \leq 0 \\ & 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0 \\ & 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \leq 0 \\ & 4x_1^2 + x_2^2 + 2x_3^2 - 3x_1x_2 + 5x_6 - 11x_7 \leq 0 \end{aligned}$$

Este é o terceiro exemplo de [30].

O ponto ótimo está em $\mathbf{x}^* = [2.33 \; 1.95 \; -0.48 \; 4.37 \; -0.62 \; 1.04 \; 1.6]^T$.

Note que, neste problema, a função objetivo não é convexa.

Problema 8

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) = \tfrac{1}{2}\Sigma_{i=1}^{n}(x_{2i-1} - x_{2i})^2 \\ \text{s.t.} \quad & x_{2i-1}^2 - x_{2i} - 1 \leq 0 \qquad i \in \{1, .., n\} \\ & 2x_{2i-1} + x_{2i} - 3 = 0 \end{aligned}$$

Este é o quarto problema apresentado em [30], onde foi implementado com $n = 300$ e $n = 600$.

O ponto ótimo está em $\mathbf{x}^* = [1.2, \;\; 0.6, ..., 1.2, \;\; 0.6]^T$.

5.5 Algoritmos para a resolução de problemas de desigualdades variacionais

Nesta seção serão mencionados e comentados alguns algoritmos, contínuos e discretos, apresentados em diversas bibliografias, destinados a resolver o problema de desigualdades variacionais ou de minimização convexa aplicando desigualdades variacionais. Estes algoritmos possuem algumas características (como a utilização de projeção ortogonal, por exemplo) que os fazem representativos de todos aqueles propostos ou referenciados na bibliografia.

5.5.1 Algoritmo apresentado por Liao ([53])

Em [53], considera-se o problema de minimização de uma função convexa sujeita apenas a restrições de desigualdade convexas. Não considera, portanto, a presença de restrições de igualdade afins. Define-se:

$$\begin{array}{rl} \min_{\mathbf{x}\in\mathbb{R}^n} & f_0(\mathbf{x}) \\ \text{s.t.} & \mathbf{g}(\mathbf{x}) \preceq \mathbf{0} \end{array} \tag{5.41}$$

onde $f_0(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ convexa, contínua e com derivadas parciais contínuas, $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})..g_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $\forall i \in \{1,..,m\} : \ g_i(\mathbf{x})$ contínua, convexa e com derivadas parciais contínuas, e $\chi = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}\}$ conjunto viável convexo. É assumido através de todo o trabalho que o problema é realizável e que a condição de Slater ([17, p. 226]) se verifica. Esta condição exige que exista $\mathbf{x} \in \chi$ tal que $\mathbf{g}(\mathbf{x}) \prec \mathbf{0}$, existindo portanto dualidade forte e garantindo a observância das condições KKT (ver [17, p. 243]), isto é, que existe $\mathbf{y}^* \in \mathbb{R}^m_+$ tal que

$$\begin{array}{l} \nabla f_0(\mathbf{x}^*) + D^T_{\mathbf{g}}(\mathbf{x}^*)\mathbf{y}^* = \mathbf{0} \\ \mathbf{y}^* \succeq \mathbf{0} \\ \mathbf{g}(\mathbf{x}^*) \preceq \mathbf{0} \\ \mathbf{g}(\mathbf{x}^*)^T\mathbf{y}^* = 0 \end{array} \tag{5.42}$$

sendo $\mathbf{x}^*$ o ponto ótimo do problema (5.41).

Liao ([53]) transforma o problema (5.41) em um problema de desigualdade varia-

cional, definindo:

$$\mathbf{u} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \in \mathbb{R}^{n+m}, \qquad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \nabla f_0(\mathbf{x}) + D_{\mathbf{g}}^T(\mathbf{x})\mathbf{y} \\ -\mathbf{g}(\mathbf{x}) \end{bmatrix} : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m} \tag{5.43}$$

e considerando o problema VI($\mathbf{f}(\mathbf{u}), \Omega$), onde $\Omega = \{\mathbf{u} \in \mathbb{R}^{n+m} \mid \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{R}^m_+\}$, consistente em achar $\mathbf{u}^*$ tal que $\forall \mathbf{u} \in \Omega :\ (\mathbf{u} - \mathbf{u}^*)^T \mathbf{f}(\mathbf{u}^*) \geq 0$.

Observe-se que poderia ser considerado simplesmente o problema de desigualdade variacional equivalente ao problema (5.41) VI($\nabla f_0(\mathbf{x}), \chi$), o qual teria uma dimensão n ao invês de uma dimensão $n + m$, como propõe o autor. O inconveniente, neste caso, consiste em que as restrições que definem o conjunto viável χ podem resultar de avaliação mais difícil que as que definem o conjunto Ω, sobre o qual, inclusive, é trivial calcular o operador de projeção ortogonal (5.2).

O autor demonstra que a função $\mathbf{f}(\mathbf{u})$ é monôtona em Ω e que $\mathbf{u}$ satisfaz (5.42) se e somente se $\mathbf{u} = \mathbf{u}^* = [\mathbf{x}^{*T}\ \mathbf{y}^{*T}]^T$. Assim, resolver o problema (5.41) é equivalente a resolver o problema VI($\mathbf{f}(\mathbf{u}), \Omega$), achando ainda o multiplicador de Lagrange ótimo $\mathbf{y}^*$.

Estenderemos aqui os lemas demonstrados por Liao ([53, lemas 2.2 e 2.3]) ao caso geral de otimização convexa.

Dado o problema:

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) \preceq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{5.44}$$

onde $f_0(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ convexa, contínua e com derivadas parciais contínuas, $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})..g_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $\forall i \in \{1, .., m\} :\ g_i(\mathbf{x})$ convexa, contínua e com derivadas parciais contínuas, $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x})..h_p(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^p$ tal que $\forall i \in \{1, .., p\} :$ $h_i(\mathbf{x})$ afim, contínua e com derivadas parciais contínuas. Seja $\chi = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$ e $\mathbf{h}(\mathbf{x}) = \mathbf{0}\} \subseteq \mathbb{R}^n$ e $\chi \neq \phi$ conjunto viável convexo. Seja $\mathbf{x}^*$ um ponto ótimo do problema (5.44), tal que para todo $\mathbf{x} \in \chi :\ (\mathbf{x} - \mathbf{x}^*)^T \nabla f_0(\mathbf{x}^*) \geq 0$. Assume-se que se observa a condição de Slater, isto é, existe $\mathbf{x} \in \chi$ tal que $\mathbf{g}(\mathbf{x}) \prec \mathbf{0}$ e $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, existindo portanto dualidade forte e observando-se as condições KKT, isto é: existem

$\mathbf{y}^* \in \mathbb{R}^m$ e $\mathbf{z}^* \in \mathbb{R}^p$ tal que

$$\begin{aligned} \nabla f_0(\mathbf{x}^*) + D_{\mathbf{g}}^T(\mathbf{x}^*)\mathbf{y}^* + D_{\mathbf{h}}^T(\mathbf{x}^*)\mathbf{z}^* &= \mathbf{0} \\ \mathbf{y}^* &\succeq \mathbf{0} \\ \mathbf{g}(\mathbf{x}^*) &\preceq \mathbf{0} \\ \mathbf{h}(\mathbf{x}^*) &= \mathbf{0} \\ \mathbf{g}(\mathbf{x}^*)^T\mathbf{y}^* &= 0 \end{aligned} \tag{5.45}$$

Definindo

$$\mathbf{u} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} \in \mathbb{R}^{n+m+p}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \nabla f_0(\mathbf{x}) + D_{\mathbf{g}}^T(\mathbf{x})\mathbf{y} + D_{\mathbf{h}}^T(\mathbf{x})\mathbf{z} \\ -\mathbf{g}(\mathbf{x}) \\ -\mathbf{h}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{n+m+p} \tag{5.46}$$

e $\Omega = \{\mathbf{u} \in \mathbb{R}^{n+m+p} \mid \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{R}^m_+,\ \mathbf{z} \in \mathbb{R}^p\}$. O problema $\mathrm{VI}(\mathbf{f}(\mathbf{u}), \Omega)$ consiste em encontrar $\mathbf{u}^* \in \Omega$ tal que $(\mathbf{u} - \mathbf{u}^*)^T\mathbf{f}(\mathbf{u}^*) \geq 0$, para todo $\mathbf{u} \in \Omega$.

Lema 5.5.1 $\mathbf{f}(\mathbf{u})$ *é monôtona em* Ω.

Prova:

$\forall \mathbf{u} = [\mathbf{x}^T\ \mathbf{y}^T\ \mathbf{z}^T]^T \in \Omega,\ \ \forall \bar{\mathbf{u}} = [\bar{\mathbf{x}}^T\ \bar{\mathbf{y}}^T\ \bar{\mathbf{z}}^T]^T \in \Omega$

$$\begin{aligned} (\mathbf{u} - \bar{\mathbf{u}})^T(\mathbf{f}(\mathbf{u}) - \mathbf{f}(\bar{\mathbf{u}})) = \ & (\mathbf{x} - \bar{\mathbf{x}})^T(\nabla f_0(\mathbf{x}) - \nabla f_0(\bar{\mathbf{x}})) \\ & + \mathbf{y}^T(D_{\mathbf{g}}(\mathbf{x})(\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{g}(\mathbf{x}) + \mathbf{g}(\bar{\mathbf{x}})) \\ & - \bar{\mathbf{y}}^T(D_{\mathbf{g}}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{g}(\mathbf{x}) + \mathbf{g}(\bar{\mathbf{x}})) \\ & + \mathbf{z}^T(D_{\mathbf{h}}(\mathbf{x})(\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{h}(\mathbf{x}) + \mathbf{h}(\bar{\mathbf{x}})) \\ & - \bar{\mathbf{z}}^T(D_{\mathbf{h}}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{h}(\mathbf{x}) + \mathbf{h}(\bar{\mathbf{x}})) \end{aligned} \tag{5.47}$$

Por ser f_0 *convexa e por condição de primeira ordem*

$$\left.\begin{aligned} \nabla^T f_0(\mathbf{x})(\bar{\mathbf{x}} - \mathbf{x}) &\leq f_0(\bar{\mathbf{x}}) - f_0(\mathbf{x}) \\ \nabla^T f_0(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}) &\leq f_0(\mathbf{x}) - f_0(\bar{\mathbf{x}}) \end{aligned}\right\} \Rightarrow (\mathbf{x} - \bar{\mathbf{x}})^T(\nabla f_0(\mathbf{x}) - \nabla f_0(\bar{\mathbf{x}})) \geq 0$$

Por ser g_i *convexa* $\forall i \in \{1, .., m\}$

$$\left.\begin{aligned} D_{\mathbf{g}}(\mathbf{x})(\mathbf{x} - \bar{\mathbf{x}}) &\succeq \mathbf{g}(\mathbf{x}) - \mathbf{g}(\bar{\mathbf{x}}) \\ D_{\mathbf{g}}(\bar{\mathbf{x}})(\bar{\mathbf{x}} - \mathbf{x}) &\succeq \mathbf{g}(\bar{\mathbf{x}}) - \mathbf{g}(\mathbf{x}) \end{aligned}\right\} \Rightarrow \begin{aligned} \mathbf{g}(\bar{\mathbf{x}}) - \mathbf{g}(\mathbf{x}) + D_{\mathbf{g}}(\mathbf{x})(\mathbf{x} - \bar{\mathbf{x}}) &\succeq \mathbf{0} \\ \mathbf{g}(\bar{\mathbf{x}}) - \mathbf{g}(\mathbf{x}) + D_{\mathbf{g}}(\bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}}) &\preceq \mathbf{0} \end{aligned}$$

Por ser h_i afim $\forall i \in \{1, .., p\}$

$$\left.\begin{array}{l} D_{\mathbf{h}}(\mathbf{x})(\mathbf{x}-\bar{\mathbf{x}}) = \mathbf{h}(\mathbf{x}) - \mathbf{h}(\bar{\mathbf{x}}) \\ D_{\mathbf{h}}(\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}}) = \mathbf{h}(\mathbf{x}) - \mathbf{h}(\bar{\mathbf{x}}) \end{array}\right\} \Rightarrow \begin{array}{l} D_{\mathbf{h}}(\mathbf{x})(\mathbf{x}-\bar{\mathbf{x}}) - \mathbf{h}(\mathbf{x}) + \mathbf{h}(\bar{\mathbf{x}}) = \mathbf{0} \\ D_{\mathbf{h}}(\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}}) - \mathbf{h}(\mathbf{x}) + \mathbf{h}(\bar{\mathbf{x}}) = \mathbf{0} \end{array}$$

Por ser $\mathbf{y} \succeq \mathbf{0}$ e $\bar{\mathbf{y}} \succeq \mathbf{0}$

$$(\mathbf{u}-\bar{\mathbf{u}})^T(\mathbf{f}(\mathbf{u}) - \mathbf{f}(\bar{\mathbf{u}})) \geq 0$$

Portanto $\mathbf{f}(\mathbf{u})$ é monôtona em Ω

Lema 5.5.2 $\mathbf{u} = [\mathbf{x}^T \ \mathbf{y}^T \ \mathbf{z}^T]^T$ *satisfaz as condições KKT (5.45) se e somente se* $\mathbf{u} = \mathbf{u}^*$.

Prova:

$\Rightarrow$) Se $\mathbf{u}$ satisfaz KKT, então $\mathbf{u} \in \Omega$ porque $\mathbf{y} \succeq \mathbf{0}$

$$\forall \bar{\mathbf{u}} = \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{y}} \\ \bar{\mathbf{z}} \end{bmatrix} \in \Omega : \qquad \bar{\mathbf{x}} \in \mathbb{R}^n, \ \bar{\mathbf{y}} \in \mathbb{R}^m_+, \ \bar{\mathbf{z}} \in \mathbb{R}^p$$

Então:

$$(\bar{\mathbf{u}}-\mathbf{u})^T\mathbf{f}(\mathbf{u}) = (\bar{\mathbf{x}}-\mathbf{x})^T(\nabla f_0(\mathbf{x}) + D^T_{\mathbf{g}}(\mathbf{x})\mathbf{y} + D^T_{\mathbf{h}}(\mathbf{x})\mathbf{z}) - (\bar{\mathbf{y}}-\mathbf{y})^T\mathbf{g}(\mathbf{x}) - (\bar{\mathbf{z}}-\mathbf{z})^T\mathbf{h}(\mathbf{x})$$

Por $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ satisfazerem KKT:

$$\nabla f_0(\mathbf{x}) + D^T_{\mathbf{g}}(\mathbf{x})\mathbf{y} + D^T_{\mathbf{h}}(\mathbf{x})\mathbf{z} = \mathbf{0}$$
$$\mathbf{y}^T\mathbf{g}(\mathbf{x}) = 0$$
$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

Portanto

$$(\bar{\mathbf{u}}-\mathbf{u})^T\mathbf{f}(\mathbf{u}) = -\bar{\mathbf{y}}^T\mathbf{g}(\mathbf{x})$$

Por satisfazer KKT, $\mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$ e por $\bar{\mathbf{y}} \in \Omega$, $\bar{\mathbf{y}} \succeq \mathbf{0}$

$$\Rightarrow (\bar{\mathbf{u}}-\mathbf{u})^T\mathbf{f}(\mathbf{u}) \geq 0 \Rightarrow \mathbf{u} = \mathbf{u}^* = [\mathbf{x}^{*T} \ \mathbf{y}^{*T} \ \mathbf{z}^{*T}]^T$$

$\Leftarrow$) *Se* $\mathbf{u} = \mathbf{u}^* \Rightarrow \mathbf{u} \in \Omega \Rightarrow \mathbf{y} \succeq \mathbf{0}$ *(a)*

$$\begin{aligned} &\forall \bar{\mathbf{u}} = [\bar{\mathbf{x}}^T \ \ \bar{\mathbf{y}}^T \ \ \bar{\mathbf{z}}^T]^T \in \Omega : \\ &(\bar{\mathbf{u}} - \mathbf{u})^T \mathbf{f}(\mathbf{u}) = \ (\bar{\mathbf{x}} - \mathbf{x})^T (\nabla f_0(\mathbf{x}) + D_{\mathbf{g}}^T(\mathbf{x})\mathbf{y} + D_{\mathbf{h}}^T(\mathbf{x})\mathbf{z}) \\ &\qquad\qquad -(\bar{\mathbf{y}} - \mathbf{y})^T \mathbf{g}(\mathbf{x}) - (\bar{\mathbf{z}} - \mathbf{z})^T \mathbf{h}(\mathbf{x}) \geq 0 \end{aligned} \tag{5.48}$$

Escolhendo $\bar{\mathbf{y}} = \mathbf{y}$ *e* $\bar{\mathbf{z}} = \mathbf{z}$*, como (5.48) é válido para todo* $\bar{\mathbf{x}} \in \mathbb{R}^n$

$$\Rightarrow \ \nabla f_0(\mathbf{x}) + D_{\mathbf{g}}^T(\mathbf{x})\mathbf{y} + D_{\mathbf{h}}^T(\mathbf{x})\mathbf{z} = \mathbf{0} \ \ (b)$$

$$\Rightarrow \ -(\bar{\mathbf{y}} - \mathbf{y})^T \mathbf{g}(\mathbf{x}) - (\bar{\mathbf{z}} - \mathbf{z})^T \mathbf{h}(\mathbf{x}) \geq 0 \tag{5.49}$$

onde $\mathbf{y} \succeq \mathbf{0}$ *e* $\bar{\mathbf{y}} \succeq \mathbf{0}$.

Escolhendo em (5.49) $\bar{\mathbf{z}} = \mathbf{z}$ *e* $\bar{\mathbf{y}} = \mathbf{y} + \mathbf{e}_i \succeq \mathbf{0}$ *tal que* $I_m = [\mathbf{e}_1 \ \mathbf{e}_2 .. \mathbf{e}_m]$

$$\Rightarrow \ -\mathbf{e}_i^T \mathbf{g}(\mathbf{x}) \geq 0 \ \ \forall i \in \{1, .., m\} \ \Rightarrow \ g_i(\mathbf{x}) \leq 0 \ \Rightarrow \ \mathbf{g}(\mathbf{x}) \preceq \mathbf{0} \ \ (c)$$

Escolhendo em (5.49) $\bar{\mathbf{z}} = \mathbf{z}$ *e* $\bar{\mathbf{y}} = 2\mathbf{y}$

$$\Rightarrow \ -\mathbf{y}^T \mathbf{g}(\mathbf{x}) \geq 0 \tag{5.50}$$

Escolhendo em (5.49) $\bar{\mathbf{z}} = \mathbf{z}$ *e* $\bar{\mathbf{y}} = \mathbf{0}$

$$\Rightarrow \ \mathbf{y}^T \mathbf{g}(\mathbf{x}) \geq 0 \tag{5.51}$$

De (5.50) e (5.51): $\mathbf{y}^T \mathbf{g}(\mathbf{x}) = 0$ *(d)*

Escolhendo em (5.49) $\bar{\mathbf{y}} = \mathbf{y}$ *e* $\bar{\mathbf{z}} = 2\mathbf{z}$

$$\Rightarrow \ -\mathbf{z}^T \mathbf{h}(\mathbf{x}) \geq 0 \tag{5.52}$$

Escolhendo em (5.49) $\bar{\mathbf{y}} = \mathbf{y}$ *e* $\bar{\mathbf{z}} = \mathbf{0}$

$$\Rightarrow \ \mathbf{z}^T \mathbf{h}(\mathbf{x}) \geq 0 \tag{5.53}$$

De (5.52) e (5.53): $\mathbf{z}^T \mathbf{h}(\mathbf{x}) = 0 \ \Rightarrow \ -\bar{\mathbf{z}}^T \mathbf{h}(\mathbf{x}) \geq 0$

Por ser válido para todo $\bar{\mathbf{z}} \in \mathbb{R}^p \Rightarrow \mathbf{h}(\mathbf{x}) = \mathbf{0}$ *(e)*

De (a), (b), (c), (d), (e), $\mathbf{u} = \mathbf{u}^*$ *verifica KKT.*

Portanto $\mathbf{u}^*$ é solução de VI$(\mathbf{f}(\mathbf{u}), \Omega)$ e satisfaz as condições KKT (5.45), sendo $\mathbf{x}^*$ o ponto ótimo do problema (5.44).

Liao ([53, s. 4]), propõe um algoritmo contínuo para resolver o problema VI$(\mathbf{f}, \Omega)$, onde considera apenas a presença de restrições de desigualdade, baseado na projeção ortogonal sobre o conjunto Ω (de cálculo trivial), onde avalia a cada instante o erro $\mathbf{e}(\mathbf{u}, \beta) = \mathbf{u} - \mathrm{Pr}_\Omega[\mathbf{u} - \beta\mathbf{f}(\mathbf{u})]$. Porém, a estrutura do algoritmo varia segundo a observância de determinadas condições, o que na prática dificulta a aplicação deste. Nenhuma forma de discretização é implementada.

5.5.2 Algoritmo apresentado por Gao *et.al.* ([30])

Gao *et.al.* ([30]) propõem um algoritmo contínuo para resolver o problema de desigualdade variacional (5.7), onde a função $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ pode corresponder a um mapeamento assimétrico, e o conjunto viável está formado pela observância de p restrições de igualdade afins $h_i(\mathbf{x}),\ i \in \{1, .., p\}$ e m restrições de desigualdade convexas $g_i(\mathbf{x}),\ i \in \{1, .., m\}$, todas contínuas e com derivadas parciais contínuas. Também é assumido que o problema é realizável e que existe dualidade forte, e portanto se verificam as condições KKT e são relacionados multiplicadores de Lagrange às restrições. A principal diferença com o trabalho apresentado em [53], além de admitir a presença de restrições de igualdade afins, consiste em que a função objetivo pode não corresponder a um mapeamento gradiente, podendo não se tratar, portanto, de um problema de otimização convexa.

Assim, o algoritmo está destinado a resolver VI$(\mathbf{f}(\mathbf{x}), \chi)$, onde $\chi = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$ e $\mathbf{h}(\mathbf{x}) = \mathbf{0}\} \subseteq \mathbb{R}^n$ conjunto viável convexo, e $\mathbf{f}(\mathbf{x}) : \chi \to \mathbb{R}^n$ monôtona em todo o conjunto viável χ.

Os autores afirmam que $\mathbf{x}^*$ é solução deste problema se e somente se existe $\mathbf{u}^* = [\mathbf{x}^{*T}\ \mathbf{y}^{*T}\ \mathbf{z}^{*T}]^T \in \mathbb{R}^{n+m+p}$ solução de VI$(\mathbf{v}(\mathbf{u}), \Omega)$, onde $\Omega = \{\mathbf{u} \in \mathbb{R}^{n+m+p} \mid \mathbf{x} \in$

$\mathbb{R}^n,\ \mathbf{y} \in \mathbb{R}^m_+,\ \mathbf{z} \in \mathbb{R}^p\}$ e

$$\mathbf{v}(\mathbf{u}) = \begin{bmatrix} \mathbf{f}(\mathbf{x}) + D^T_{\mathbf{g}}(\mathbf{x})\mathbf{y} + D^T_{\mathbf{h}}(\mathbf{x})\mathbf{z} \\ -\mathbf{g}(\mathbf{x}) \\ -\mathbf{h}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{n+m+p}$$

Embora em [30] não é apresentada, a prova desta afirmação está dada pelos aqui demonstrados lemas 5.5.1 e 5.5.2, onde devem ser substituidos $\mathbf{f}(\mathbf{u})$ por $\mathbf{v}(\mathbf{u})$, e $\nabla f_0(\mathbf{x})$ por $\mathbf{f}(\mathbf{x})$, e no lema 5.5.1, deve ser considerada a monotonia de $\mathbf{f}(\mathbf{x})$ em χ, isto é $\forall \mathbf{x},\ \bar{\mathbf{x}} \in \chi :\ (\mathbf{x} - \bar{\mathbf{x}})^T(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\bar{\mathbf{x}})) \geq 0$ no lugar da convexidade de $f_0(\mathbf{x})$.

Para a resolução do problema, é proposta a seguinte rede neural:

$$\dot{\mathbf{u}} = -\kappa \begin{bmatrix} \mathbf{f}(\mathbf{x}) + D^T_{\mathbf{g}}(\mathbf{x})\tilde{\mathbf{y}} + D^T_{\mathbf{h}}(\mathbf{x})(\mathbf{z} - \mathbf{h}(\mathbf{x})) \\ -(\mathbf{y} - \tilde{\mathbf{y}}) \\ -\mathbf{h}(\mathbf{x}) \end{bmatrix} \tag{5.54}$$

onde $\kappa > 0$ é uma constante escalar e $\tilde{y}_i = (y_i - g_i(\mathbf{x}))\text{uhsgn}(y_i - g_i(\mathbf{x}))\ \forall i \in \{1, .., m\}$.

Sob algumas condições relativamente pouco exigentes, os autores demonstram que o algoritmo (5.54) converge assintóticamente a $\mathbf{u}^*$, solução do problema $\text{VI}(\mathbf{v}(\mathbf{u}), \Omega)$, achando assim $\mathbf{x}^*$, solução do problema $\text{VI}(\mathbf{f}(\mathbf{x}), \chi)$.

A aplicação deste algoritmo parece mais simples daquele proposto em [53], além de considerar problemas mais abrangentes; porém, aqui também a dimensão do problema cresce de n variáveis a $n + m + p$ (embora com a vantagem de virar um problema sujeito a restrições triviais). Os autores também não apresentam qualquer forma de discretização.

5.5.3 Rede neural utilizando projeção ortogonal

Em [46] e suas referências é mencionada a utilização da seguinte rede neural para achar o ponto ótimo de um problema de desigualdade variacional, onde a função objetivo $\mathbf{f}(\mathbf{x})$ é Lipschitz contínua e pseudomonôtona no conjunto viável convexo χ.

$$\begin{aligned} &\dot{\mathbf{x}} = -\kappa\{\mathbf{x} - \text{Pr}_\chi\left[\mathbf{x} - \alpha\mathbf{f}(\mathbf{x})\right]\} \\ &\mathbf{x}_0 \in \mathbb{R}^n \end{aligned} \tag{5.55}$$

onde κ e α são escalares positivos. Em [54, lema 3] é demonstrado que o operador de projeção ortogonal é Lipschitz contínuo se assim o for a função objetivo. Portanto, o fato da lei de atualização das variáveis (5.55) ser Lipschitz contínua garante unicidade da trajetória $\mathbf{x}(t)$ para todo $t > t_0$.

A lei de atualização das variáveis (5.55) já foi utilizada em diversas bibliografias mencionadas em [46]. Porém, os autores destacam que a condição de pseudomonotonicidade da função objetivo em todo o domínio desta, exigida nas bibliografias referenciadas, não é suficiente, e mostram um contraexemplo onde, para diferentes valores das constantes κ e α as trajetórias não convergem a um ponto solução $\mathbf{x}^*$. São apresentadas as seguintes condições de convergência:

a) Se $D_{\mathbf{f}}(\mathbf{x})$ for simétrica e $\mathbf{f}(\mathbf{x})$ continuamente diferenciável e pseudomonôtona apenas no conjunto viável χ, as trajetórias $\mathbf{x}(t)$ determinadas por (5.55) são assintoticamente convergentes a um ponto solução $\mathbf{x}^*$.

Se $\mathbf{f}(\mathbf{x})$ for fortemente pseudomonôtona, então para todo $\mathbf{x}_0 \in \chi$ as trajetórias são exponencialmente convergentes a $\mathbf{x}^*$.

b) Se $D_{\mathbf{f}}(\mathbf{x})$ não for simétrica mas $\mathbf{f}(\mathbf{x})$ for fortemente pseudomonôtona com constante β e localmente Lipschitz contínua com constante L, e se verificar que $\beta > 2L$, então para todo $\mathbf{x}_0 \in \chi$ a convergência também é exponencial.

c) Se $\mathbf{f}(\mathbf{x})$ for pseudomonôtona componente a componente em χ e χ for um hipercubo, então as trajetórias $\mathbf{x}(t)$ determinadas por (5.55) são assintoticamente convergentes a um ponto solução $\mathbf{x}^*$.

d) Se $\mathbf{f}(\mathbf{x})$ for fortemente pseudomonôtona componente a componente em χ e χ for um hipercubo, então para todo $\mathbf{x}_0 \in \chi$ as trajetórias convergem exponencialmente a um ponto solução $\mathbf{x}^*$.

Em [46] nenhuma discretização da rede neural apresentada é desenvolvida. Evidentemente, a grande desvantagem na utilização deste algoritmo contínuo consiste na utilização do operador de projeção ortogonal. Os autores apresentam cinco exemplos onde, em todos os casos, o conjunto viável possui uma forma sobre a qual o cálculo da

projeção ortogonal é trivial.

5.5.4 Algoritmo discreto de Goldstein-Levitin-Polyak

O algoritmo de Goldstein-Levitin-Polyak é um conhecido algoritmo discreto mencionado em diversas bibliografias (ver, por exemplo, [38] e [83]). Este apresenta a seguinte lei de atualização das variáveis:

$$\mathbf{x}_{k+1} = \Pr_{\chi}\left[\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)\right] \tag{5.56}$$

sendo α_k um comprimento do passo escolhido adequadamente.

A seguir, estabeleceremos uma condição suficiente para a trajetória ser convergente neste caso.

Definição 5.5.3 *Função fracamente co-coerciva ([83]).*

A função objetivo $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ *define-se como* fracamente co-coerciva *em* χ *se*

$$\begin{aligned} &\forall \mathbf{x},\, \mathbf{y} \in \chi,\ \ \exists g(\mathbf{x},\mathbf{y}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{++} \text{ tal que} \\ &\left[\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\right]^T (\mathbf{x} - \mathbf{y}) \geq g(\mathbf{x},\mathbf{y}) \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\|^2 \end{aligned} \tag{5.57}$$

Se para todo $\mathbf{x}, \mathbf{y} \in \chi$, *observa-se (5.57) e a função* $g(\mathbf{x},\mathbf{y})$ *for maior que uma constante positiva, então a função objetivo é* co-coerciva.

Observe-se que a condição (5.57) é mais restritiva que a monotonicidade para todo o conjunto viável.

A condição suficiente é enunciada no seguinte teorema:

Teorema 5.5.4 *Seja o algoritmo com lei de atualização das variáveis* $\mathbf{x}_{k+1} = \Pr_{\chi}\left[\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)\right]$, *sendo* α_k *um comprimento do passo adequado, e seja a função objetivo* $\mathbf{f}(\mathbf{x})$ *fracamente co-coerciva em* χ *com função* $g(\mathbf{x},\mathbf{y})$ *tal que* $\forall \mathbf{x},\, \mathbf{y} \in \chi, \quad g(\mathbf{x},\mathbf{y}) > \frac{\alpha_k}{2}$, *então a distância ao ponto ótimo é monotônicamente decrescente a cada iteração.*

Prova:

Utilizando o fato que o operador de projeção ortogonal é não expansivo e assumindo que a função objetivo observa (5.57):

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 = \|\mathrm{Pr}_\chi\left[\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)\right] - \mathrm{Pr}_\chi\left[\mathbf{x}^* - \alpha_k \mathbf{f}(\mathbf{x}^*)\right]\|^2 \leq$$
$$\|\mathbf{x}_k - \mathbf{x}^* - \alpha_k[\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)]\|^2 = \tag{5.58}$$
$$\|\mathbf{x}_k - \mathbf{x}^*\|^2 - 2\alpha_k(\mathbf{x}_k - \mathbf{x}^*)^T[\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)] + \alpha_k^2\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2$$

De (5.57):

$$(\mathbf{x}_k - \mathbf{x}^*)^T[\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)] \geq g(\mathbf{x}_k, \mathbf{x}^*)\|\mathbf{f}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}^*)\|^2$$
$$\Rightarrow\ -2\alpha_k(\mathbf{x}_k - \mathbf{x}^*)^T[\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)] + \alpha_k^2\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2 \leq$$
$$-2\alpha_k g(\mathbf{x}_k, \mathbf{x}^*)\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2 + \alpha_k^2\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2 =$$
$$(-2\alpha_k g(\mathbf{x}_k, \mathbf{x}^*) + \alpha_k^2)\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2$$

e substituindo este resultado em (5.58):

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_k - \mathbf{x}^*\|^2 - (2\alpha_k g(\mathbf{x}_k, \mathbf{x}^*) - \alpha_k^2)\|\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}^*)\|^2$$

Assim, se

$$\forall \mathbf{x},\, \mathbf{y} \in \chi \quad g(\mathbf{x}, \mathbf{y}) > \frac{\alpha_k}{2} \tag{5.59}$$

$$\Rightarrow\ 2\alpha_k g(\mathbf{x}_k, \mathbf{x}^*) - \alpha_k^2 > 0$$
$$\Rightarrow\ \ \|\mathbf{x}_{k+1} - \mathbf{x}^*\|^2 < \|\mathbf{x}_k - \mathbf{x}^*\|^2$$

Este teorema é apresentado em [83], embora a prova aqui desenvolvida é mais simples. Nesta bibliografia destaca-se que a condição (5.59) é mais fraca que a monotonicidade forte, condição proposta originalmente no algoritmo.

Assim, a condição (5.59) é suficiente para garantir a aproximação do ponto calculado ao ponto ótimo em cada iteração.

Han e Sun ([38]) apresentam uma variação do algoritmo de Goldstein-Levitin-Polyak, consistente no método empregado para a escolha do comprimento do passo. O critério de parada consiste em conferir que o erro $\mathbf{e}(\mathbf{x}_k, \beta_k) = \mathbf{x}_k - \mathrm{Pr}_\chi[\mathbf{x}_k - \beta_k \mathbf{f}(\mathbf{x}_k)]$ seja menor que uma determinada constante positiva arbitrária. A dificuldade na escolha do comprimento do passo β_k consiste em que o erro $\mathbf{e}(\mathbf{x}_k, \beta_k)$ pode ser igual a zero mesmo em um ponto $\mathbf{x}_k$ diferente da solução $\mathbf{x}^*$, o que acontece, por exemplo, se $\mathbf{x}_k \in \chi$ e

$\beta_k = 0$. Além dessa dificuldade, o principal problema do algoritmo proposto consiste no cálculo do operador de projeção ortogonal, como já apontado, nem sempre possível de realizar[2]. É mencionado também (embora não pareça necessário) que o ponto inicial deve estar dentro do conjunto viável.

5.5.5 Algoritmos discretos baseados em hiperplanos separadores apresentados por Solodov

Solodov ([71]), apresenta um conjunto de algoritmos discretos onde a distância à solução $\mathbf{x}^*$ é monotonicamente decrescente com cada ponto calculado a cada iteração. O interessante destes algoritmos consiste na interpretação geométrica utilizada para demonstrar a convergência. Dado um ponto $\mathbf{x} \in \chi$, acha-se outro ponto $\mathbf{y} \in \chi$ tal que

$$\mathbf{f}^T(\mathbf{y})(\mathbf{x} - \mathbf{y}) > 0 \tag{5.60}$$

Se for observada a condição (5.14), então o hiperplano

$$H_y = \{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{f}^T(\mathbf{y})(\mathbf{w} - \mathbf{y}) = 0\} \tag{5.61}$$

separa estritamente $\mathbf{x}$ de $\mathbf{x}^*$.

O ponto seguinte calcula-se projetando o ponto $\mathbf{x}$ sobre um hiperplano paralelo a H_y, projeção dada por

$$\hat{\mathbf{x}}(\beta) := \mathbf{x} - \beta \frac{\mathbf{f}^T(\mathbf{y})(\mathbf{x} - \mathbf{y})}{\|\mathbf{f}(\mathbf{y})\|^2} \mathbf{f}(\mathbf{y}) \tag{5.62}$$

Observe-se que $\hat{\mathbf{x}}(1)$ é a projeção sobre o hiperplano H_y. Solodov demonstra que para $0 < \beta < 2$, $\|\hat{\mathbf{x}}(\beta) - \mathbf{x}^*\| < \|\mathbf{x} - \mathbf{x}^*\|$, mostrando que o ponto $\hat{\mathbf{x}}(\beta)$ se aproxima da solução $\mathbf{x}^*$.

Lema 5.5.5 *([71, lema 14.1]) Sejam $\mathbf{x} \in \mathbb{R}^n$ e $\mathbf{y} \in \chi$ dois pontos que satisfazem (5.60), e seja $\hat{\mathbf{x}}(\beta)$ definida como em (5.62), onde $\beta \geq 0$. Então, para toda solução $\mathbf{x}^*$*

[2]Em [38], os autores apresentam um exemplo numérico restrito ao ortante não negativo, conjunto sobre o qual o cálculo da projeção ortogonal é trivial.

que satisfaça (5.14),

$$\|\hat{\mathbf{x}}(\beta) - \mathbf{x}^*\|^2 \leq \|\mathbf{x} - \mathbf{x}^*\|^2 - \beta(2-\beta)\left(\frac{\mathbf{f}^T(\mathbf{y})(\mathbf{y}-\mathbf{x})}{\|\mathbf{f}(\mathbf{y})\|}\right)^2$$

Prova:

Chamando $t := \frac{\mathbf{f}^T(\mathbf{y})(\mathbf{x}-\mathbf{y})}{\|\mathbf{f}(\mathbf{y})\|^2} > 0$, *sendo portanto* $\hat{\mathbf{x}}(\beta) = \mathbf{x} - \beta t \mathbf{f}(\mathbf{y})$,

$$\begin{aligned}\|\hat{\mathbf{x}}(\beta) - \mathbf{x}^*\|^2 &= \|\mathbf{x} - \mathbf{x}^*\|^2 + \beta^2 t^2 \|\mathbf{f}(\mathbf{y})\|^2 - 2\beta t(\mathbf{x} - \mathbf{x}^*)^T \mathbf{f}(\mathbf{y}) \\ &\leq \|\mathbf{x} - \mathbf{x}^*\|^2 + \beta^2 t^2 \|\mathbf{f}(\mathbf{y})\|^2 - 2\beta t(\mathbf{x} - \mathbf{y})^T \mathbf{f}(\mathbf{y}) \\ &= \|\mathbf{x} - \mathbf{x}^*\|^2 - \beta(2-\beta) t \mathbf{f}^T(\mathbf{y})(\mathbf{x} - \mathbf{y})\end{aligned}$$

Isto implica que para todo $\beta \in (0, 2)$ *o ponto* $\hat{\mathbf{x}}(\beta)$ *se aproxima da solução* $\mathbf{x}^*$. *Observe-se que* $\hat{\mathbf{x}}(1)$ *é a projeção ortogonal sobre o hiperplano* H_y.

As diferentes maneiras de achar o ponto $\mathbf{y}$ que observe a condição (5.60) é o que da lugar a diferentes algoritmos.

Em [74] os autores apresentam dois algoritmos discretos baseados em projeções sobre semiespaços adequadamente escolhidos. A pesar que em cada iteração dois projeções ortogonais devem ser avaliadas, o fato de serem sobre hiperplanos pode simplificar seu cálculo.

5.5.6 Outros algoritmos apresentados na bibliografia

Em [81], os autores apresentam uma rede neural para resolver problemas de desigualdades variacionais generalizadas. Dados dois mapeamentos $\mathbf{g}$, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ não necessariamente gradientes, e um conjunto χ determinado por limites superior e inferior das variáveis (isto é, um hipercubo), a equação dinâmica

$$\dot{\mathbf{x}} = -\Lambda\left[\mathbf{g}(\mathbf{x}) - \mathrm{Pr}_\chi\left[\mathbf{g}(\mathbf{x}) - \mathbf{f}(\mathbf{x})\right]\right] \tag{5.63}$$

onde $\Lambda = \mathrm{diag}\,(\lambda_i) \succ 0$, é globalmente assintoticamente convergente à solução do problema GVI, $\mathbf{x}^* \in \chi$. Observe-se que este algoritmo é uma generalização ao problema

de GVI do algoritmo (5.55).

Dentre os inconvenientes encontrados neste algoritmo, podemos mencionar o fato de restringir o domínio a um hipercubo, conjunto sobre o qual é trivial o cálculo da projeção ortogonal. Em um dos exemplos numéricos apresentados, os autores consideram restrições mais complexas, e modificam o algoritmo de maneira tal de aumentar a dimensão deste, reduzindo as restrições a um hipercubo, de maneira similar ao realizado em [30]. Nenhuma discretização do algoritmo é apresentada. Como vantagem, obviamente, cabe mencionar que o algoritmo se aplica a uma maior variedade de problemas.

Em [80] os autores estudam a convergência exponencial de diversas redes neurais aplicadas à resolução de problemas de otimização. Dentre os algoritmos contínuos mencionados, um apresentado pelos autores em outro artigo é utilizado para a resolução de problemas de desigualdades variacionais. Este algoritmo apresenta a desvantagem de utilizar projeção ortogonal.

Dentre os outros algoritmos discretos achados na bibliografia, cabe mencionar o trabalho de Yuan ([85]), que propõe uma melhora no algoritmo discreto proposto por He *et.al.* para a resolução de desigualdades variacionais que respondem à seguinte estrutura:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix} \quad \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{f}_1(\mathbf{x}_a) \\ \mathbf{f}_2(\mathbf{x}_b) \end{bmatrix} : (X \cap Y) \subset \mathbb{R}^{m+p} \to \mathbb{R}^{m+p}$$

e o conjunto viável $\chi = \{ \begin{bmatrix} \mathbf{x}_a^T \, \mathbf{x}_b^T \end{bmatrix}^T \mid \mathbf{x}_a \in X, \, \mathbf{x}_b \in Y, \, A\mathbf{x}_a + B\mathbf{x}_b = \mathbf{b} \}$, sendo X e Y conjuntos convexos fechados, $A \in \mathbb{R}^{z \times m}$, $B \in \mathbb{R}^{z \times p}$, $\mathbf{b} \in \mathbb{R}^z$, matrizes e vetores dados.

Este algoritmo está baseado no cálculo de pontos inexatos em cada iteração (a diferença do proposto por He *et.al.*). O erro com respeito ao ponto calculado exatamente vai diminuindo a cada iteração, o que resulta em uma simplificação dos cálculos.

Em [82], os autores estudam as condições de convergência em tempo finito de uma seqüência de pontos fornecida por algoritmos discretos conhecidos. O primeiro algoritmo estudado é o algoritmo de ponto próximo, aplicado à resolução do problema

VI($\mathbf{f}, \chi$):

$$\mathbf{x}_{k+1} = \mathrm{Pr}_{\chi}\left[\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_{k+1})\right] \tag{5.64}$$

sendo α_k uma seqüência conveniente de ganhos positivos. Observe-se a similaridade deste algoritmo com (5.56).

O segundo algoritmo estudado é o método de inêrcia próxima (IPM), aplicado à resolução de problemas de desigualdades variacionais generalizadas:

$$\mathbf{g}(\mathbf{x}_{k+1}) = \mathrm{Pr}_{\chi}\left[\mathbf{g}(\mathbf{x}_k) - \alpha_k \mathbf{f}(\mathbf{x}_{k+1}) + \gamma_k(\mathbf{g}(\mathbf{x}_k) - \mathbf{g}(\mathbf{x}_{k-1}))\right] \tag{5.65}$$

onde $\alpha_k > 0$ e $\gamma_k \geq 0$ são dois números determinados por alguma seqüência conveniente. Evidentemente, a principal desvantagem destes algoritmos consiste na utilização do operador de projeção ortogonal.

Solodov, em [72], estuda a taxa de convergência de diferentes algoritmos conhecidos para resolver o problema de desigualdades variacionais, tais como o método extragradiente, matrix splitting, e método de ponto próximo.

Auslender e Teboulle ([7]) propõem dois algoritmos discretos, chamados de Interior Hyperplane Projection (IHP) e Interior Extragradient (IEG). Ambos algoritmos utilizam uma pseudo projeção ortogonal que, dependendo de uma escolha particular de um operador distância (como pode ser a distância de Bregman, por exemplo), pode gerar operadores de projeção que podem ser calculados analiticamente. Partindo de um ponto interior ao conjunto viável, os algoritmos geram trajetórias que permanecem neste conjunto, e o comprimento do passo é escolhido por métodos conhecidos tais como a regra de Armijo.

Em [75] os autores apresentam dois algoritmos discretos baseados na função *D-gap* (5.36).

Han ([37]) apresenta uma modificação para o método de direção alternada, método consistente em um algoritmo discreto que aumenta a dimensão do problema a $n + m$ sendo m o número de restrições de igualdade. Embora a alternativa proposta apre-

senta algumas vantagens com respeito ao método original, esta precisa da utilização de projeção ortogonal sobre um conjunto convexo, o que acarreta os inconvenientes já mencionados.

Em seguida, será apresentado um algoritmo contínuo baseado em sistema de controle em malha fechada a estrutura variável, projetado utilizando uma função de Liapunov de controle, que não apresenta as dificuldades mencionadas nos algoritmos contínuos propostos. O algoritmo será discretizado por Euler calculando o comprimento do passo de maneira ótima pelo método de controle ótimo de Liapunov.

5.6 Algoritmo contínuo para a resolução de problemas de desigualdades variacionais baseado em uma função de controle de Liapunov

Considere-se o problema (5.7), VI($\mathbf{f}(\mathbf{x}), \chi$). Seja $\mathbf{f}(\mathbf{x}) : \chi \rightarrow \mathbb{R}^n$ um mapeamento contínuo que verifica (5.14), isto é, para todo $\mathbf{x} \in \chi \backslash \chi^* : \ \mathbf{f}^T(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) > 0$, sendo $\mathbf{x}^*$ um elemento do conjunto solução χ^*. Seja $\chi = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0} \text{ e } \mathbf{h}(\mathbf{x}) = \mathbf{0}\} \subseteq \mathbb{R}^n$ conjunto viável convexo tal que $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})..g_m(\mathbf{x})]^T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ tal que $\forall i \in \{1, .., m\} : \ g_i(\mathbf{x})$ convexa, contínua com derivadas parciais contínuas e $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x})..h_p(\mathbf{x})]^T : \mathbb{R}^n \rightarrow \mathbb{R}^p$ tal que $\forall i \in \{1, .., p\}, \ h_i(\mathbf{x})$ afim, contínua com derivadas parciais contínuas.

Seguiremos aqui a abordagem utilizada no capítulo de otimização convexa. Nesse trabalho, foi utilizada uma função de energia com termos de penalização exata (função que apresenta descontinuidades sobre a fronteira do conjunto viável), os quais diferem de zero quando alguma restrição não for observada, isto é, fora do conjunto viável. Um sistema dinâmico do tipo gradiente é utilizado para achar o mínimo desta função não condicionada. Por possuir a função um lado direito descontínuo, o sistema em malha fechada comporta-se como um sistema a estrutura variável, cujo comportamento típico consiste em apresentar trajetórias em modo deslizante. Algoritmos a estrutura variável foram utilizados em [78], [86], [13], entre outros.

Sistemas do tipo gradiente normalmente têm por objetivo o decrescimento de uma função de Liapunov. Esta função, em geral, é escolhida como a norma quadrado de um

resíduo ou erro igual a uma função do tipo distância ao conjunto solução ([72]; em [73], o autor apresenta outras funções de erro ou mérito para o problema de desigualdades variacionais generalizadas).

Escolhe-se uma função de Liapunov candidata tipo Persidskii (ver [13])

$$V(\mathbf{x}) = \sigma(\mathbf{x})\|\mathbf{x} - \mathbf{x}^*\|^2 + \mathbf{g}^T(\mathbf{x})\Lambda \text{uhsgn}(\mathbf{g}(\mathbf{x})) + \mathbf{h}^T(\mathbf{x})\Gamma \text{sgn}(\mathbf{h}(\mathbf{x})) \tag{5.66}$$

onde $\Lambda \in \mathbb{R}^{m\times m}$ tal que $\Lambda = diag(\lambda_i) \succ 0$ e $\Gamma \in \mathbb{R}^{p\times p}$ tal que $\Gamma = diag(\nu_i) \succ 0$ e

$$\sigma(\mathbf{x}) := \begin{cases} 1 & \forall \mathbf{x} \in \chi \\ 0 & \forall \mathbf{x} \notin \chi \end{cases} \tag{5.67}$$

$$\text{uhsgn}(u) := \begin{cases} 1 & \forall u > 0 \\ 0 & \forall u < 0 \end{cases} \tag{5.68}$$

tal como definidos no capítulo referente a otimização convexa.

O coeficiente $\sigma(\mathbf{x})$ pode ser avaliado como:

$$\sigma(\mathbf{x}) = \prod_{i=1}^{m}(1 - \text{uhsgn}(g_i(\mathbf{x}))) \prod_{j=1}^{p}(1 - |\text{sgn}(h_j(\mathbf{x}))|)$$

comentendo o abuso matemático de assumir $\text{sgn}(0) = 0$ e $\text{uhsgn}(0) = 0$.

Os dois últimos termos em (5.66), constituem a penalização da função, e são chamados de penalização exata porque claramente se anulam para todo $\mathbf{x} \in \chi$.

Em seguida, será escolhida a lei de atualização das variáveis para as trajetórias tanto dentro como fora do conjunto viável.

5.6.1 Fase de alcançar o conjunto viável

Nesta fase, $\mathbf{x} \notin \chi$ e a função de Liapunov candidata (5.66) resulta

$$V_{rp}(\mathbf{x}) = \mathbf{g}^T(\mathbf{x})\Lambda \,\text{uhsgn}(\mathbf{g}(\mathbf{x})) + \mathbf{h}^T(\mathbf{x})\Gamma \,\text{sgn}(\mathbf{h}(\mathbf{x})) > 0$$

Calculamos

$$\dot{V}_{rp}(\mathbf{x}) = \left[D_{\mathbf{g}}^T(\mathbf{x})\Lambda \mathrm{uhsgn}(\mathbf{g}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}))\right]\dot{\mathbf{x}}$$

Os termos que contêm os gradientes generalizados dos fatores descontínuos se anulam porque estes gradientes são diferentes de zero apenas nos pontos onde $g_i(\mathbf{x}) = 0$ ou $h_i(\mathbf{x}) = 0$. Pelo método de função de Liapunov de controle, escolhe-se como lei de atualização das variáveis nesta fase

$$\dot{\mathbf{x}} = -\kappa\left[D_{\mathbf{g}}^T(\mathbf{x})\Lambda \mathrm{uhsgn}(\mathbf{g}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}))\right] \tag{5.69}$$

sendo κ um ganho escalar positivo.

Portanto

$$\dot{V}_{rp}(\mathbf{x}) = -\kappa\|D_{\mathbf{g}}^T(\mathbf{x})\Lambda \mathrm{uhsgn}(\mathbf{g}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}))\|_2^2 \leq 0$$

No teorema 4.5.1 apresentado no capítulo de otimização convexa demonstra-se que, fora das superfícies de descontinuidade de $\dot{\mathbf{x}}$, não existe $\lambda_i,\ \nu_i > 0$ tal que $\dot{V}_{rp}(\mathbf{x})$ seja igual a zero. Portanto $\dot{V}_{rp}$ é negativa definida. Nas superfícies de descontinuidade de $\dot{\mathbf{x}}$, o teorema 4.5.2 demostra que uma outra função de Liapunov adequadamente escolhida é negativa definida também, e portanto as trajetórias não se estacionam sobre estas superfícies. Por se tratar de um sistema dinâmico do tipo gradiente, as trajetórias convergirão assintóticamente ao conjunto viável (ver [13, teorema 1.34] e [44]).

5.6.2 Fase de convergência

Nesta fase, $\mathbf{x} \in \chi$ e a função de Liapunov candidata (5.66) resulta $V_{cp}(\mathbf{x}) = \|\mathbf{x}-\mathbf{x}^*\|^2 \geq 0$, a qual é positiva definida. Derivando com respeito ao tempo:

$$\dot{V}_{cp}(\mathbf{x}) = 2(\mathbf{x} - \mathbf{x}^*)^T\dot{\mathbf{x}}$$

e escolhendo, por função de Liapunov de controle

$$\dot{\mathbf{x}} = -\kappa\,\mathbf{f}(\mathbf{x}) \tag{5.70}$$

sendo κ um ganho escalar positivo,

$$\dot{V}_{cp}(\mathbf{x}) = -2\kappa(\mathbf{x} - \mathbf{x}^*)^T \mathbf{f}(\mathbf{x})$$

Por $\mathbf{f}(\mathbf{x})$ satisfazer a condição (5.14),

$$\dot{V}_{cp} < 0$$

Pelo lema de Barbalat ([69, s. 4.5]), por ser $\mathbf{f}(\mathbf{x})$ contínua, $\dot{V}_{cp}(\mathbf{x})$ tende a zero quando $t \to \infty$. No caso trivial, se $\chi^* \subset \text{int}\{\chi\}$, e portanto $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, a convergência das trajetórias ao conjunto solução se produzirá efetivamente nesta fase. No caso geral, $\chi^* \subset \partial\chi$, e as trajetórias convergirão à fronteira do conjunto viável.

De (5.69) e (5.70), conclue-se que a lei de atualização das variáveis é

$$\dot{\mathbf{x}} = -\kappa \left[\sigma(\mathbf{x})\mathbf{f}(\mathbf{x}) + D_{\mathbf{g}}^T(\mathbf{x})\Lambda \,\text{uhsgn}(\mathbf{g}(\mathbf{x})) + D_{\mathbf{h}}^T(\mathbf{x})\Gamma \,\text{sgn}(\mathbf{h}(\mathbf{x}))\right] := \mathbf{m}(\mathbf{x}) \qquad (5.71)$$

a qual apresenta descontinuidades na fronteira do conjunto viável e em toda superfície tal que $f_i(\mathbf{x}) = 0$ para todo $i \in \{1, .., m\}$ ou $h_i(\mathbf{x}) = 0$ para todo $i \in \{1, .., p\}$, sendo portanto o lado direito de (5.71), $\mathbf{m}(\mathbf{x})$, um mapeamento de ponto em conjunto (*set valued map*).

5.6.3 Análise da convergência durante o deslizamento

No capítulo referente a otimização convexa foi realizada uma análise da convergência das trajetórias ao ponto ótimo durante o deslizamento ao longo da fronteira do conjunto viável, podendo ser realizada aqui uma análise similar.

Definindo:

$$\forall \mathbf{x} \in \partial\chi : \quad \mathbf{q}(\mathbf{x}) = \lim_{\substack{\mathbf{x}_i \to \mathbf{x} \\ \mathbf{x}_i \notin \chi}} \left[D_{\mathbf{f}}^T(\mathbf{x}_i)\Lambda \text{uhsgn}(\mathbf{f}(\mathbf{x}_i)) + D_{\mathbf{h}}^T(\mathbf{x}_i)\Gamma \text{sgn}(\mathbf{h}(\mathbf{x}_i))\right]$$

o sistema (5.71) sobre a fronteira do conjunto viável resulta na solução de Filippov de

$\dot{\mathbf{x}}$ (ver [68]):

$$\forall \mathbf{x} \in \partial\chi : \quad \dot{\mathbf{x}} \in \mathcal{F}[\mathbf{m}](\mathbf{x}) = \text{co}\{-\kappa \mathbf{f}(\mathbf{x}) \ , \ -\kappa \mathbf{q}(\mathbf{x})\} \tag{5.72}$$

onde co denota o fecho convexo entre vetores e $\mathcal{F}[\mathbf{m}](\mathbf{x})$ é a solução de Filippov de (5.71).

O vetor $\mathbf{q}(\mathbf{x})$ é perpendicular à fronteira do conjunto viável para quase todo $\mathbf{x} \in \partial\chi$. Nas quinas do conjunto viável $\mathbf{q}(\mathbf{x}) \in \mathcal{N}(\mathbf{x}, \chi)$, sendo $\mathcal{N}(\mathbf{x}, \chi)$ o cone normal a χ no ponto $\mathbf{x}$ ([26]).

O deslizamento sobre a fronteira do conjunto viável se produzirá caso $\mathbf{f}^T(\mathbf{x})\mathbf{q}(\mathbf{x}) \leq 0$, a partir do ponto de contato com esta fronteira. Neste caso, por ser $\dot{\mathbf{x}}$ mensurável e localmente essencialmente limitada, existe pelo menos uma solução de Filippov para $\dot{\mathbf{x}}$, para todo $\mathbf{x} \in \partial\chi$ (ver [23, prop. 4]).

Escolhendo nesta fase a função de Liapunov candidata (5.66):

$$\forall \mathbf{x} \in \partial\chi : \quad V_{bp}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|^2 \tag{5.73}$$

Conferimos

i) Evidentemente, $V_{bp}(\mathbf{x})$ é positiva para todo $\mathbf{x} \in \partial\chi \backslash \chi^*$, e $V_{bp}(\mathbf{x}^*) = 0$, sendo portanto $V_{bp}(\mathbf{x})$ positiva definida.

ii) $\mathbf{0} \in \mathcal{F}[\mathbf{m}](\mathbf{x}^*)$. Portanto $\mathbf{x}^*$ é ponto de equilíbrio de (5.72). Se $\mathbf{x}^*$ estiver em uma quina do conjunto viável, dependerá do caminho do limite na definição de $\mathbf{q}(\mathbf{x})$ para $\mathbf{0} \in \mathcal{F}[\mathbf{m}](\mathbf{x}^*)$.

iii) No caso não trivial $\chi^* \subset \partial\chi$. No caso trivial, as trajetórias alcançam o ponto de equilíbrio durante a fase de convergência.

iv) $V(\mathbf{x})$ é Lipschitz contínuo para todo $\mathbf{x} \in \mathbb{R}^n$.

v) $\mathcal{F}[\mathbf{m}](\mathbf{x})$ é superiormente semi-contínua para todo $\mathbf{x} \in \partial\chi$ e localmente limitada.

vi) Da condição (5.14): $\mathbf{f}^T(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) > 0$, para todo $\mathbf{x} \in \partial\chi$.

Por, para todo $\mathbf{x} \in \partial\chi$, $\mathbf{q}(\mathbf{x})$ ser perpendicular à fronteira do conjunto viável, ou, caso o ponto $\mathbf{x}$ estar em uma quina deste, $\mathbf{q}(\mathbf{x}) \in \mathcal{N}(\mathbf{x}, \chi)$, e ser χ convexo:

$$\begin{aligned} &\forall \mathbf{y} \in \chi, \ \forall \mathbf{x} \in \partial\chi, \quad \mathbf{q}^T(\mathbf{x})(\mathbf{y} - \mathbf{x}) \leq 0 \\ &\Rightarrow \ \forall \mathbf{x} \in \partial\chi, \quad \mathbf{q}^T(\mathbf{x})(\mathbf{x} - \mathbf{x}^*) \geq 0 \end{aligned} \tag{5.74}$$

A derivada de Lie da função de Liapunov ao longo da fronteira onde se produz a trajetória por modo deslizante está definida ([23]):

$$\dot{V}(\mathbf{x}) \in \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V(\mathbf{x}) = 2(\mathbf{x}-\mathbf{x}^*)^T \dot{\mathbf{x}} = 2 \text{ co}\{-\kappa \mathbf{f}^T(\mathbf{x})(\mathbf{x}-\mathbf{x}^*), -\kappa \mathbf{q}^T(\mathbf{x})(\mathbf{x}-\mathbf{x}^*)\}$$

e de (5.14) e (5.74), concluimos:

$$\max \bar{\mathcal{L}}_{\dot{\mathbf{x}}} V(\mathbf{x}) \leq 0$$

e portanto o sistema (5.7)-(5.72), para todo $\mathbf{x} \in \partial\chi$ é estável ([23, teorema 1]).

Observe-se que $\mathbf{q}^T(\mathbf{x})(\mathbf{x}-\mathbf{x}^*) = 0$ apenas se a restrição violada em $B(\partial\chi, \delta) \cap (\mathbb{R}^n \backslash \chi)$, for afim, e, neste caso, $\mathbf{q}(\mathbf{x})$ é perpendicular à fronteira do conjunto viável neste ponto. Pela condição (5.14), para todo $\mathbf{x} \notin \chi^*$, $\mathbf{f}(\mathbf{x})$ não é perpendicular à fronteira neste caso, e portanto $\mathbf{0} \notin \mathcal{F}[\mathbf{m}](\mathbf{x})$, o que implica $\dot{\mathbf{x}} \neq \mathbf{0}$ e portanto a trajetória não se estaciona neste ponto. Portanto $\mathbf{x}^* \in \chi^*$ é ponto de equilíbrio assintóticamente estável do sistema (5.72)-(5.7) ([23, teorema 1]).

Zak ([86, s. 6.9]) apresenta uma análise detalhada da convergência das trajetórias durante o deslizamento para um sistema em malha fechada similar, utilizando também penalização exata, análise que também pode ser realizada neste conexto.

Observe-se que a condição (5.14), única condição aqui exigida à função objetivo, é mais fraca que a de pseudomonotonicidade no conjunto viável χ, como exigido em [46], ou que a monotonicidade simples como exigido em [30], entre outras bibliografias.

Podemos enunciar, assim, o seguinte teorema

Teorema 5.6.1 *Para toda função objetivo contínua, que satisfaz (5.14), sujeita a restrições de desigualdade convexas e de igualdade afins, ambas continuamente diferenciáveis, com as trajetórias definidas em (5.71), e para todo* $\mathbf{x}_0 \in \mathbb{R}^n$*, o sistema descrito pelas equações (5.7) e (5.71) é globalmente assintóticamente convergente ao conjunto solução* χ^*.

5.7 Execução do algoritmo

O algoritmo contínuo foi implementado utilizando a função ODE23 do programa MATLAB 6, utilizando, exceto nos casos especificados, os parâmetros **RelTol**=$1e-4$ e **AbsTol**=$1e-5$. As constantes de penalização λ_i e ν_i foram escolhidas unitárias e o ganho κ variável para cada teste. Em todos os casos, o conjunto viável está definido pela observância das restrições.

Primeiramente será apresentada, a maneira de ilustração, o algoritmo implementado com o problema 1, e onde são escolhidos como função objetivo $\mathbf{f}(\mathbf{x}) = \nabla f_0(\mathbf{x}) = [2x_1\ 2x_2]^T$ e como ganho $\kappa = 1$.

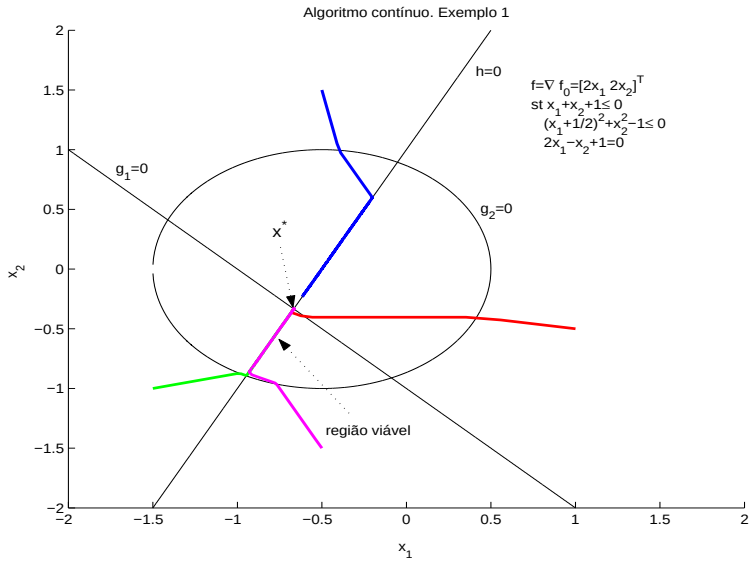

Figura 5.2: Algoritmo contínuo aplicado ao exemplo 1 a partir de 4 pontos iniciais diferentes, mostrando a convergência das trajetórias

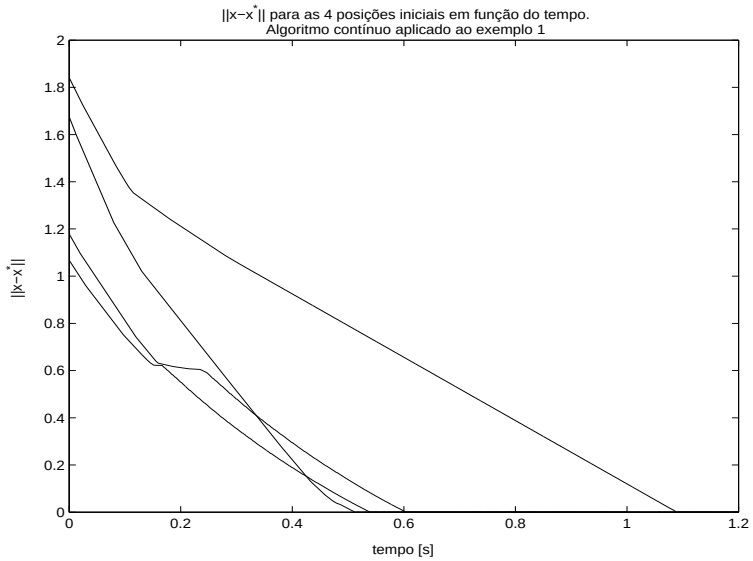

Figura 5.3: Norma do erro de trajetória em função do tempo para o algoritmo contínuo aplicado ao exemplo 1 a partir de cada um dos pontos iniciais testados

A seguir, serão implementados os problemas 5 a 8 (os 4 exemplos apresentados em [30]), a fim de poder comparar o desempenho do algoritmo proposto com o daquele apresentado nessa bibliografia. Serão utilizados os mesmos ganhos que [30].

Os gráficos seguintes mostram as normas dos erros de trajetória em função do tempo para cada um dos exemplos apresentados a partir de diversos pontos iniciais escolhidos aleatoriamente.

O exemplo 5 é implementado com ganho $\kappa = 10$.

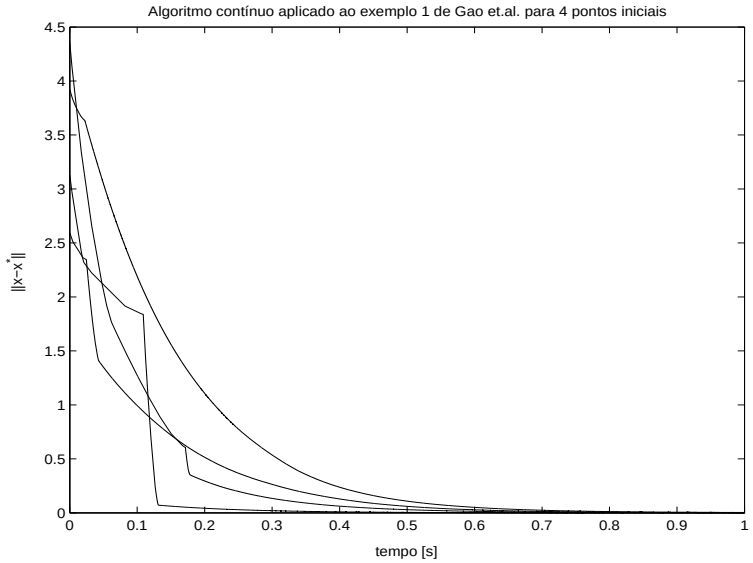

Figura 5.4: Norma dos erros de trajetória do algoritmo para o exemplo 5 a partir de 4 pontos iniciais

O exemplo 6 é implementado com ganho $\kappa = 1$.

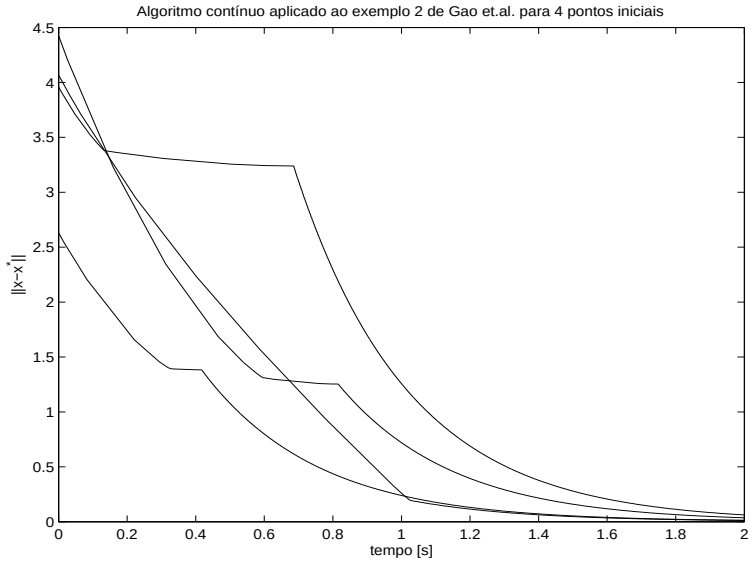

Figura 5.5: Norma dos erros de trajetória do algoritmo para o exemplo 6 a partir de 4 pontos iniciais

O exemplo 7 é implementado com ganho $\kappa = 10$.

Este problema não é um problema de otimização convexa, entretanto, o algoritmo mostra-se capaz de resolver $\text{VI}(\nabla f_0(\mathbf{x}), \chi)$.

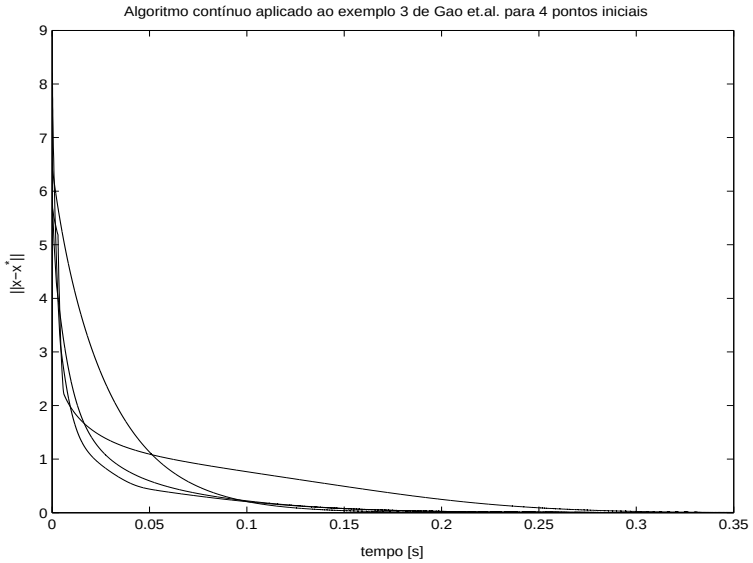

Figura 5.6: Norma dos erros de trajetória do algoritmo para o exemplo 7 a partir de 4 pontos iniciais

O problema 8 foi implementado em [30] para $n = 300$ e $n = 600$, com ganho $\kappa = 20$, sendo a função objetivo $\nabla f_0(\mathbf{x})$. Aqui será implementado apenas para $n = 300$, mesmo ganho, e os parâmetros **AbsTol**=$1e-4$ e **RelTol**=$1e-3$.

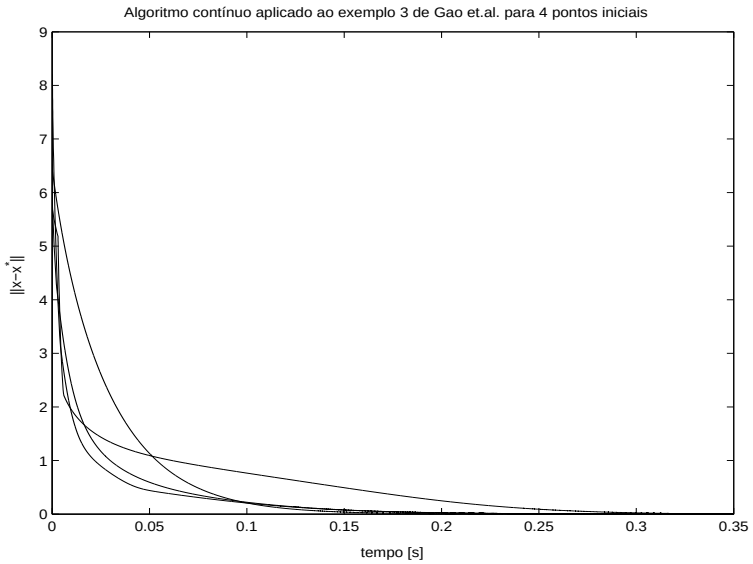

Figura 5.7: Norma dos erros de trajetória do algoritmo para o exemplo 8 a partir de 2 pontos iniciais, $n = 300$

Comparando com os gráficos apresentados em [30], resulta evidente que o algoritmo aqui apresentado converge ao valor ótimo muito mais rapidamente que o apresentado naquela bibliografia, utilizando sempre os mesmos ganhos. Porém, consideramos que a principal vantagem não é a taxa de convergência e sim a dimensão do problema, que aqui permanece sendo igual ao número de variáveis n, enquanto em [30] é de $n+m+p$. Além desta vantagem, cabe reiterar que a condição (5.14) é mais fraca que a monotonicidade exigida em [30], assim como a pseudomonotonicidade exigida em [46], abrangendo nosso algoritmo um maior número de problemas.

5.8 Algoritmo discreto para a resolução de problemas de desigualdades variacionais baseado em controle ótimo de Liapunov

Nesta seção, será discretizado o agoritmo contínuo (5.71) por Euler, calculando o passo ótimo de iteração por controle ótimo de Liapunov. Assim, o algoritmo obtido apresentará as seguintes vantagens: (i) possuir um comprimento do passo otimizado e não calculado de maneira arbitrária apenas para garantir a convergência do algoritmo (como em [38]); (ii) não apresentar restrições com respeito à posição inicial; (iii) não utilizar

o operador de projeção ortogonal que, como já foi apontado, pode ser computacionalmente caro de calcular.

Uma versão preliminar deste algoritmo foi publicada em [60].

No contexto da discretização do algoritmo contínuo, o controle ótimo de Liapunov é aplicado no cálculo do comprimento do passo a fim de minimizar a norma quadrado de um resíduo ou erro. A maior dificuldade neste processo consiste na escolha adequada desse resíduo. Efetivamente, o problema de desigualdade variacional pode ser planteado como um problema de minimização da norma de um resíduo, o qual poderia ser definido das seguintes maneiras:

a) $\mathbf{r}(\mathbf{x}) = \mathbf{x} - \mathbf{x}^*$

b) $r(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|$

c) $r(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^T \mathbf{f}(\mathbf{x})$

d) $\mathbf{r}(\mathbf{x}) = \mathbf{e}(\mathbf{x}) = \mathbf{x} - \mathrm{Pr}_{\chi}[\mathbf{x} - \mathbf{f}(\mathbf{x})]$

No caso c), se a função objetivo satifazer (5.14), $r(\mathbf{x}) = 0$ se e somente se $\mathbf{x} = \mathbf{x}^*$.

O desconhecimento *a priori* do ponto ótimo $\mathbf{x}^*$ impossibilita a utilização dos três primeiros resíduos no cálculo do comprimento do passo. No caso do quarto resíduo, seu cálculo exige a utilização da projeção ortogonal, o que pretendemos evitar aqui.

Tentaremos, portanto, escolher um outro valor de resíduo que não apresente os problemas mencionados.

5.8.1 Cálculo do comprimento do passo por controle ótimo de Liapunov

Seja a lei de atualização das variáveis

$$\mathbf{u}_k := -\left[\sigma(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) + D_{\mathbf{g}}^T(\mathbf{x}_k)\Lambda \mathrm{uhsgn}(\mathbf{g}(\mathbf{x}_k)) + D_{\mathbf{h}}^T(\mathbf{x}_k)\Gamma \mathrm{sgn}(\mathbf{h}(\mathbf{x}_k))\right] \tag{5.75}$$

Então, a discretização de (5.71) por Euler, com comprimento do passo α_k pode ser escrita:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k \tag{5.76}$$

O método LOC será utilizado para otimizar o cálculo de α_k.

Seguindo a abordagem utilizada no capítulo de otimização convexa, será escolhida como função de energia

$$E(\mathbf{x}) := \sigma(\mathbf{x})\|\mathbf{f}(\mathbf{x})\|_2 + \mathbf{g}^T(\mathbf{x})\Lambda \text{uhsgn}(\mathbf{g}(\mathbf{x})) + \mathbf{h}^T(\mathbf{x})\Gamma \text{sgn}(\mathbf{h}(\mathbf{x})) \tag{5.77}$$

a qual apresenta um lado direito com descontinuidades sobre a fronteira do conjunto viável.

Para fazer o cálculo do passo ótimo de iteração serão consideradas as trajetórias dentro e fora do conjunto viável.

5.8.1.1 Fase de alcançar o conjunto viável

Nesta fase $\mathbf{x}_k \notin \chi$.

Fora do conjunto viável, o primeiro termo do lado direito de (5.77) se anula por ser o fator $\sigma(\mathbf{x}_k) = 0$, resultando a função de energia nos termos de penalização. Escolhendo como resíduo nesta fase, esta função de energia:

$$r_k := E(\mathbf{x}_k) = \mathbf{g}^T(\mathbf{x}_k)\Lambda \text{uhsgn}(\mathbf{g}(\mathbf{x}_k)) + \mathbf{h}^T(\mathbf{x}_k)\Gamma \text{sgn}(\mathbf{h}(\mathbf{x}_k))$$

A lei de atualização das variáveis (5.75) nesta fase é $\mathbf{u}_k = -\nabla E(\mathbf{x}_k)$, onde, similarmente ao que acontece no caso contínuo, se anulam os termos que contêm as derivadas das funções descontínuas.

Calculando o resíduo na iteração seguinte, aproximando por Taylor até o termo de primeira ordem e utilizando (5.76):

$$r_{k+1} \simeq r_k + \nabla^T E(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = E(\mathbf{x}_k) - \alpha_k \nabla^T E(\mathbf{x}_k)\nabla E(\mathbf{x}_k)$$

Nesta fase, a função de Liapunov candidata discreta será escolhida como o quadrado do resíduo a ser minimizado:

$$V_k = r_k^2$$
$$\Rightarrow \ \Delta V_k = V_{k+1} - V_k = r_{k+1}^2 - r_k^2 \simeq$$
$$-2\alpha_k E(\mathbf{x}_k)\|\nabla E(\mathbf{x}_k)\|_2^2 + \alpha_k^2 \|\nabla E(\mathbf{x}_k)\|_2^4$$

Otimizando o comprimento do passo segundo o controle ótimo de Liapunov

$$\frac{\partial \Delta V_k}{\partial \alpha_k} = -2E(\mathbf{x}_k)\|\nabla E(\mathbf{x}_k)\|_2^2 + 2\alpha_k\|\nabla E(\mathbf{x}_k)\|_2^4 = 0$$

$$\Rightarrow \ \alpha_k = \frac{E(\mathbf{x}_k)}{\nabla^T E(\mathbf{x}_k)\nabla E(\mathbf{x}_k)} \tag{5.78}$$

$$\Rightarrow \ \Delta V_k \simeq -E^2(\mathbf{x}_k) = -V_k < 0$$

$$\Rightarrow \ \mathbf{x}_{k+1} = \mathbf{x}_k - \frac{E(\mathbf{x}_k)\nabla E(\mathbf{x}_k)}{\|\nabla E(\mathbf{x}_k)\|_2^2} \tag{5.79}$$

Observe-se que, se a aproximação por Taylor fosse exata, isto implicaria que $V_{k+1} = 0$, e portanto $\mathbf{x}_{k+1} \in \chi$, o que efetivamente acontece se o ponto $\mathbf{x}_k$ violar apenas uma restrição afim.

A trajetória, portanto, convergirá ao conjunto viável em um número finito de iterações.

5.8.1.2 Fase de convergência

Nesta fase $\mathbf{x}_k \in \chi$.

A partir de (5.22) e (5.23), e considerando (5.28), é possível provar que

$$\|\mathbf{e}(\mathbf{x})\| \leq \left\| \mathbf{f}(\mathbf{x}) + \frac{E_{rp}(\mathbf{x} - \mathbf{f}(\mathbf{x}))\nabla E_{rp}(\mathbf{x} - \mathbf{f}(\mathbf{x}))}{\nabla^T E_{rp}(\mathbf{x} - \mathbf{f}(\mathbf{x}))\nabla E_{rp}(\mathbf{x} - \mathbf{f}(\mathbf{x}))} \right\| \leq \|\mathbf{f}(\mathbf{x})\| \tag{5.80}$$

sendo $E_{rp}(\mathbf{x}) = \mathbf{g}^T(\mathbf{x})\Lambda \text{uhsgn}(\mathbf{g}(\mathbf{x})) + \mathbf{h}^T(\mathbf{x})\Gamma \text{sgn}(\mathbf{h}(\mathbf{x}))$, os termos de penalização (dois últimos termos do lado direito) de (5.77). Se $\mathbf{x} - \mathbf{f}(\mathbf{x}) \in \chi$ todos os membros da desigualdade resultam iguais a $\|\mathbf{f}(\mathbf{x})\|$.

Isto significa que, por enquanto a trajetória permanecer dentro do conjunto viável, a norma do erro será igual à norma da função objetivo. Portanto, será escolhido como resíduo nesta fase a própria função objetivo.

A lei de atualização das variáveis (5.75), nesta fase é dada por $\mathbf{u}_k = -\mathbf{f}(\mathbf{x}_k)$.

Portanto:

$$\mathbf{r}_k := \mathbf{f}(\mathbf{x}_k)$$

$$\Rightarrow \ \mathbf{r}_{k+1} \simeq \mathbf{r}_k + D_{\mathbf{f}}^T(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$$

A função de Liapunov candidata discreta nesta fase também será escolhida como o quadrado da norma do resíduo a ser minimizado:

$$\begin{gathered} V_k = \|\mathbf{r}_k\|^2 \\ \Rightarrow\ \Delta V_k = V_{k+1} - V_k = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} - \mathbf{r}_k^T \mathbf{r}_k \simeq \\ -2\alpha_k \mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k) + \alpha_k^2 \|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|^2 \end{gathered}$$

Otimizando o comprimento do passo segundo o controle ótimo de Liapunov (derivando ΔV_k com respeito a α_k e igualando a zero)

$$\alpha_k = \frac{\mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|^2} \tag{5.81}$$

$$\Rightarrow\ \Delta V_k \simeq -\frac{\left[\mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)\right]^2}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|^2} \leq 0$$

$$\Rightarrow\ \mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|^2} \mathbf{f}(\mathbf{x}_k) \tag{5.82}$$

5.8.1.3 Considerações sobre o passo de iteração na fase de convergência

Considerando o mapeamento $\mathbf{f}(\mathbf{x})$ estritamente monôtono com respeito ao conjunto viável χ, e usando a aproximação $\mathbf{f}(\mathbf{x}_{k+1}) \simeq \mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$ (a qual é exata no caso de $\mathbf{f}(\mathbf{x})$ ser um mapeamento afim):

$$\begin{gathered} (\mathbf{x}_{k+1} - \mathbf{x}_k)^T \left[\mathbf{f}(\mathbf{x}_{k+1}) - \mathbf{f}(\mathbf{x}_k)\right] > 0 \\ \Rightarrow\ -\alpha_k \mathbf{f}^T(\mathbf{x}_k) \left[-\alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\right] > 0 \\ \Rightarrow\ \mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) > 0 \end{gathered} \tag{5.83}$$

Este resultado prova que, neste caso, $\Delta V_k \simeq -\frac{\left[\mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\right]^2}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|^2} < 0$ e portanto $\|\mathbf{f}(\mathbf{x}_k)\|$ é monotônicamente decrescente por enquanto a trajetória permanecer dentro do conjunto viável.

A partir de (5.81) e (5.83), pode-se concluir que

$$0 < \alpha_k \leq \frac{\|\mathbf{f}(\mathbf{x}_k)\|}{\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\|}$$

Para esses valores de α_k, considerando (5.83), pode se provar que

$$0 \leq \|\mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\| < \|\mathbf{f}(\mathbf{x}_k)\| \tag{5.84}$$

Supondo agora $\mathbf{f}(\mathbf{x})$ mapeamento não expansivo

$$\begin{aligned} &\|\mathbf{f}(\mathbf{x}_{k+1}) - \mathbf{f}(\mathbf{x}_k)\| \leq \|\mathbf{x}_{k+1} - \mathbf{x}_k\| \\ \Rightarrow\ &\| - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\| \leq \| - \alpha_k \mathbf{f}(\mathbf{x}_k)\| \\ \Rightarrow\ &\|D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\| \leq \|\mathbf{f}(\mathbf{x}_k)\| \end{aligned} \tag{5.85}$$

A seguinte figura ilustra as possíveis localizações dos vetores $\mathbf{f}(\mathbf{x}_k) - D_{\mathbf{f}}^T(\mathbf{x_k})\mathbf{f}(\mathbf{x}_k)$ e $\mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$ para um determinado vetor $\mathbf{f}(\mathbf{x}_k)$ em duas dimensões supondo o mapeamento estritamente monôtono e não expansivo.

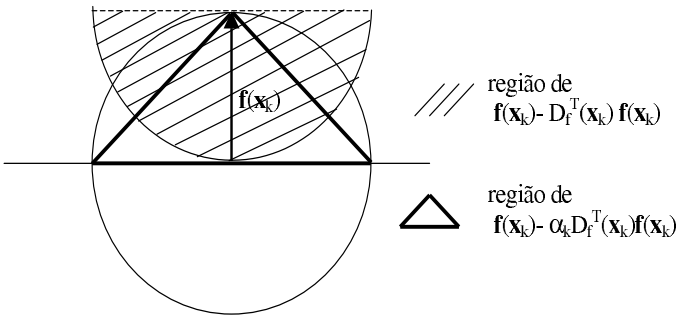

Figura 5.8: Localizações dos diferentes vetores para o α_k escolhido supondo o mapeamento $\mathbf{f}(\mathbf{x})$ estritamente monôtono e não expansivo

5.8.1.4 Análise dos erros na fase de convergência

Caso a) Será analisada, em primeiro lugar, a convergência dos erros para trajetórias dentro do conjunto viável ($\mathbf{x}_{k+1} \in \chi$).

Definiremos o erro como [38]: $\mathbf{e}(\mathbf{x}_k, \beta) = \mathbf{x}_k - \mathrm{Pr}_\chi\left[\mathbf{x}_k - \beta\mathbf{f}(\mathbf{x}_k)\right]$, para algum $\beta > 0$. Por (5.28), sabe-se que para todo $\mathbf{x}_k \in \chi:\ \|\mathbf{e}(\mathbf{x}_k, \beta)\| \leq \beta\|\mathbf{f}(\mathbf{x}_k)\|$.

Por ser χ convexo, existe $0 < \beta \leq \alpha_k$ tal que $\mathbf{x}_k - \beta\mathbf{f}(\mathbf{x}_k) \in \chi$.

Portanto:

$$\begin{aligned}\|\mathbf{e}(\mathbf{x}_{k+1},\beta)\| &= \|\mathbf{x}_{k+1} - \mathrm{Pr}_{\chi}\left[\mathbf{x}_{k+1} - \beta\mathbf{f}(\mathbf{x}_{k+1})\right]\| \leq \beta\|\mathbf{f}(\mathbf{x}_{k+1})\| \\ &\simeq \beta\|\mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\| < \beta\|\mathbf{f}(\mathbf{x}_k)\| \\ &= \|\mathbf{x}_k - \mathrm{Pr}_{\chi}\left[\mathbf{x}_k - \beta\mathbf{f}(\mathbf{x}_k)\right]\| = \|\mathbf{e}(\mathbf{x}_k,\beta)\|\end{aligned}$$

onde foi considerado (5.84) (e portanto o mapeamento estritamente monôtono) e que, neste caso, $\mathrm{Pr}_{\chi}\left[\mathbf{x}_k - \beta\mathbf{f}(\mathbf{x}_k)\right] = \mathbf{x}_k - \beta\mathbf{f}(\mathbf{x}_k)$.

Portanto, $\|\mathbf{e}(\mathbf{x}_k,\beta)\|$ decresce monotônicamente a cada iteração quando a trajetória permanece dentro do conjunto viável para uma função objetivo estritamente monôtona.

Caso b) O caso $\mathbf{x}_{k+1} \notin \chi$ resulta em maior complexidade.

A lei de atualização das variáveis (5.76) com os comprimentos dos passos (5.78) e (5.81) garante que, uma vez que as trajetórias saem do conjunto viável, retornarão a este percorrendo uma trajetória determinada por uma combinação linear positiva dos gradientes das restrições não observadas (equação 5.79).

Se o ponto de retorno ao conjunto viável for igual à projeção ortogonal do ponto $\mathbf{x}_k - \alpha_k\mathbf{f}(\mathbf{x}_k)$ sobre este conjunto, o algoritmo resulta similar àquele apresentado por Goldstein-Levitin-Polyak (5.56), o qual possui uma lei de atualização das variáveis: $\mathbf{x}_{k+1} = \mathrm{Pr}_{\chi}[\mathbf{x}_k - \alpha_k\mathbf{f}(\mathbf{x}_k)]$, sendo a diferença a escolha do comprimento do passo. Neste caso, a observância das condições (5.57) e (5.59) é suficiente para garantir que as trajetórias determinadas por (5.76) convergem à solução $\mathbf{x}^*$ (como demonstrado no teorema 5.5.4).

Porém, para que o ponto de retorno, passadas p iterações, seja igual à projeção ortogonal sobre o conjunto viável do ponto de partida, isto é $\mathbf{x}_{k+p} = \mathrm{Pr}_{\chi}[\mathbf{x}_{k+1}]$, este ponto de partida $\mathbf{x}_{k+1}$ e as restrições violadas devem observar as seguintes condições:

a) Se $\mathbf{x}_{k+1}$ viola restrições de igualdade, este ponto não deve violar nenhuma restrição de desigualdade.

b) Se $\mathbf{x}_{k+1}$ não viola nenhuma restrição de igualdade, este ponto não deve violar mais de uma restrição de desigualdade.

c) As restrições de desigualdade devem ser afins ou esféricas (as de igualdade sempre são afins por hipótese), o que provoca que ao longo de uma reta determinada por um vetor gradiente, todos os vetores gradientes tenham a mesma direção.

Para poder observar as condições a) e b), calcula-se o número ℓ = número de

restrições violadas no ponto $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)$. Se ℓ for maior que o máximo entre o número de restrições de igualdade violadas e 1, se divide o comprimento do passo por este número: $\alpha_k := \alpha_k/\ell$, e se repite o processo até $\ell =$ máximo entre o número de restrições de igualdade violadas e 1.

Por construção do algoritmo, se uma única restrição for violada quando a trajetória sai do conjunto viável, e esta for afim (seja esta restrição de desigualdade ou de igualdade), o ponto calculado na iteração seguinte já será a projeção ortogonal do ponto de partida sobre o conjunto viável, isto é, $\mathbf{x}_{k+2} = \mathrm{Pr}_\chi[\mathbf{x}_{k+1}]$.

Em geral, se as restrições observam o condicionamento c), existe $p > 1$ tal que $\mathbf{x}_{k+p} = \mathrm{Pr}_\chi[\mathbf{x}_{k+1}] \in \chi$. Se a restrição violada for afim, $p = 2$.

A seguinte figura ilustra a possível localização dos pontos $\mathbf{x}_k$, $\mathbf{x}_{k+1}$ e $\mathbf{x}_{k+p}$, para o caso da restrição violada ser de desigualdade e sua curva de nível, uma circunferência.

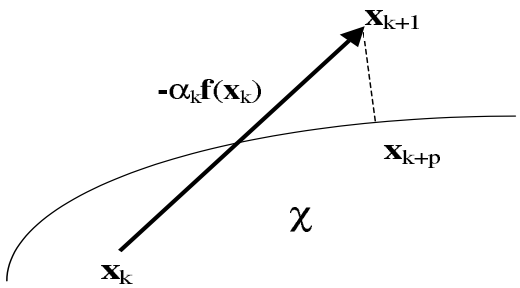

Figura 5.9: Localização dos pontos representados

Neste caso, o erro pode ser definido como $\mathbf{e}(\mathbf{x}_k, \beta) := \mathbf{x}_k - \mathrm{Pr}_\chi[\mathbf{x}_k + \beta(\mathbf{x}_{k+p} - \mathbf{x}_k)]$ para algum $\beta > 0$.

Observe-se que, se $\mathbf{x}_{k+1} \in \chi$, então $\mathbf{x}_{k+p} = \mathbf{x}_{k+1}$ e a situação corresponde ao caso (a) para algum $0 < \beta \leq 1$.

Observe-se também que:

Se $0 < \beta \leq 1 \;\Rightarrow\; \mathbf{x}_k + \beta(\mathbf{x}_{k+p} - \mathbf{x}_k) \in \chi \;\Rightarrow\; \|\mathbf{e}(\mathbf{x}_k, \beta)\| = \beta\|\mathbf{x}_{k+p} - \mathbf{x}_k\|$.

Se $\beta > 1 \;\Rightarrow\; \mathbf{x}_k + \beta(\mathbf{x}_{k+p} - \mathbf{x}_k) \notin \chi \;\Rightarrow\; \|\mathbf{e}(\mathbf{x}_k, \beta)\| < \beta\|\mathbf{x}_{k+p} - \mathbf{x}_k\|$.

Esta definição de erro é válida, pois, $\mathbf{e}(\mathbf{x}_k, \beta) = \mathbf{0}$ se e somente se $\mathbf{x}_k = \mathbf{x}_{k+p} = \mathbf{x}^*$.

Pode se verificar com facilidade que a exigência da função objetivo ser fracamente co-coerciva dentro do conjunto viável, com uma função que satisfaça (5.59), não é

muito restritiva. No caso do exemplo 2 apresentado anteriormente, a função objetivo $\mathbf{f}(\mathbf{x}) = \nabla((x_1-1)^2 - x_2 + 2)$ só não satisfaz a condição no intervalo $0.646 < x_1 < 1.354$, observando-a dentro do conjunto viável. A exigência que apresenta maior dificuldade em ser observada é aquela referida às restrições de desigualdade (condição c). Por exemplo, $g_1(\mathbf{x}) = x_1 + x_2^2 + 1 \leq 0$, também presente no exemplo 2, não é nem afim nem esférica.

Entretanto, na prática, o algoritmo mostra-se convergente também na presença desta restrição, embora o ponto de retorno da trajetória ao conjunto viável, quando esta restrição é violada, não corresponda à projeção ortogonal sobre este conjunto, isto é $\mathbf{x}_{k+p} \neq \Pr_{\chi}[\mathbf{x}_{k+1}]$ (ver problema 2).

Por exemplo, se $\mathbf{x}_{k+1} = [-1 \;\; 1/2]^T \notin \chi$, $g_1(\mathbf{x}_{k+1}) > 0$, então $\mathbf{x}_{k+5} = [-1.1350 \;\; 0.3675] \in \chi$ e $\Pr_{\chi}[\mathbf{x}_{k+1}] = [-1.1486 \;\; 0.3854]^T$.

O motivo desta convergência, mesmo não se verificando a condição c) pode ser entendida da seguinte maneira. Se a curvatura da restrição violada no ponto $\mathbf{x}_{k+1}$ (no caso do exemplo 2, $g_1(\mathbf{x}_{k+1})$), tiver o mesmo sentido da curvatura desta restrição no ponto de saída, $(g_1(\mathbf{x}_k))$, com respeito à função objetivo nesse ponto $\mathbf{f}(\mathbf{x}_k)$, então o ponto de retorno estará mais próximo da solução do problema que o ponto de partida, isto é $\|\mathbf{x}_{k+p} - \mathbf{x}^*\| < \|\mathbf{x}_k - \mathbf{x}^*\|$. A condição para garantir esta convergência deve ser similar à de co-coercividade fraca mencionada anteriormente.

Para conferir se as curvaturas tem o mesmo sentido no ponto de saída do conjunto viável $\mathbf{x}_k$ e no seguinte fora do conjunto viável $\mathbf{x}_{k+1}$, projeta-se o gradiente da restrição violada sobre o plano normal ao vetor da função objetivo:

$$\mathbf{v} = \left[I - \frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} \right] \nabla g_j(\mathbf{x}_k) \tag{5.86}$$

sendo g_j a restrição violada no ponto $\mathbf{x}_{k+1}$ e $\mathbf{v}$ a componente do vetor $\nabla g_j(\mathbf{x}_k)$ normal a $\mathbf{f}(\mathbf{x}_k)$. Caso o vetor $\nabla g_j(\mathbf{x}_{k+1})$, esteja do mesmo lado do hiperplano determinado pelo vetor $\mathbf{v}$ (isto é $\nabla^T g_j(\mathbf{x}_{k+1})\mathbf{v} \geq 0$), então as curvaturas tem o mesmo sentido com respeito ao vetor $\mathbf{f}(\mathbf{x}_k)$, caso contrário tem sentidos opostos.

Neste último caso, pode se refletir o vetor $\nabla g_j(\mathbf{x}_{k+1})$ sobre o vetor $\mathbf{f}(\mathbf{x}_k)$, através do cálculo

$$\mathbf{u} = \left[2\frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} - I \right] \nabla g_j(\mathbf{x}_{k+1}) \tag{5.87}$$

e este novo vetor $\mathbf{u}$ terá o mesmo sentido que $\nabla g_j(\mathbf{x}_k)$ com respeito à função objetivo $\mathbf{f}(\mathbf{x}_k)$.

Prova desta afirmação:

Se $\nabla^T g_j(\mathbf{x}_{k+1})\mathbf{v} = \nabla^T g_j(\mathbf{x}_{k+1}) \left[I - \frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} \right] \nabla g_j(\mathbf{x}_k) < 0$

Portanto

$$\begin{aligned}
\mathbf{u}^T\mathbf{v} &= \nabla^T g_j(\mathbf{x}_{k+1}) \left[2\frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} - I \right] \left[I - \frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} \right] \nabla g_j(\mathbf{x}_k) \\
&= -\nabla^T g_j(\mathbf{x}_{k+1}) \left[I - 3\frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} + 2\frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} \right] \nabla g_j(\mathbf{x}_k) \\
&= -\nabla^T g_j(\mathbf{x}_{k+1}) \left[I - \frac{\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)}{\|\mathbf{f}(\mathbf{x}_k)\|^2} \right] \nabla g_j(\mathbf{x}_k) > 0
\end{aligned}$$

Portanto, se este for tomado como lei de atualização das variáveis neste caso, o ponto de retorno $\mathbf{x}_{k+p}$ também estará mais próximo da solução $\mathbf{x}^*$ que o ponto de partida $\mathbf{x}_k$, sob as mesmas condições que no caso anterior.

A figura 5.10 ilustra os dois casos mencionados, com os vetores respectivos.

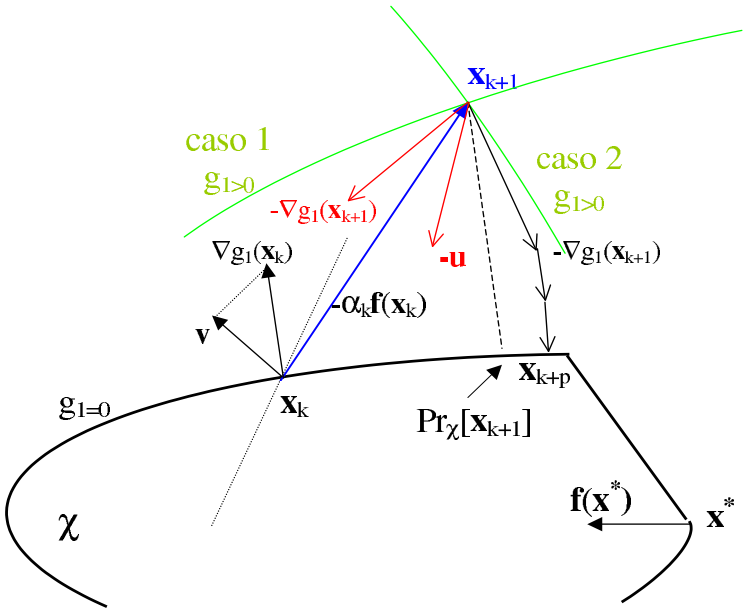

Figura 5.10: Representação dos pontos de saída e retorno do conjunto viável e dos vetores de atualização das variáveis em ambos casos

Observe-se que o ponto $\mathbf{x}_k$ não necessariamente deve estar na fronteira do conjunto viável, como ilustrado na figura 5.10. Inclusive, caso este esteja no seu interior, é possível que a solução $\mathbf{x}^*$ esteja do outro lado do vetor $-\alpha_k \mathbf{f}(\mathbf{x}_k)$, em cujo caso o ponto de

retorno $\mathbf{x}_{k+p}$ se afastará da solução ao invês de se aproximar. Mas isto acontecerá apenas em uma iteração, porque para as iterações seguintes, para todo k tal que $\mathbf{x}_k \in \chi$, então $\mathbf{x}_k \in \partial\chi$, que é o caso ilustrado na figura.

Esta lei de atualização das variáveis garante que o ponto de retorno $\mathbf{x}_{k+p}$ estará do mesmo lado de $\mathbf{x}^*$, com respeito a um hiperplano separador contendo o vetor $\mathbf{f}(\mathbf{x}_k)$ com direção $\mathbf{v}$. Isto é, $\mathbf{v}^T(\mathbf{x}_k - \mathbf{x}_{k+p}) > 0$ o que implica

$$(\mathbf{x}_k - \mathbf{x}_{k+p})^T(\mathbf{x}_k - \mathbf{x}^*) > 0 \tag{5.88}$$

Esta condição é observada mesmo que a função objetivo não seja fracamente co-coerciva, e portanto mesmo que $\|\mathbf{x}_k - \mathbf{x}^*\| < \|\mathbf{x}_{k+p} - \mathbf{x}^*\|$.

A desigualdade (5.88) indica que um hiperplano separador pode ser estabelecido na direção $\mathbf{x}_k - \mathbf{x}^*$, passando por algum ponto entre $\mathbf{x}_k$ e $\mathbf{x}_{k+p}$, por exemplo

$$H_k = \{\mathbf{w} \in \mathbb{R}^n \mid (\mathbf{x}_k - \mathbf{x}^*)^T(\mathbf{w} - \mathbf{x}_k) + \epsilon(\mathbf{x}_k - \mathbf{x}_{k+p})^T(\mathbf{x}_k - \mathbf{x}^*) = 0\} \tag{5.89}$$

para algum $\epsilon > 0$.

No ponto $\mathbf{w} = \mathbf{x}_k$: $\quad \epsilon(\mathbf{x}_k - \mathbf{x}_{k+p})^T(\mathbf{x}_k - \mathbf{x}^*) > 0$

No ponto $\mathbf{w} = \mathbf{x}_{k+p}$: $\quad (-1+\epsilon)(\mathbf{x}_k - \mathbf{x}^*)^T(\mathbf{x}_k - \mathbf{x}_{k+p}) < 0 \quad \forall \epsilon < 1$

Portanto para todo $0 < \epsilon < 1$, (5.89) é um hiperplano que separa estritamente $\mathbf{x}_{k+p}$ e $\mathbf{x}^*$ do ponto $\mathbf{x}_k$.

Também aqui, esta condição se verifica mesmo que a função objetivo não seja fracamente co-coerciva com a função de co-coercividade satisfazendo (5.59), sendo suficiente com exigir (5.14), que é uma condição mais fraca.

Para que o seguinte ponto dentro do conjunto viável, chamado $\mathbf{x}_{k+p_2}$, esteja do mesmo lado desse hiperplano separador, é suficiente com exigir a condição:

$$\forall \mathbf{x},\ \mathbf{y} \in \chi : \quad \mathbf{f}^T(\mathbf{x})(\mathbf{x} - \mathbf{y}) > 0 \ \Rightarrow\ \mathbf{f}^T(\mathbf{y})(\mathbf{x} - \mathbf{y}) > 0 \tag{5.90}$$

Esta condição é observada, por exemplo, por mapeamentos afins da forma $\mathbf{f}(\mathbf{x}) = \mathbf{q} + M\mathbf{x}$, onde a matriz M é negativa definida ([26, p. 7]).

Observe-se que esta é a condição exigida por Solodov em [71] e apresentada aqui na equação (5.60) . O hiperplano dado por (5.61), $H_{x_{k+p}} = \{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{f}^T(\mathbf{x}_{k+p})(\mathbf{w} - \mathbf{x}_{k+p}) = 0\}$, separa estritamente $\mathbf{x}_k$ de $\mathbf{x}^*$, pelas condições (5.90) e (5.14).

A projeção de $\mathbf{x}_k$ sobre este hiperplano se aproxima da solução $\mathbf{x}^*$, como demonstra Solodov (lema 5.5.5).

A condição (5.90) evita que o seguinte ponto dentro do conjunto viável retorne do mesmo lado do hiperplano separador, como ilustrado na seguinte figura:

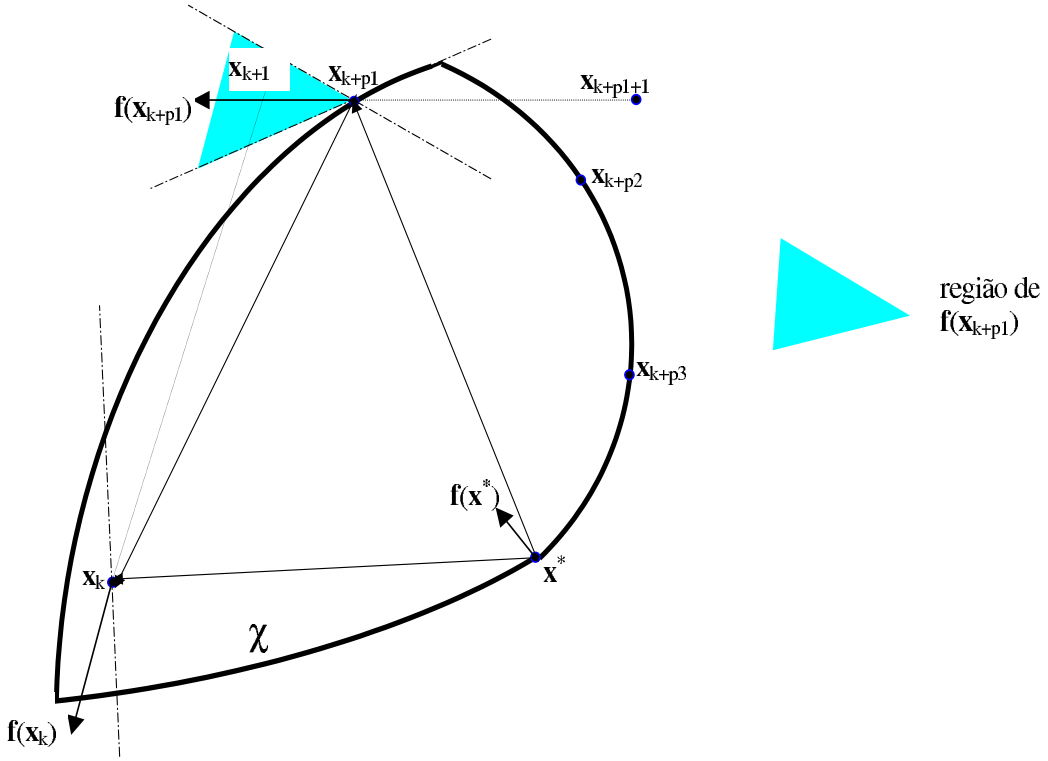

Figura 5.11: Representação dos pontos calculados dentro do conjunto viável mostrando o decrescimento da distância entre $\mathbf{x}^*$ e o ponto mais próximo dos sucessivos hiperplanos separadores

5.8.2 Critério de parada

Por construção do algoritmo, existe uma iteração k tal que $\mathbf{x}_k = \mathbf{x}^*$, e portanto, $\mathbf{x}_{k+p} = \mathbf{x}_k$. Na prática, o algoritmo pode ser iterado até atingir um ponto $\mathbf{x}_k \in \chi$ arbitrariamente perto do último ponto calculado dentro do conjunto viável. Isto é, dado um valor arbitrariamente pequeno $\epsilon > 0$, iterar k vezes até que $\mathbf{x}_k \in \chi$ e $\|\mathbf{x}_k - \mathbf{x}_{k-r}\| < \epsilon$, sendo $\mathbf{x}_{k-r}$ o último ponto calculado dentro do conjunto viável.

5.8.3 Implementação do algoritmo

A seguir, será apresentada a forma de implementar o algoritmo passo a passo.

0) Escolher x_0 e ϵ. Determinar $k = 0$ e $\mathbf{x}_{ant} = -\infty$

1) Se $\sigma(\mathbf{x}_k) = 1$

$\alpha_k = \frac{\mathbf{f}^T(\mathbf{x}_k) D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k)}{\| D_{\mathbf{f}}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k) \|^2}$; $\quad \mathbf{x}_{ant} = \mathbf{x}_k$

1.1) $\mathbf{x}_r = \mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)$

1.2) Se $\sigma(\mathbf{x}_r) = 0$

1.2.1) Se $p > 0$

$ec = 0$

for $i = 1 : p$,

se $h_i(\mathbf{x}_r) \neq 0$ then $ec = ec + 1$;

end;

1.2.2) Se $m > 0$

$ic = 0$; $\quad rv = 0$

for $j = 1 : m$,

se $g_j(\mathbf{x}_r) > 0$ then $ic = ic + 1$; $rv = j$

end;

1.2.3) $\ell = ec + ic$

1.2.4) se $\ell > \max(ec, 1)$

$\alpha_k := \alpha_k / \ell$

voltar passo 1.1

2) Se $\sigma(\mathbf{x}_k) = 0$

$\alpha_k = \frac{E(\mathbf{x}_k)}{\| \nabla E(\mathbf{x}_k) \|^2}$

2.1) Se $(rv > 0)$ e $(\mathbf{x}_{ant} \neq -\infty)$ e $\left(\nabla^T g_{rv}(\mathbf{x}_k) \left[I - \frac{\mathbf{f}(\mathbf{x}_{ant}) \mathbf{f}^T(\mathbf{x}_{ant})}{\| \mathbf{f}(\mathbf{x}_{ant}) \|^2} \right] \nabla g_{rv}(\mathbf{x}_{ant}) < 0 \right)$

$\alpha_k := \alpha_k \left[2 \frac{\mathbf{f}(\mathbf{x}_{ant}) \mathbf{f}^T(\mathbf{x}_{ant})}{\| \mathbf{f}(\mathbf{x}_{ant}) \|^2} - I \right]$

2.2) $\mathbf{x}_r = \mathbf{x}_k - \alpha_k \nabla E(\mathbf{x}_k)$

3) $k = k + 1$; $\quad \mathbf{x}_k = \mathbf{x}_r$

4) Se $\sigma(\mathbf{x}_k) = 1$ e $\| \mathbf{x}_k - \mathbf{x}_{ant} \| < \epsilon$

parar e $\mathbf{x}^* = \mathbf{x}_k$

se não voltar passo 1

Neste algoritmo, chamamos de $\mathbf{x}_r$ ao candidato a próximo ponto (que efetivamente o será caso este ponto observe as condições a) e b) anteriormente citadas); ec às restrições de igualdade não observadas no ponto $\mathbf{x}_r$, ic às restrições de desigualdade não observadas neste ponto; ℓ é o número total de restrições não observadas; rv é o índice da restrição de desigualdade não observada (caso haja alguma, caso contrário este valor permanece em 0). Mantemos a nomenclatura de m e p para o número total de restrições de desigualdade e de igualdade, respectivamente.

5.8.4 Prova da convergência do algoritmo

Para o algoritmo apresentado na seção anterior, cuja função objetivo satisfaz as condições (5.14) e (5.90), dado um ponto $\mathbf{x}_k \in \chi$, chamaremos $\mathbf{x}_{k+pi}$ ao i-ésimo ponto seguinte dentro do conjunto viável.

Observe-se que este ponto pode corresponder a um ponto de retorno ao conjunto viável (no caso que $\mathbf{x}_{k+pi-1} \notin \chi$), ou a uma iteração dentro do conjunto viável (no caso que $\mathbf{x}_{k+pi-1} \in \chi$). Esta diferença não afeta a prova de convergência.

Por construção do algoritmo, de (5.88):

$$(\mathbf{x}_k - \mathbf{x}^*)^T(\mathbf{x}_k - \mathbf{x}_{k+p1}) > 0 \tag{5.91}$$

Pela condição (5.90):

$$(\mathbf{x}_k - \mathbf{x}_{k+p1})^T(\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2}) > 0 \tag{5.92}$$

Aplicando a condição (5.91) ao ponto $\mathbf{x}_{k+p1}$:

$$(\mathbf{x}_{k+p1} - \mathbf{x}^*)^T(\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2}) > 0 \tag{5.93}$$

O que implica

$$
\begin{aligned}
& (\mathbf{x}_k - \mathbf{x}_{k+p2}) = (\mathbf{x}_k - \mathbf{x}_{k+p1}) + (\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2}) \\
\Rightarrow \quad & (\mathbf{x}_k - \mathbf{x}_{k+p2})^T(\mathbf{x}_k - \mathbf{x}^*) = \\
& (\mathbf{x}_k - \mathbf{x}_{k+p1})^T(\mathbf{x}_k - \mathbf{x}^*) + (\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2})^T(\mathbf{x}_k - \mathbf{x}^*) = \\
& (\mathbf{x}_k - \mathbf{x}_{k+p1})^T(\mathbf{x}_k - \mathbf{x}^*) + (\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2})^T(\mathbf{x}_k - \mathbf{x}_{k+p1}) + \\
& (\mathbf{x}_{k+p1} - \mathbf{x}_{k+p2})^T(\mathbf{x}_{k+p1} - \mathbf{x}^*) > 0
\end{aligned}
$$

onde foram usados (5.91), (5.92) e (5.93). Portanto

$$
(\mathbf{x}_k - \mathbf{x}_{k+p2})^T(\mathbf{x}_k - \mathbf{x}^*) > (\mathbf{x}_k - \mathbf{x}_{k+p1})^T(\mathbf{x}_k - \mathbf{x}^*) > 0 \tag{5.94}
$$

Construindo hiperplanos paralelos determinados pelo vetor $\mathbf{x}_k - \mathbf{x}^*$, passantes pelos pontos $\mathbf{x}_{k+pi}$:

$$
H_i = \{\mathbf{w} \in \mathbb{R}^n \mid (\mathbf{x}_k - \mathbf{x}^*)^T(\mathbf{w} - \mathbf{x}_{k+pi}) = 0\} \tag{5.95}
$$

define-se a distância entre o ponto ótimo $\mathbf{x}^*$ e cada um destes hiperplanos como

$$
\text{dist}(\mathbf{x}^*, H_i) = \left| \frac{(\mathbf{x}_{k+pi} - \mathbf{x}^*)^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| \tag{5.96}
$$

Portanto:

$$
\begin{aligned}
\text{dist}(\mathbf{x}^*, H_i) &= \left| \tfrac{(\mathbf{x}_{k+pi} - \mathbf{x}^*)^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| = \left| \tfrac{(\mathbf{x}_{k+pi} - \mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*) + (\mathbf{x}_k - \mathbf{x}^*)^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| \\
&= \left| \|\mathbf{x}_k - \mathbf{x}^*\| + \tfrac{(\mathbf{x}_{k+pi} - \mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| \\
&= \left| \|\mathbf{x}_k - \mathbf{x}^*\| - \tfrac{(\mathbf{x}_k - \mathbf{x}_{k+pi})^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| \\
&> \left| \|\mathbf{x}_k - \mathbf{x}^*\| - \tfrac{(\mathbf{x}_k - \mathbf{x}_{k+p_{i+1}})^T(\mathbf{x}_k - \mathbf{x}^*)}{\|\mathbf{x}_k - \mathbf{x}^*\|} \right| = \text{dist}(\mathbf{x}^*, H_{i+1})
\end{aligned}
$$

onde foi utilizado (5.94).

Isto demonstra que os sucessivos hiperplanos paralelos vão se aproximando do ponto ótimo. Por estarem os pontos $\mathbf{x}_{k+pi}$ que determinam tais hiperplanos dentro do conjunto viável, e o ponto ótimo $\mathbf{x}^*$ necessariamente estar sobre a fronteira desse conjunto (a não ser no caso trivial em que $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, em cujo caso o algoritmo convergirá pelo caso (a) mencionado anteriormente), $\mathbf{x}_{k+pi}$ tende a $\mathbf{x}^*$, o que demonstra a convergência do algoritmo.

5.9 Algoritmo projetado

O algoritmo apresentado na seção anterior, pode ter sua versão projetada sobre o gradiente das restrições violadas quando as trajetórias saírem do conjunto viável, tal como aplicado no capítulo referente a otimização convexa. Assim, uma vez que a trajetória entra no conjunto viável, só se afastará deste se, neste ponto, a fronteira for estritamente convexa, e não mais sairá se a restrição violada for afim. Assim, procura-se diminuir o chattering caracterísico das trajetórias em modo deslizante, o qual aumenta a previsibilidade da trajetória e a possibilidade de observar as condições a) e b) anteriormente citadas.

Dado um vetor $\mathbf{y}$, sabe-se que a matriz $P_{\mathbf{y}} := I - \frac{\mathbf{y}\mathbf{y}^T}{\|\mathbf{y}\|_2^2}$ projeta um vetor sobre o hiperplano normal a $\mathbf{y}$. Assim, definindo o operador de projeção:

$$\text{Proj}(\mathbf{u}(\mathbf{x}), \mathbf{y}) \begin{cases} \mathbf{u}(\mathbf{x}) & \text{se } \sigma(\mathbf{x})(1 - \sigma(\mathbf{x} + \beta\mathbf{u}(\mathbf{x}))) = 0 \\ P_{\mathbf{y}}\mathbf{u}(\mathbf{x}) & \text{se } \sigma(\mathbf{x}) = 1 \text{ e } \sigma(\mathbf{x} + \beta\mathbf{u}(\mathbf{x})) = 0 \end{cases} \tag{5.97}$$

sendo $\beta > 0$ um escalar suficientemente pequeno para garantir a violação de apenas uma restrição no ponto $\mathbf{x} - \beta\mathbf{u}(\mathbf{x})$.

Assim, chamando $\bar{\mathbf{x}}_k = \mathbf{x}_k + \beta\mathbf{u}_k$, se $\sigma(\mathbf{x}_k) = 1$ e $\sigma(\bar{\mathbf{x}}_k) = 0$, então existe j tal que $f_j(\mathbf{x}_k) \leq 0$ e $f_j(\bar{\mathbf{x}}_k) > 0$, ou $h_j(\mathbf{x}_k) = 0$ e $h_j(\bar{\mathbf{x}}_k) \neq 0$. A projeção será feita sobre o gradiente desta função, e o escalar β escolhido de maneira tal que $f_j(\bar{\mathbf{x}}_k)$ ou $h_j(\bar{\mathbf{x}}_k)$ estejam arbitrariamente próximos de zero.

Portanto:

$$\mathbf{x}_r = \mathbf{x}_k - \alpha_k \left(\mathbf{u}_k - \sigma(\mathbf{x}_k)(1 - \sigma(\bar{\mathbf{x}}_k)) \frac{\mathbf{u}_k^T \nabla E(\bar{\mathbf{x}}_k) \nabla E(\bar{\mathbf{x}}_k)}{\nabla^T E(\bar{\mathbf{x}}_k) \nabla E(\bar{\mathbf{x}}_k)} \right) \tag{5.98}$$

onde α_k é calculado como na seção anterior e $\mathbf{u}_k$ será $\mathbf{f}(\mathbf{x}_k)$ ou $\nabla E(\mathbf{x}_k)$, segundo estiver no passo 1 ou 2 na implementação do algoritmo.

5.10 Execução do algoritmo discreto

O algoritmo será executado com os problemas 1, 2, 3 e 4. Em todos os casos, o conjunto viável estará definido pela observância das restrições. A função objetivo será, em todos os casos $\nabla f_0(\mathbf{x})$, que por ser estritamente convexa em todos os casos o resultado único

é garantido. Os ganhos dos termos de penalização λ_i e ν_i continuam sendo unitários.

A continuação são apresentados os gráficos dos dois primeiros problemas utilizando o algoritmo discreto e sua versão projetada. Os números entre parêntesis indicam o número de iterações.

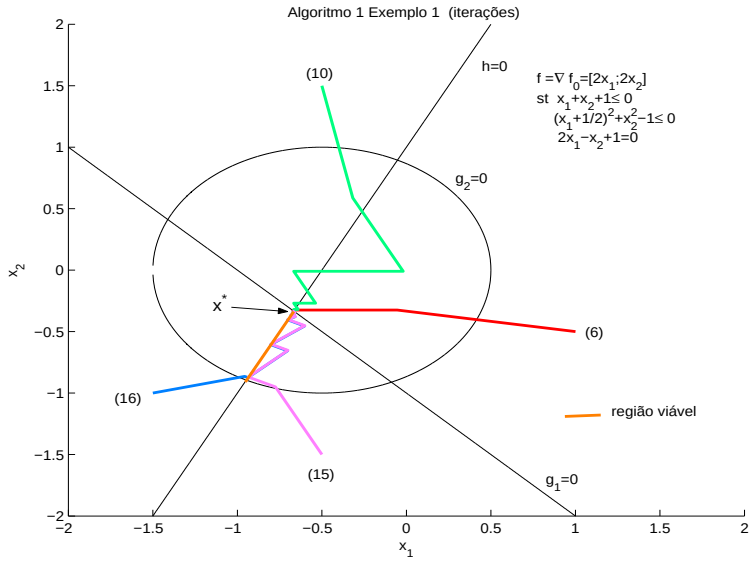

Figura 5.12: Algoritmo discreto aplicado ao exemplo 1 ilustrando a convergência das trajetórias

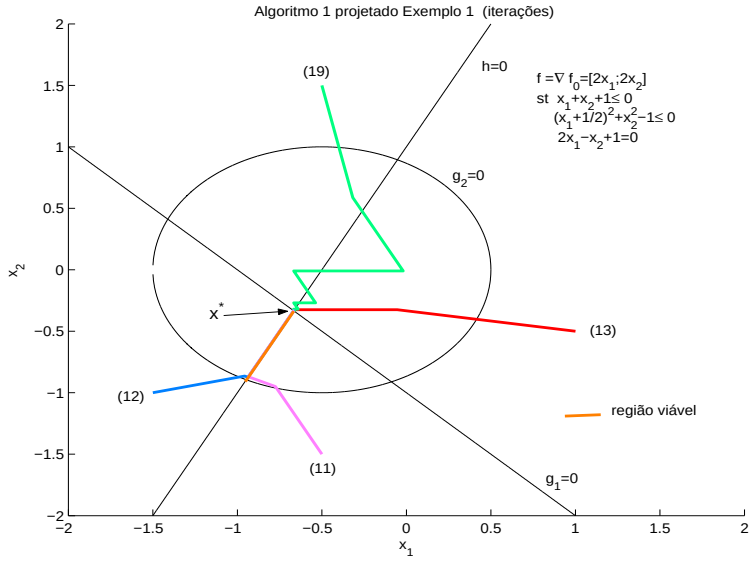

Figura 5.13: Algoritmo projetado aplicado ao exemplo 1 ilustrando a convergência das trajetórias

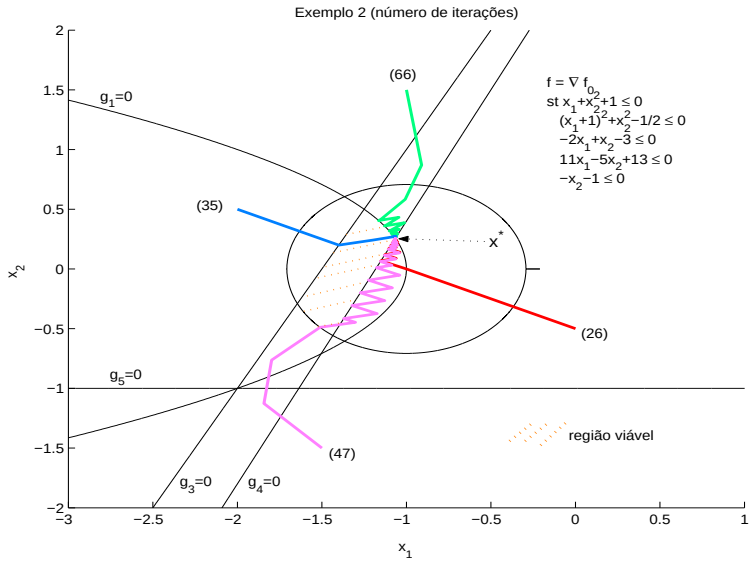

Figura 5.14: Algoritmo discreto aplicado ao exemplo 2 ilustrando a convergência das trajetórias

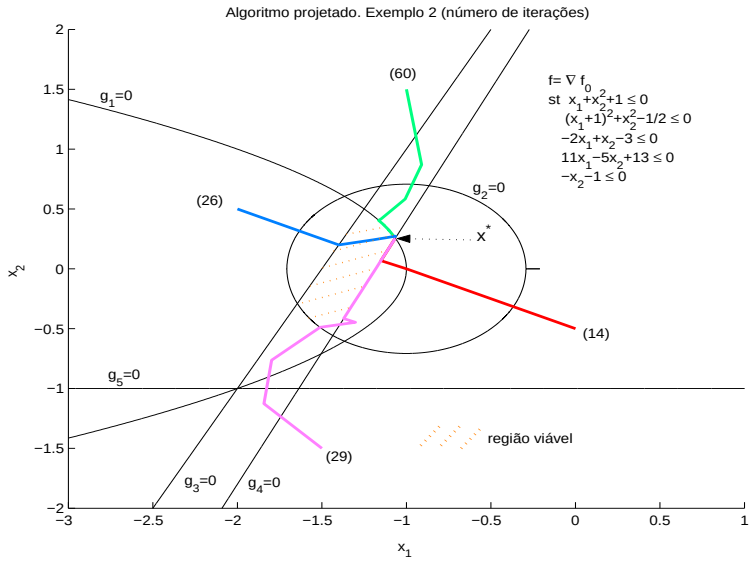

Figura 5.15: Algoritmo projetado aplicado ao exemplo 2 ilustrando a convergência das trajetórias

Os resultados da execução dos algoritmos para os problemas 3 e 4 serão apresentados posteriormente. Em todos os casos, o algoritmo discreto e sua versão projetada foram capazes de alcançar o ponto ótimo do problema. No caso da versão projetada, reduzindo ou mantendo ainda o número de iterações e sem exhibir o chattering característico das trajetórias em modo deslizante.

No caso do problema 2, a trajetória verde e em menor medida a azul, violam em alguns tramos a restrição $g_1(\mathbf{x}) \leq 0$, que, como apontado, é parabólica, não sendo

portanto o ponto de retorno igual à projeção ortogonal. A convergência do algoritmo, entretanto, não é afetada.

5.11 Discretização do algoritmo contínuo apresentado em [30], otimizando o comprimento do passo por LOC

Como tratado na seção 5.5.2, em [30] os autores propõem um algoritmo contínuo para resolver o problema VI($\mathbf{f}, \chi$), sendo $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ monôtona e $\chi = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$ e $\mathbf{h}(\mathbf{x}) = \mathbf{0}\}$ conjunto convexo viável tal que $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x})..g_m(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^m$ tal que $\forall\ i \in \{1,..,m\},\ g_i(\mathbf{x})$ convexa, contínua e com derivadas parciais contínuas e $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x})..h_p(\mathbf{x})]^T : \mathbb{R}^n \to \mathbb{R}^p$ tal que $\forall\ i \in \{1,..,p\},\ h_i(\mathbf{x})$ afim, contínua e com derivadas parciais contínuas.

Os autores planteam o problema VI($\mathbf{v}(\mathbf{u}), \Omega$), onde $\mathbf{u} = \left[\mathbf{x}^T\ \mathbf{y}^T\ \mathbf{z}^T\right]^T \in \mathbb{R}^{n+m+p}$, sendo $\mathbf{y}$ e $\mathbf{z}$ multiplicadores de Lagrange,

$$\mathbf{v}(\mathbf{u}) = \begin{bmatrix} \mathbf{f}(\mathbf{x}) + D^T_{\mathbf{g}}(\mathbf{x})\mathbf{y} + D^T_{\mathbf{h}}(\mathbf{x})\mathbf{z} \\ -\mathbf{g}(\mathbf{x}) \\ -\mathbf{h}(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{n+m+p} \tag{5.99}$$

e $\Omega = \{\mathbf{u} \in \mathbb{R}^{n+m+p} \mid \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{R}^m_+,\ \mathbf{z} \in \mathbb{R}^p\}$.

No lema 5.5.1, substituindo $\mathbf{f}(\mathbf{u})$ por $\mathbf{v}(\mathbf{u})$, $\nabla f_0(\mathbf{x})$ por $\mathbf{f}(\mathbf{x})$, e utilizando a monotonicidade de $\mathbf{f}(\mathbf{x})$ no lugar da convexidade de $f_0(\mathbf{x})$, é demonstrado que $\mathbf{v}(\mathbf{u})$ é monôtona em Ω.

No lema 5.5.2, fazendo as mesmas substituições, é demonstrado que se se verifica a condição de Slater, isto é, se existe $\mathbf{x}$ tal que $\mathbf{g}(\mathbf{x}) \prec \mathbf{0}$ e $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, então existe dualidade forte e portanto $\mathbf{u}^* = \left[\mathbf{x}^{*T}\ \mathbf{y}^{*T}\ \mathbf{z}^{*T}\right]^T$ satisfaz as condições KKT (5.45) se e somente se $\mathbf{u}^*$ é solução de VI($\mathbf{v}(\mathbf{u}), \Omega$), sendo $\mathbf{x}^*$ a solução de VI($\mathbf{f}(\mathbf{x}), \chi$), e $\mathbf{y}^*$ e $\mathbf{z}^*$ os multiplicadores de Lagrange.

Sobre o conjunto Ω é trivial calcular o operador de projeção ortogonal:

$$\mathrm{Pr}_{\Omega}[\mathbf{u}] = \begin{bmatrix} \mathbf{x} \\ \bar{\mathbf{y}} \\ \mathbf{z} \end{bmatrix} \quad \text{tal que} \quad \bar{y_i} = \begin{cases} y_i & \text{se } y_i \geq 0 \\ 0 & \text{se } y_i < 0 \end{cases} \tag{5.100}$$

Assim, é definida a seguinte lei de atualização das variáveis:

$$\mathbf{u}_{k+1} = \mathrm{Pr}_{\Omega}\left[\mathbf{u}_k - \alpha_k \mathbf{v}_k\right] \tag{5.101}$$

onde α_k é o comprimento do passo que será otimizado por controle ótimo de Lipunov (LOC).

Escolhendo como função de Liapunov candidata discreta $V_k = \|\mathbf{r}_k\|^2 = \|\mathbf{v}_k\|^2$, o comprimento do passo que minimiza ΔV_k é (ver seção 5.8.1.2)

$$\alpha_k = \frac{\mathbf{v}^T(\mathbf{u}_k) D_{\mathbf{v}}^T(\mathbf{u}_k) \mathbf{v}(\mathbf{u}_k)}{\|D_{\mathbf{v}}^T(\mathbf{u}_k)\mathbf{v}(\mathbf{u}_k)\|^2} \tag{5.102}$$

onde

$$D_{\mathbf{v}}(\mathbf{u}) = \begin{bmatrix} D_{\mathbf{f}}(\mathbf{x}) + \sum_{i=1}^{m} \nabla^2 g_i(\mathbf{x}) y_i + \sum_{i=1}^{p} \nabla^2 h_i(\mathbf{x}) z_i & D_{\mathbf{g}}^T(\mathbf{x}) & D_{\mathbf{h}}^T(\mathbf{x}) \\ -D_{\mathbf{g}}(\mathbf{x}) & 0 & 0 \\ -D_{\mathbf{h}}(\mathbf{x}) & 0 & 0 \end{bmatrix}$$
$$\in \mathbb{R}^{(n+m+p)\times(n+m+p)}$$

O teorema 5.5.4 demonstra que se $\mathbf{v}(\mathbf{u})$ for fracamente co-coerciva, isto é:

$$\forall \mathbf{u},\ \bar{\mathbf{u}} \in \Omega :\ \exists g(\mathbf{u}, \bar{\mathbf{u}}) \in \mathbb{R}_{++} \text{ tal que } (\mathbf{u} - \bar{\mathbf{u}})^T \left[\mathbf{v}(\mathbf{u}) - \mathbf{v}(\bar{\mathbf{u}})\right] \geq g(\mathbf{u}, \bar{\mathbf{u}}) \|\mathbf{v}(\mathbf{u}) - \mathbf{v}(\bar{\mathbf{u}})\|^2$$

e se a função $g(\mathbf{u}, \bar{\mathbf{u}}) > \frac{\alpha_k}{2}$, então a distância à solução $\mathbf{u}^*$ é monotonicamente decrescente, isto é

$$\|\mathbf{u}_{k+1} - \mathbf{u}^*\| < \|\mathbf{u}_k - \mathbf{u}^*\|$$

Este algoritmo é similar ao de Goldstein-Levitin-Polyak (5.56, mencionado em [38]

e [83]), apresentando a vantagem que o cálculo da projeção ortogonal é trivial sobre o conjunto Ω o que pode não acontecer sobre o conjunto viável χ, e que o comprimento do passo é calculado de maneira ótima a fim de minimizar norma quadrado da função $\mathbf{v}(\mathbf{u})$, que é igual ao resíduo ou erro dentro do conjunto viável.

A desvantagem, claro, consiste no aumento da dimensão do problema, como no caso contínuo apresentado em [30]. Uma outra desvantagem consiste na exigência do conhecimento dos hessianos das restrições e do jacobiano da função objetivo para o cálculo do comprimento do passo, nem sempre disponíveis.

5.12 Outros algoritmos discretos

Nesta seção serão deduzidos e testados, sem qualquer prova de convergência, variações sobre o algoritmo discreto anteriormente apresentado. Serão testadas também as suas versões projetadas.

O intuito destas variações é encontrar comprimentos dos passos diferentes que minimizem outros resíduos, assim também como evitar a avaliação recorrente das restrições toda vez que as trajetórias saem do conjunto viável, o que compromete a eficiência do algoritmo. Procura-se também evitar qualquer condicionamento sobre a forma das restrições assim também como sobre o número de restrições violadas cada vez que a trajetória sai do conjunto viável.

5.12.1 Algoritmo 2

Neste segundo algoritmo, mantêm-se o comprimento do passo (5.78) e a lei de atualização das variáveis (5.79) para a fase de alcançar o conjunto viável, escolhendo a expressão do meio da desigualdade (5.80) como resíduo a ser minimizado na fase de convergência. Assim

$$\mathbf{r}_k = \mathbf{f}(\mathbf{x}_k) + \frac{E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))}{\|\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\|^2}$$

$$
\begin{aligned}
\Rightarrow \mathbf{r}_{k+1} &\simeq \mathbf{r}_k - \alpha_k \Big[D_{\mathbf{f}}(\mathbf{x}_k) + \tfrac{\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) \nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))}{\|\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\|^2} + \\
&\quad \tfrac{E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) \nabla^2 E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))}{\|\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\|^2} - \\
&\quad \tfrac{2 \nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) \nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) \nabla^2 E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))}{\|\nabla E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\|^4} \Big]^T \mathbf{f}(\mathbf{x}_k) \\
&= \mathbf{r}_k - \alpha_k D_k^T \mathbf{f}(\mathbf{x}_k)
\end{aligned}
$$

sendo D_k a expressão entre colchetes. Escolhendo como função de Liapunov candidata discreta à norma quadrado deste resíduo:

$$
V_k = \mathbf{r}_k^T \mathbf{r}_k \;\Rightarrow\; \Delta V_k = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} - \mathbf{r}_k^T \mathbf{r}_k \simeq -2\alpha_k \mathbf{r}_k^T D_k^T \mathbf{f}(\mathbf{x}_k) + \alpha_k^2 \mathbf{f}^T(\mathbf{x}_k) D_k D_k^T \mathbf{f}(\mathbf{x}_k)
$$

e derivando e igualando a zero segundo a metodologia LOC:

$$
\alpha_k = \frac{\mathbf{r}_k D_k^T \mathbf{f}(\mathbf{x}_k)}{\|D_k^T \mathbf{f}(\mathbf{x}_k)\|^2} \tag{5.103}
$$

e

$$
\Delta V_k \simeq -\frac{\mathbf{r}_k^T D_k^T \mathbf{f}(\mathbf{x}_k)}{\|D_k^T \mathbf{f}(\mathbf{x}_k)\|^2} \leq 0
$$

Para o cálculo do comprimento de passo (5.103) é necessário o conhecimento dos hessianos das restrições (as expressões $\nabla^2 E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))$). Diversos experimentos mostraram que, na prática, o algoritmo não se vê comprometido se retirarmos os dois últimos termos de D_k, aqueles que contêm justamente os hessianos das restrições.

Além disso, a cada iteração na qual a trajetória sai do conjunto viável, isto é, para todo k tal que $\mathbf{x}_k \in \chi$ e $\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k) \notin \chi$, é agregado a $\mathbf{x}_{k+1}$ o termo $\frac{E(\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k))\mathbf{f}(\mathbf{x}_k)}{\|\nabla E(\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k))\| \|\mathbf{f}(\mathbf{x}_k)\|}$. Esse termo representa um vetor na direção $\mathbf{f}(\mathbf{x}_k)$ de norma igual a um percentual da distância entre o conjunto viável e o ponto $\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k)$. Assim:

$$
\forall \mathbf{x}_k \in \chi \text{ e } \mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k) \notin \chi : \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k) + \frac{E(\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k))\mathbf{f}(\mathbf{x}_k)}{\|\nabla E(\mathbf{x}_k - \alpha_k \mathbf{f}(\mathbf{x}_k))\| \|\mathbf{f}(\mathbf{x}_k)\|}
$$

O intuito disto é retirar o máximo possível do vetor $\mathbf{x}_{k+1} - \mathbf{x}_k$ fora do conjunto viável, conseguindo assim que as trajetórias se afastem o menos possível deste. Se apenas uma restrição for violada no ponto $\mathbf{x}_{k+1} \notin \chi$, e esta for afim ou esférica, então o ponto de retorno ao conjunto viável será a projeção ortogonal sobre este, como ilustrado na figura 5,16.

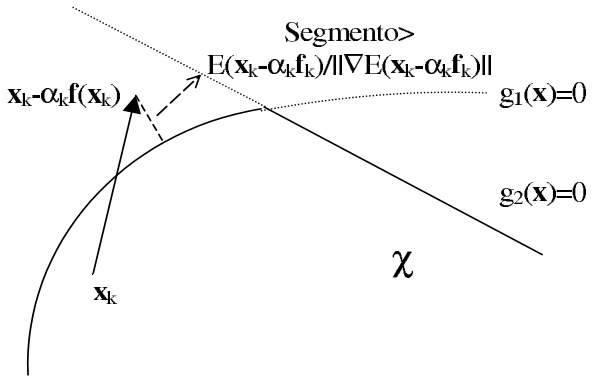

Figura 5.16: Representação da projeção ortogonal

5.12.2 Algoritmo 3

Neste terceiro algoritmo, também mantêm-se o comprimento do passo (5.78) e a lei de atualização das variáveis (5.79) para a fase de alcançar o conjunto viável. Para a fase de convergência, escolhe-se como resíduo o segundo termo da expressão do meio da desigualdade (5.80).

$$r_k := E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))$$
$$\Rightarrow\ r_{k+1} = E(\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_{k+1})) \simeq E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) + \nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\Delta\bar{\mathbf{x}}$$

onde utilizamos a nomenclatura $\Delta\bar{\mathbf{x}} = \mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_{k+1}) - (\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))$.

A aproximação é válida supondo que $\mathbf{x}_k \in \chi$ e $\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k) \notin \chi$.

Observe-se que, por ser $\mathbf{f}(\mathbf{x})$ contínuo, $\mathbf{f}(\mathbf{x}_{k+1}) = \mathbf{f}(\mathbf{x}_k - \alpha_k\mathbf{f}(\mathbf{x}_k)) \simeq \mathbf{f}(\mathbf{x}_k) - \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)$.

Sendo portanto $\Delta\bar{\mathbf{x}} = -\alpha_k\mathbf{f}(\mathbf{x}_k) + \alpha_k D_{\mathbf{f}}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) = -\left[I - D_{\mathbf{f}}^T(\mathbf{x}_k)\right]\alpha_k\mathbf{f}(\mathbf{x}_k)$

$$\Rightarrow\ r_{k+1} \simeq E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) - \nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\left[I - D_{\mathbf{f}}^T(\mathbf{x}_k)\right]\alpha_k\mathbf{f}(\mathbf{x}_k)$$

Escolhendo como função de Liapunov candidata discreta $V_k = r_k^2$:

$$\begin{aligned}\Delta V_k = r_{k+1}^2 - r_k^2 \simeq\ & -2\alpha_k E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\left[I - D_{\mathbf{f}}^T(\mathbf{x}_k)\right]\mathbf{f}(\mathbf{x}_k) + \\ & \alpha_k^2\left(\nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))\left[I - D_{\mathbf{f}}^T(\mathbf{x}_k)\right]\mathbf{f}(\mathbf{x}_k)\right)^2\end{aligned}$$

e por LOC, derivando com respeito a α_k e igualando a zero

$$\alpha_k = \frac{E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k))}{\nabla^T E(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) \left[I - D_{\mathbf{f}}^T(\mathbf{x}_k)\right] \mathbf{f}(\mathbf{x}_k)} \tag{5.104}$$

$$\Rightarrow \ \Delta V_k \simeq -E^2(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_k)) < 0$$

Além destes 3 algoritmos, outros 17 comprimentos de passos foram testados experimentalmente. Porém, os resultados não foram apresentados por não mostrar diferenças apreciáveis com respeito aos aqui desenvolvidos.

5.12.3 Execução dos algoritmos discretos

A seguir são apresentados os gráficos dos algoritmos 2 e 3 aplicados aos problemas 1 e 2, assim como os gráficos das versões projetadas.

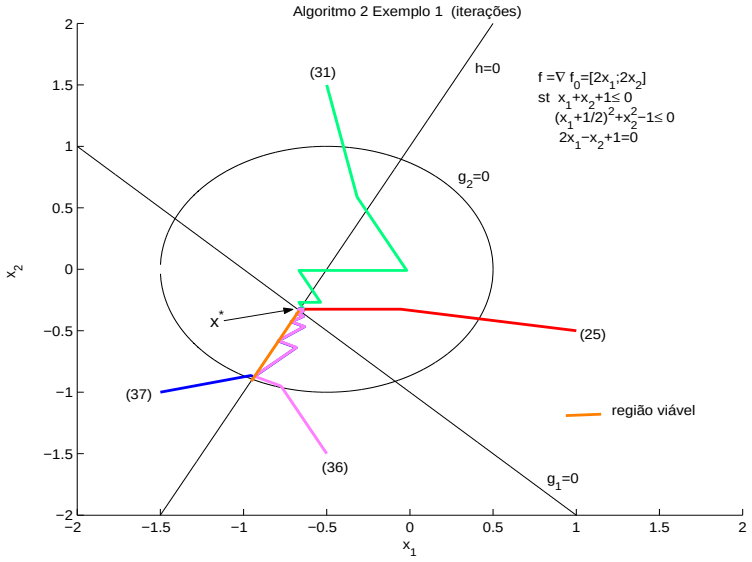

Figura 5.17: Algoritmo 2 aplicado ao exemplo 1

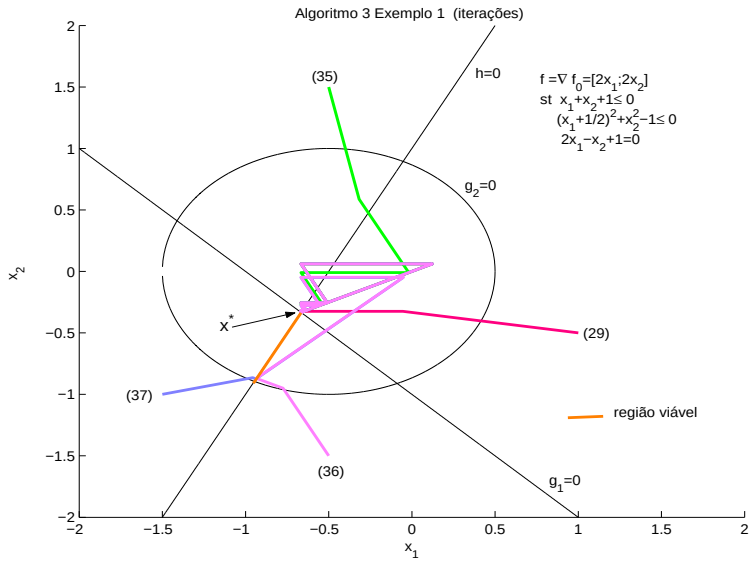

Figura 5.18: Algoritmo 3 aplicado ao exemplo 1

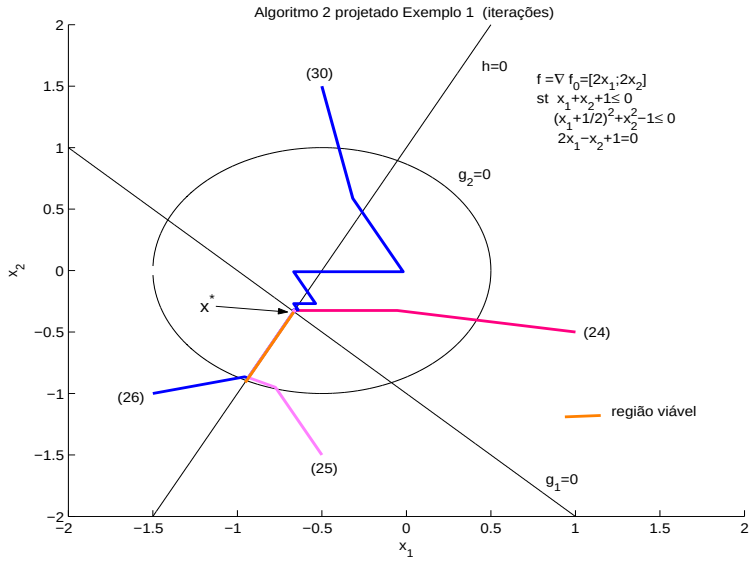

Figura 5.19: Algoritmo 2 projetado aplicado ao exemplo 1

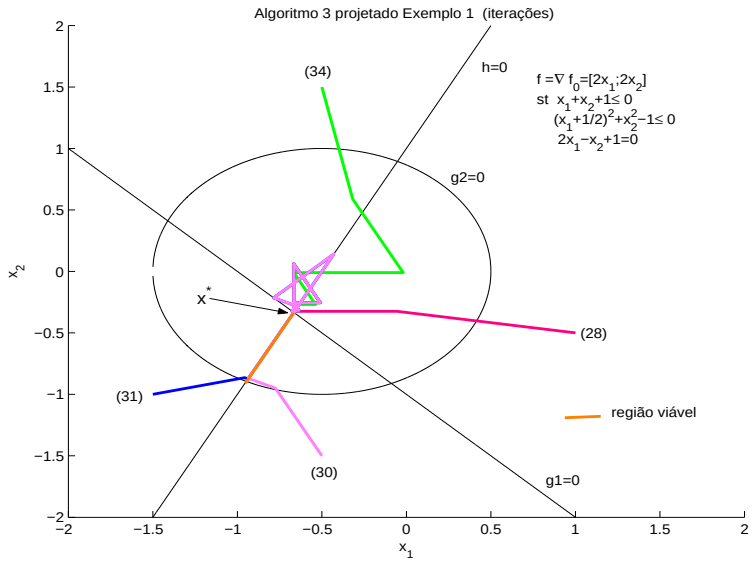

Figura 5.20: Algoritmo 3 projetado aplicado ao exemplo 1

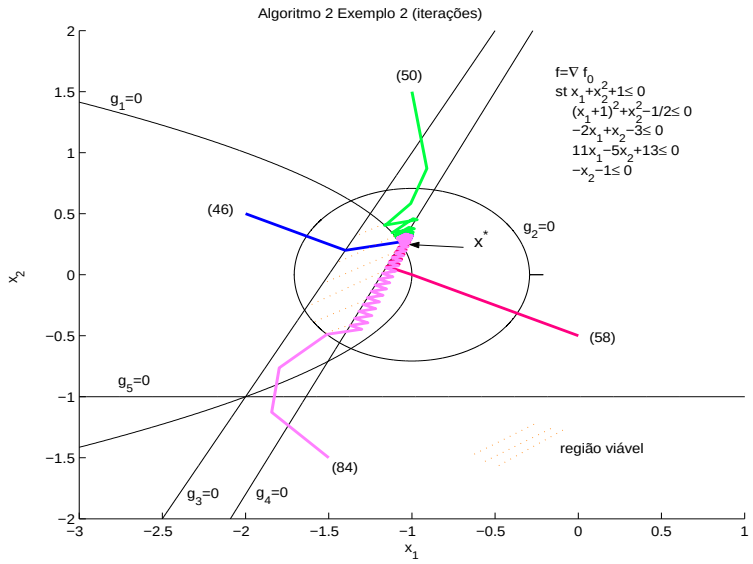

Figura 5.21: Algoritmo 2 aplicado ao exemplo 2

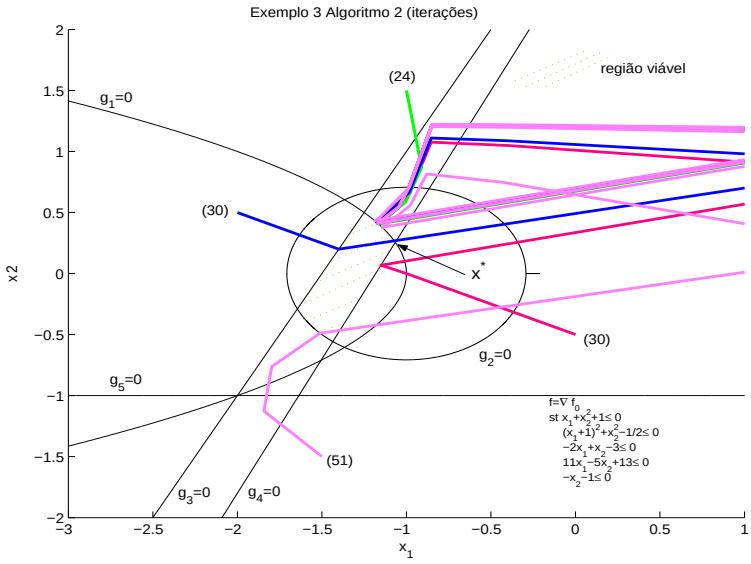

Figura 5.22: Algoritmo 3 aplicado ao exemplo 2

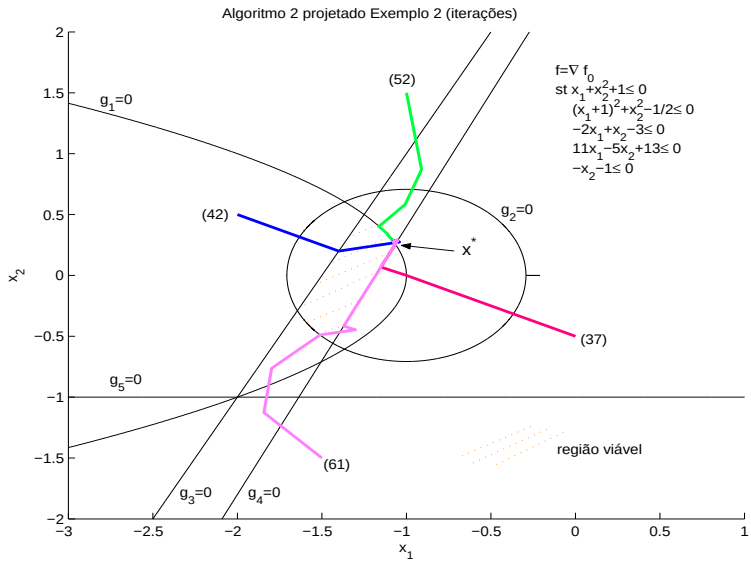

Figura 5.23: Algoritmo 2 projetado aplicado ao exemplo 2

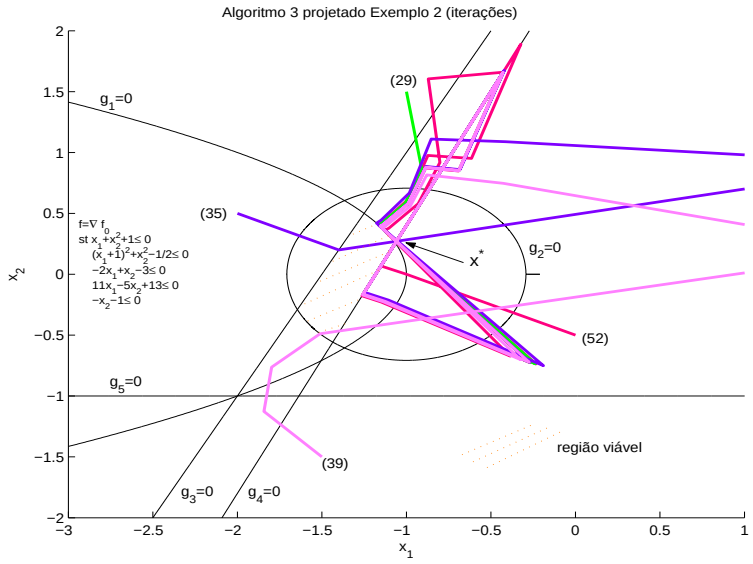

Figura 5.24: Algoritmo 3 projetado aplicado ao exemplo 2

Observando estes gráficos, assim como os do primeiro algoritmo discreto apresentado, pode-se concluir:

1) O algoritmo 1 é o que apresenta melhor comportamento, convergindo sempre ao ponto ótimo em menor número de iterações que os outros algoritmos. A versão projetada não apresenta o chattering característico dos sistemas a estrutura variável, o que faz esta opção atrativa. Em compensação, toda vez que a trajetória sai do conjunto viável, cada iteração exige um maior número de avaliações das restrições.

2) O algoritmo 2 apresenta um maior número de iterações que o primeiro. Além disso, no caso da inexistência de restrições de igualdade (exemplo 2), o ponto de convergência exibe um erro maior que aquele apresentado pelo algoritmo 1. Este erro pode ser reduzido diminuindo a constante ϵ, mas ao custo de aumentar o número de iterações. A versão projetada também se mostra bem comportada.

3) O algoritmo 3 é o que apresenta pior comportamento, convergindo a um ciclo limite no caso da ausência de restrições de igualdade (exemplo 2), e portanto não sendo capaz de achar o ponto ótimo. O mesmo problema se observa na versão projetada.

Os algoritmos foram também testados com os problemas 3 e 4, a partir dos mesmos pontos iniciais que os utilizados no capítulo de otimização convexa. Os resultados de todas as experiências (número de iterações a partir de cada ponto inicial testado) são apresentados resumidamente na seguinte tabela.

	$\mathbf{x}_0$	alg. 1	alg. 2	alg. 3	alg. 1p.	alg. 2p.	alg. 3p.
Problema 1	$[1; -0.5]$	6	25	29	13	24	28
	$[-0.5; 1.5]$	10	31	35	19	30	34
	$[-1.5; -1]$	16	37	37	12	26	31
	$[-0.5; -1.5]$	15	36	36	11	25	30
Problema 2	$[0; -0.5]$	26	58	30	14	37	52
	$[-1; 1.5]$	66	50	24	60	52	29
	$[-2; 0.5]$	35	46	30	26	42	35
	$[-1.5; -1.5]$	47	84	51	29	61	39
Problema 3	$\mathbf{x}_{0_1}$	-	-	-	-	-	403
	$\mathbf{x}_{0_2}$	20	-	12	10	-	-
Problema 4	$x_{0_i} = 0.01$ $i \in \{1, .., 14\}$	336	-	-	336	1429	-
	$x_{0_i} = 0.02$ $i \in \{1, .., 14\}$	15	28	-	15	397	-

Tabela 5.1: Resultados das implementações dos diferentes algoritmos, onde - indica que o algoritmo não convergiu em até 1000 iterações

$\mathbf{x}_{0_1} = [20\,55\,15\,20\,60\,20\,20\,60\,20\,20\,60\,20\,20\,60\,20]^T$

$\mathbf{x}_{0_2} = [10\,50\,2\,1\,55\,0\,0\,60\,5\,5\,70\,12\,7\,70\,16]^T$

5.13 Conclusões deste capítulo

Seguindo a metodologia de funções de Liapunov de controle e de controle ótimo de Liapunov é derivado o algoritmo contínuo (5.71) e sua versão discreta (5.76) capazes de achar o ponto ótimo em um problema de desigualdades variacionais, onde o conjunto viável está determinado por restrições de desigualdade convexas e de igualdade afins. As exigências sobre a função objetivo são a condição (5.14), para o caso contínuo, mais a condição (5.90) para o caso discreto. Ambas restrições são mais fracas que as exigidas

nas bibliografias referenciadas, sendo os nossos algoritmos adequados para a resolução de uma maior quantidade de problemas.

Os algoritmos são inteiramente similares àqueles desenvolvidos no capítulo 4 e apresentados em [61], os quais eram aplicados na resolução de problemas de otimização convexa, sendo portanto o caso aqui abordado de aplicação mais geral.

A maior desvantagem apresentada, no caso discreto, consiste na possível redução do comprimento do passo (5.81) toda vez que a trajetória sai do conjunto viável, pois isso exige uma avaliação das restrições no ponto de destino a fim de conferir que não seja violada mais de uma restrição de desigualdade (ou nenhuma, caso sejam violadas restrições de igualdade). Quanto maior for o número de restrições que conformam o conjunto viável, maior será o número de avaliações destas, o que aumenta a demora de cada iteração e encarece, do ponto de vista computacional, a execução do algoritmo. Entretanto, apesar deste inconveniente, esta metodologia constitui uma maneira simples de garantir o retorno ao conjunto viável na projeção ortogonal (ou em um ponto próximo) sem necessidade de calculá-la ou de aplicar um algoritmo de minimização convexa para determinar esta projeção a cada iteração, o que seria extremadamente caro do ponto de vista computacional. A observância das condições (5.14) e (5.90) por parte da função objetivo garante a convergência das trajetórias geradas pelo algoritmo ao ponto ótimo.

Assim sendo, para um número reduzido de restrições, o algoritmo discreto (5.76), e sua versão projetada (5.98), mostram-se capazes de alcançar o ponto ótimo em um número reduzido de iterações e em um tempo razoavelmente pequeno.

Também aqui, como no caso do algoritmo para resolver problemas de otimização convexa, mostrou-se a necessidade de realizar uma avaliação quantitativa dos erros produzidos pelos algoritmos, isto é, quantificar o erro na determinação do ponto ótimo achado e o nível de satisfação das restrições neste ponto.

Capítulo 6

Conclusões, contribuições e propostas de trabalho futuro

No presente trabalho foram apresentados algoritmos, tanto contínuos quanto discretos, capazes de resolver diversos problemas de otimização. As leis de atualização das variáveis, em todos os casos, foram derivadas utilizando a metodologia de funções de Liapunov de controle e controle ótimo de Liapunov. Os problemas abordados foram o de achar zeros de funções vetoriais não lineares, de minimização de funções escalares, o problema geral de otimização convexa e desigualdades variacionais. Muitos problemas comuns em diversas áreas da ciência e da engenharia podem ser formulados como algum destes problemas de otimização.

O trabalho de Bhaya e Kaszkurewicz (ver, principalmente [13]), consiste na interpretação dos algoritmos para resolver problemas de otimização como sistemas dinâmicos de controle em malha fechada. Uma primeira vantagem desta abordagem consiste em que a teoria de controle oferece poderosas ferramentas de análise dos sistemas dinâmicos, tanto lineares como não lineares, tanto contínuos como discretos, como por exemplo avaliação da convergência das trajetórias, taxas de convergência, regiões de convergência, estabilidade, entre outras. Além desta vantagem, a teoria de controle oferece uma metodologia sistemática de desenvolvimento de algoritmos através das funções de Liapunov de controle e do controle ótimo de Liapunov. Assim, alguns algoritmos amplamente conhecidos e tratados na bibliografia podem ser interpretados como sistemas dinâmicos de controle em malha fechada, o que constitui uma nova abordagem e

possibilita análises alternativas sob esta nova perspectiva, assim como também outros novos algoritmos podem ser desenvolvidos através desta metodologia.

Bhaya e Kaszkurewicz focalizaram principalmente o problema de achar zeros de funções vetoriais não lineares. O presente trabalho teve como intuito a utilização desta metodologia para o desenvolvimento de algoritmos capazes de resolver os outros problemas de otimização mencionados, assim como também foram analisados os algoritmos apresentados em [13].

Cabe destacar que, sob o foco da teoria de controle, muitos dos algoritmos apresentados constituem os chamados sistemas dinâmicos de controle a estrutura variável, amplamente conhecidos na literatura de controle, a qual oferece uma abundante quantidade de ferramentas de análise. Os sistemas dinâmicos a estrutura variável geram trajetórias em modos deslizantes até alcançar o ponto ótimo. Algoritmos a estrutura variável possuem características que os tornam atrativos, por exemplo a previsibilidade do percurso da trajetória sobre a superfície de deslizamento, ou a possibilidade de apresentarem convergência em tempo finito.

A conclusão geral do trabalho é que os algoritmos apresentados se mostraram capazes de resolver os problemas de otimização testados, em alguns casos de maneira mais eficiente que outros algoritmos apresentados na bibliografia, e mostrando diversas vantagens com respeito a estes. Demonstra-se assim, que a teoria de controle oferece um caminho promissor de futuras pesquisas destinadas a achar algoritmos alternativos, assim como de análise dos já existentes, para a resolução tanto destes problemas de otimização como de outros ainda não abordados sob esta perspectiva.

Entretanto, cabe mencionar algumas conclusões pontuais referentes a cada problema abordado em particular:

1) Os algoritmos de primeira ordem para achar zeros de funções mostraram que, embora o conhecido algoritmo de Newton oferece um melhor comportamento (menor número de iterações no caso discreto, por exemplo), este se vê afetado pela presença de singularidades estranhas, o que pode comprometer seu desempenho, a diferença de outros algoritmos apresentados que não se vêem afetados por estas singularidades. Também, o fato do algoritmo de Newton inverter a matriz jacobiana pode comprometer seu rendimento quando aplicado a problemas de grande porte, isto é, com um grande

número de variáveis, o que não foi testado no presente trabalho[1]. Alguns dos algoritmos apresentados se mostraram mais eficientes do que o algoritmo de Newton para achar um zero de alguma das funções testadas, razão pela qual a utilização destes merece ser considerada.

A utilização do "time" de algoritmos também oferece uma alternativa interessante quando se pretende aumentar a possibilidade de achar um zero de uma função vetorial, mesmo que isto redunde em uma demora maior na convergência da trajetória.

Quando a partir de um determinado ponto inicial interessa aumentar a previsibilidade do percurso da trajetória gerada pelo algoritmo, o estudo das bacias de atração mostrou-se como uma ferramenta adequada para este fim. Observou-se que o algoritmo de Newton, para as funções testadas, apresenta bacias de contornos mais irregulares e superfícies menores que outros, sendo portanto, o percurso das suas trajetórias menos previsível.

O estudo das singularidades estranhas mostrou também que a alternativa proposta por Branin ([18]) nem sempre é capaz de achar sucessivamente todos os zeros de uma função vetorial, e que inclusive uma singularidade estranha pode ser ponto de atração das trajetórias. Se encontraram contraexemplos para as duas conjecturas apresentadas no trabalho do Branin.

2) Os algoritmos de primeira ordem desenvolvidos para achar mínimos de funções escalares convexas se mostraram eficientes com os problemas testados. O estudo destes algoritmos, utilizando as ferramentas fornecidas pela teoria de controle, mostrou as limitações de alguns dos algoritmos propostos na bibliografia. O algoritmo a estrutura variável apresentado mostrou-se eficiente na localização do mínimo global das funções testadas.

3) Os algoritmos de segunda ordem, tanto para achar zeros de funções vetoriais não lineares como aqueles destinados a achar mínimos de funções escalares não convexas, também se mostraram eficientes na busca dos objetivos. Entretanto, estes mostraram uma necessidade de sintonização dos ganhos mais cuidadosa que no caso dos algoritmos de primeira ordem, assim como de uma escolha adequada do valor inicial da variável de controle. As trajetórias geradas por estes também mostraram percursos maiores

[1]Os métodos quase-Newton, já mencionados, que não invertem a matriz jacobiana, não foram testados no presente trabalho.

pelo plano de fase. Apesar destes inconvenientes, no caso de achar zeros de funções vetoriais, em alguns casos se mostraram mais eficientes que os algoritmos de primeira ordem, notadamente com a função de Branin para a constante $c = 1$.

No caso dos algoritmos para minimização de funções escalares não convexas, observou-se que a capacidade de ultrapassar mínimos locais para acharem o mínimo global da função só é conseguida com uma cuidadosa sintonização dos ganhos também, e apenas em alguns casos. O algoritmo DIN proposto na bibliografia ([3]), também mostrou igual inconveniente e parece atingir o objetivo apenas quando se aproxima do algoritmo HBF.

No caso dos algoritmos discretos, como já foi mencionado, as trajetórias geradas por estes podem se estacionar em selas, o que faz fracassar o objetivo pretendido. Porém, é possível que a escolha de outros resíduos (isto é, reformulando a malha de controle do sistema discreto), elimine este inconveniente.

4) O algoritmo apresentado para resolver o problema geral de otimização convexa se mostrou eficiente e simples de implementar apesar da dificuldade do problema. As vantagens mostradas contrastam visivelmente com a maioria dos algoritmos apresentados na bibliografia, que condicionam as restrições que determinam o conjunto viável ou a função objetivo (limitando a aplicabilidade do algoritmo), aumentam a dimensão do problema, ou utilizam operadores como o de projeção ortogonal que, na maioria dos casos, é computacionalmente caro de calcular e portanto compromete a eficiência do algoritmo. O critério de parada adotado para o caso discreto também é mais simples do que a maioria dos apresentados na bibliografia, embora empregue algumas iterações a mais.

A alternativa do algoritmo projetado sobre a fronteira do conjunto viável também mostrou-se atrativa devido à previsibilidade da trajetória e a eliminaçào do chattering característico das trajetórias em modos deslizantes.

5) O algoritmo proposto para a resolução de problemas de desigualdades variacionais também mostrou-se eficiente com os problemas testados. Além de apresentar as mesmas vantagens que as mencionadas para o algoritmo para a resolução dos problemas de otimização convexa, o requerimento exigido para a função objetivo é mais fraco do que aqueles requeridos na maioria dos algoritmos apresentados na bibliografia, permitindo assim a aplicação com uma maior variedade de problemas.

No caso do algoritmo discreto, as modificações realizadas ao comprimento do passo calculado por LOC para garantir convergência, podem ser caras do ponto de vista computacional, notadamente diante da presença de um grande número de restrições. Porém, o fato de não utilizar o operador de projeção ortogonal, que também pode ser caro de implementar, é uma vantagem evidente com respeito aos seus concorrentes.

Dentre as principais contribuições desta tese, cabe destacar:

- O estudo mais detalhado dos algoritmos de primeira ordem apresentados em ([13]) para achar zeros de funções vetoriais, incluindo bacias de atração de cada zero para diversas funções de teste, e admissão de ganhos nestes algoritmos. Foram estudadas as singularidades estranhas para o algoritmo de Newton, e as conseqüências da presença de singularidades estranhas tanto nos algoritmos contínuos quanto nos discretos. Estudo e apresentação de métodos de linearização do algoritmo de Newton, e estudo do algoritmo de Branin. Foram realizados testes com funções de mais de duas variáveis. Foi apresentada a utilização do "time" de algoritmos.

- Foram desenvolvidos e testados algoritmos contínuos de primeira ordem para minimização de funções escalares convexas. Os algoritmos foram discretizados com comprimento de passo otimizado. Alguns dos algoritmos destinados a este fim, já conhecidos na bibliografia, foram interpretados como sistemas dinâmicos de controle e estudados utilizando a teoria de controle.

- Foram desenvolvidos e testados algoritmos contínuos de segunda ordem para achar zeros de funções vetoriais. Os algoritmos foram discretizados com comprimento de passo otimizado.

- Foram desenvolvidos e testados algoritmos contínuos de segunda ordem para minimização de funções escalares não necessariamente convexas. Os algoritmos foram discretizados com comprimento de passo otimizado. Os algoritmos já exisentes na bibliografia destinados a este fim, foram interpretados como sistemas de controle e estudados utilizando as ferramentas da teoria de controle.

- Foi desenvolvido e testado um algoritmo contínuo para resolução do problema

geral de otimização convexa que não apresenta muitos dos inconvenientes de outros propostos na bibliografia. Este foi discretizado com comprimento de passo otimizado, e foi derivada uma versão projetada deste. O desempenho dos algoritmos discretos foi comparado com o de outros algoritmos.

- Foram estendidos os lemas apresentados por Liao ([53, lemas 2.2 e 2.3]) ao caso geral de otimização convexa.

- Foi desenvolvido e testado um algoritmo contínuo para a resolução do problema de desigualdades variacionais com exigências sobre a função objetivo e as restrições mais fracas que aquelas apresentadas por outros algoritmos presentes na bibliografia. O algoritmo foi discretizado com comprimento de passo otimizado, e foi derivada uma versão projetada deste. As condições de convergência do algoritmo discreto foram analisadas.

- Discretização do algoritmo apresentado por Gao *et.al.* ([30]) com comprimento de passo otimizado por LOC.

O trabalho realizado nesta tese possui um caráter claramente qualitativo, onde o objetivo principal foi a utilização da teoria de controle para o projeto e análise de algoritmos destinados a resolver diversos problemas de otimização. Porém, evidentemente não foram esgotadas todas as possibilidades de estudo dos algoritmos apresentados, e análises destes em diversas aplicações específicas, assim também como testes para diversos problemas particulares, ou de maior porte, se fazem imprescindíveis.

Existe, portanto, a necessidade de prosseguimento do trabalho. Enumeraremos, resumidamente, algumas propostas de trabalho futuro.

1) Como já foi mencionado, testar a eficiência dos algoritmos em problemas de maior porte, isto é, com maior número de variáveis, e diante da presença de restrições, com um maior número destas, com o intuito de apreciar mais claramente as vantagens e desvantagens destes com respeito a outros apresentados na bibliografia, assim como delimitar melhor os campos de aplicação específicos para cada algoritmo. Em problemas de grande porte, também, o tempo de convergência poderia ser testado, oferecendo assim um novo elemento de comparação. Nos problemas pequenos aqui apresentados

de maneira ilustrativa, estas demoras foram inconclusivas em quase todos os casos.

2) Realização de uma análise quantitativa de algumas das características dos algoritmos, o que permitiria a delimitação mais precisa do campo de aplicação de cada um destes. Por exemplo, uma análise mais exaustiva de como o algoritmo de Newton se vê afetado diante da presença de singularidades estranhas. Outro exemplo poderia ser a utilização das ferramentas oferecidas pela teoria de sistemas dinâmicos para análise das bacias de atração de cada zero, podendo assim prever as formas e tamanhos destas e melhorando portanto a previsibilidade de percurso das trajetórias a partir de um determinado ponto inicial. Foi notado que os algoritmos de segunda ordem apresentam maior sensibilidade ao valor dos ganhos que os algoritmos de primeira ordem, assim como a necessidade de uma escolha adequada do valor inicial da variável de controle. A robustes dos algoritmos com respeito a variações destes valores precisa ser testada. Uma análise das propriedades numéricas dos algoritmos apresentados forneceria, também, um elemento de comparação e delimitação da aplicabilidade destes. Analisar a relação com outros algoritmos tradicionais de otimização convexa, por exemplo, aqueles baseados em multiplicadores de Lagrange, ou no enfoque primal-dual. Finalmente, uma avaliação quantitativa dos erros dos pontos ótimos achados em todos os casos, assim como também, diante da presença de restrições, a quantificação da satisfação destas no ponto ótimo, se faz imprescindível para contribuir com a avaliação da eficiência dos algoritmos propostos.

3) Estudar a aplicação dos algoritmos em problemas com exigências diferentes. Por exemplo, permitir que as funções objetivos (em todos os casos) sejam descontínuas ou não diferenciáveis, o que exigiria que a lei de atualização das variáveis seja um mapeamento de conjunto (*set valued maps*), e a análise de funções com lado direito descontínuo através das ferramentas desenvolvidas por Filippov ([28]). No caso do problemas de otimização convexa, permitir que a função objetivo seja não convexa, e implementar outro algoritmo, talvez de segunda ordem, na fase de convergência com o intuito de ultrapassar mínimos locais dentro do conjunto viável para achar o mínimo global restrito.

4) Utilizar a teoria de controle e a metodologia CLF/LOC para a dedução de algoritmos capazes de resolver outros problemas de otimização. Por exemplo, no contexto de achar zeros de funções, dentre os muitos algoritmos contínuos existentes, a classe

de algoritmos de continuação ou homotopia [1, 2] também utiliza uma EDO do tipo (1.2) como ponto de partida dos métodos chamados preditor-corretor, entretanto, não há a utilização dos conceitos de controle no projeto e na análise destes algoritmos. Outro exemplo poderia ser a utilização da teoria de controle para o desenvolvimento de algoritmos para a resolução de problemas duais, no contexto de otmização convexa.

Referências Bibliográficas

[1] E. Allgower and K. Georg. Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *SIAM Review*, 22:28–85, 1980.

[2] E. Allgower and K. Georg. *Introduction to numerical continuation methods*, volume 45 of *Classics in applied mathematics*. SIAM, 2003.

[3] F. Alvarez, H. Attouch, J. Bolte, and P. Redont. A second-order gradient-like dissipative dynamical system with Hessian-driven damping. Application to optimization and mechanics. *J. Math. Pures Appl.*, 81:747–779, 2002.

[4] K. J. Åström and B. Wittenmark. *Computer–Controlled Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1984.

[5] H. Attouch and R. Cominetti. A dynamical approach to convex minimization coupling approximation with the steepest descent method. *Journal of differential equations*, 128:519–540, 1996.

[6] H. Attouch and M. Teboulle. Regularized Lotka-Volterra dynamical system as continuous proximal-like method in optimization. *Journal of optimization theory and applications*, 121(3):541–570, 2004.

[7] A. Auslender and M. Teboulle. Interior projection-like methods for monotone variational inequalities. *Mathematical Programming*, 104:39–68, 2005.

[8] A. Bacciotti and F. Ceragioli. Stability and stabilization of discontinuous systems and nonsmooth Liapunov functions. *ESAIM: Control, Optimisation and Calculus of Variations*, 4:361–376, 1999. ESAIM = European Series in Applied and Industrial Mathematics. Available at http://calvino.polito.it/˜bacciott.

[9] M. Barbarosou and N. G. Maratos. Non-feasible gradient projection recurrent neural network for equality constrained optimization. In *Proc. of the IEEE international joint conference on neural networks*, volume 3, pages 2251–2256, 2004.

[10] A. Bhaya and E. Kaszkurewicz. Iterative methods as dynamical systems with feedback control. In *Proc. 42nd IEEE Conference on Decision and Control*, pages 2374–2380, Maui, Hawaii, USA, December 2003.

[11] A. Bhaya and E. Kaszkurewicz. Newton algorithms via control Liapunov functions for polynomial zero finding. In *Proc. 43rd IEEE Conference on Decision and Control*, pages 1629–1634, Bahamas, December 2004.

[12] A. Bhaya and E. Kaszkurewicz. Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method. *Neural Networks*, 17(1):65–71, 2004.

[13] A. Bhaya and E. Kaszkurewicz. *Control perspectives on numerical algorithms and matrix problems*. Advances in Control. SIAM, Philadelphia, 2006.

[14] A. Bhaya and E. Kaszkurewicz. A control-theoretic approach to the design of zero finding numerical methods. *IEEE Transactions on Automatic Control*, 52(6):1014–1026, June 2007.

[15] S. I. Birbil and S. C. Fang. An electromagnetism-like mechanism for global optimization. *Journal of global optimization*, 25:263–282, 2003.

[16] E. Bobrovnikova. Nonlinear optimization models. Available at http://www.sor.princeton.edu/~rvdb/ampl/nlmodels/index.html, 1996.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, U.K., 2004.

[18] F. H. Branin. Widely convergent method to finding multiple solutions of simultaneous nonlinear equations. *IBM Journal Res. Develop.*, 16(5):504–522, 1972.

[19] A. Cabot. Inertial gradient-like dynamical system controlled by a stabilizing term. *Journal of Optimization Theory and Applications*, 120(2):275–303, 2004.

[20] A. Cabot. The steepest descent dynamical system with control. application to constrained minimization. *Control, Optimisation and Calculus of Variations*, 10:243–258, 2004.

[21] K. S. Chao and R. J. P. de Figueiredo. Optimally controlled iterative schemes for obtaining the solution of a non-linear equation. *Internat. J. Control*, 18(2):377–384, 1973.

[22] J-P. Chehab and J. Laminie. Differential equations and solution of linear systems. *Numerical Algorithms*, 40:103–124, 2005.

[23] J. Cortés. Discontinuous dynamical systems. a tutorial on notions of solutions, nonsmooth analysis, and stability. *IEEE Control systems magazine*, January 2007.

[24] J. P. Dedieu. Newton's method and some complexity aspects of the zero-finding problem. *Fundation of computational mathematics*, pages 45–67, 2001.

[25] L. C. W. Dixon, J. Gomulka, and G. P. Szegö. Towards a global optimization technique. In L. C. W. Dixon and G. P. Szegö, editors, *Towards a Global Optimization*, pages 29–54. North Holland Publishing Company, Amsterdam, 1975.

[26] F. Facchinei and J. S. Pang. *Finite-dimensional variational inequalities and complementary problems*, volume 1. Springer-Verlag, New York, 2003.

[27] L. V. Ferreira. *Utilização de sistemas gradientes para resolução de problemas de otimização em computadores paralelos*. PhD thesis, COPPE/UFRJ, Programa de Engenharia Elétrica, Agosto 2006. In Portuguese.

[28] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides*. Kluwer Academic, Dordrecht, 1988.

[29] A. L. Fradkov and A. Yu. Pogromsky. *Introduction to Control of Oscillations and Chaos*, volume A 35 of *World Scientific Series on Nonlinear Science*. World Scientific, Singapore, 1998.

[30] X. B. Gao, L. Z. Liao, and L. Qi. A novel neural network for variational inequalities with linear and nonlinear constraints. *IEEE Transactions on Neural Networks*, 16(6):1305–137, 2005.

[31] M. Gasparo, S. Pieraccini, and A. Armellini. An infeasible interior-point method with nonmonotonic complemenary gaps. *Optimization Methods and Software*, 17(4):561–586, 2002.

[32] M. P. Glazos, S. Hui, and S. H. Żak. Sliding modes in solving convex programming problems. *SIAM J. Control Optim.*, 36(2):680–697, March 1998.

[33] B. S. Goh. Algorithms for unconstrained optimization via control theory. *J. Optimization Theory Appl.*, 92(3):581–604, March 1997.

[34] J. Gomulka. Remarks on Branin's method for solving nonlinear equations. In L. C. W. Dixon and G. P. Szegö, editors, *Towards a Global Optimization*, pages 96–106. North Holland Publishing Company, Amsterdam, 1975.

[35] A. Groenwold and J. Snyman. Global optimization using dynamic search trajectories. *Journal of Global Optimization*, 24:51–60, 2002.

[36] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, Berlin, 2nd, revised edition, 1993.

[37] D. Han. A modified alternating direction method for variational inequality problems. *Applied mathematics and optimization*, 45:63–74, 2002.

[38] D. R. Han and W. Y. Sun. A new modified Goldstein-Levitin-Polyak projection method for variational inequalities problems. *Computers and Mathematics with applications*, 47:1817–1825, 2004.

[39] J. W. Hardy. An implemented extension of Branin's method. In L. C. W. Dixon and G. P. Szegö, editors, *Towards a Global Optimization*, pages 117–142. North Holland Publishing Company, Amsterdam, 1975.

[40] R. Hauser and J. Nedić. On the relationship between the convergence rates of discrete and continuous iterative processes. Numerical Analysis Group Research Report NA-04/10, Oxford University Computing Laboratory, Oxford, 2004. To appear in *SIAM J. Optim.*

[41] R. Hauser and J. Nedić. The continuous Newton–Raphson method can look ahead. 15(3):915–925, 2005.

[42] A. R. Hedar and H. Fukushima. Minimizing multimodal functions by simplex coding genetic algorithm. Available at http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar-files/go-files/SCGA.pdf, 2003.

[43] S. Hedge and S. Kacera. Safeguarded zero-finding methods. Computer science department research report, Courant institute of mathematical sciences. New York University, New York.

[44] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems and Linear Algebra*. Academic Press, New York, 1974.

[45] L. Hsu and R. R. Costa. B-mrac: Global exponential stability with a new model reference adaptive controller based on binary control theory. *Control theory and advanced technology*, 10(4):649–668, 1994.

[46] X. L. Hu and J. Wang. Solving pseudomonotone variational inequalities and pseudoconvex optimizaion problems using the projection neural network. *IEEE Transactions on Neural Networks*, 17(6):1487–1499, 2006.

[47] W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14:331–355, 1999.

[48] A. N. Iusem. On the convergence properties of the projected gradient method for convex optimization. *Computational and applied mathematics*, 22(1):37–52, 2003.

[49] K. Kashima, S. Ashida, and Y. Yamamoto. System theory for numerical analysis. In *Preprints of the 16th IFAC World Congress*, pages 1–6, Prague, Czech Republic, July 2005. IFAC. Paper 01980.pdf on Preprints CD-ROM.

[50] D. Kinderlehrer and G. Stampacchia. *An introduction to variational inequalities and their applications*. Academic Press, New York, 1980.

[51] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Evolutionary operators in global optimization with dynamic search trajectories. *Numerical Algorithms*, 34:393–403, 2003.

[52] D. Leake and H. Mukai. Acceleration of one-step stationary root-finding algorithms. *Journal of Numerical Analysis*, 9:81–93, 1989.

[53] L. Z. Liao. A continuous method for convex programming problems. *Journal of optimization theory and applications*, 124(1):207–226, 2005.

[54] L. Z. Liao, H. D. Qi, and L. Q. Qi. Neurodynamical optimization. *J. Global Optimization*, 28(2):175–195, February 2004.

[55] M. Locatelli and G. R. Wood. Objective function features providing barriers to rapid global optimization. *Journal of global optimization*, 31:549–565, 2005.

[56] D. G. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, Reading, MA, 1984.

[57] G. Malajovich. On generalized newton's methods. *Theoretical Com. Sci.*, 133:65–84, 1994.

[58] J. McNamee. A bibliography on roots of polynomials. *J. Computational and Applied Mathematics*, 47:391–394, 1993. Also see http://www1.elsevier.com/homepage/sac/cam/mcnamee/ for an updated bibliography.

[59] B. Paden and S. Sastry. A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators. *IEEE Transactions on Circuits and Systems*, 34(1):73–82, 1987.

[60] F. Pazos and A. Bhaya. Algorithm for variational inequality problems based on a gradient dynamical system designed using a control Liapunov function. In *IEEE Multi-conference on Systems and Control*, Cingapura, October 2007.

[61] F. Pazos, A. Bhaya, and E. Kaszkurewicz. Algorithm for convex optimization based on gradient dynamical system designed using a control Liapunov function. In *Proc. of the 16rd Brazilian Conference on Automatic Control*, Salvador, 2006. In Portuguese. Original title: *Algoritmo para otimização convexa baseado em sistema dinâmico gradiente projetado utilizando função de Liapunov de controle.*

[62] R. G. Regis and C. A. Shoemaker. Local function approximation in evolutionary algorithms for the application of costly functions. *IEEE transactions on evolutionary computation*, 8:490–505, 2004.

[63] R. G. Regis and C. A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of global optimization*, 31:153–171, 2005.

[64] G. Resta. Convex programming via functional linear programming. In L. C. W. Dixon and G. P. Szegö, editors, *Towards a Global Optimization 2*, pages 343–362. North Holland Publishing Company, Amsterdam, 1978.

[65] P. S. M. Santos and S. Scheimberg. A projection algorithm for general variational inequalities with perturbed constraint sets. *Applied mathematics and computation*, 181:649–661, 2006.

[66] C. Schaerer and E. Kaszkurewicz. The shooting method for the solution of ordinary differential equations: a control-theoretical perspective. *Internat. J. Systems Science*, 32(8):1047–1053, 2001.

[67] T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization methods and software*, 20(2-3):353–378, 2005.

[68] D. Shevitz and B. Paden. Lyapunov stability theory of nonsmooth systems. *IEEE Trans. Automat. Control*, 39(9), 1994.

[69] J-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[70] G. Soderlind. Automatic control and adaptive time-stepping. *Numerical Algorithms*, 31:281–310, 2002.

[71] M. V. Solodov. A class of globally convergent algorithms for pseudomonotone variational inequalities. In M. C. Ferris, O. L. Mangasarian, and J. S. Pang, editors, *Applications, algorithms and extensions*, chapter 14, pages 297–315. Kluwer academic publishers, 2001.

[72] M. V. Solodov. Convergence rate analysis of iteractive algorithms for solving variational inequality problems. *Mathematical Programming*, 96:513–528, 2003.

[73] M. V. Solodov. Merit functions and error bounds for generalized variational inequalities. *J. Math. Anal. Appl.*, 287:405–414, 2003.

[74] M. V. Solodov and B. F. Svaiter. A new projection method for variational inequality problems. *SIAM J. Control Optim.*, 37(3):765–776, 1999.

[75] M. V. Solodov and P. Tseng. Some methods based on the d-gap function for solving monotone variational inequalities. *Computational optimization and applications*, 17:255–277, 2000.

[76] W. M. Spears. Genetic algorithms (evolutionary algorithms): repository of test functions. Available at http://www.cs.uwyo.edu/˜spears/functs.html, 2004.

[77] G. Treccani. On the convergence of Branin's method: a counter-example. In L. C. W. Dixon and G. P. Szegö, editors, *Towards a Global Optimization*, pages 107–116. North Holland Publishing Company, Amsterdam, 1975.

[78] V. I. Utkin. *Sliding Modes In Control And Optimization*. Springer-Verlag, Berlin, 1992.

[79] T. L. Vincent and W. J. Grantham. *Nonlinear and optimal control systems*. John Wiley, New York, 1997.

[80] Y. Xia and J. Wang. Global exponential stability of recurrent neural networks for solving optimization and related problems. *IEEE Transactions on Neural Networks*, 11(4):1017–1022, 2000.

[81] Y. Xia and J. Wang. A general projection neural network for solving monotone variational inequalities and related optimization problems. *IEEE Transactions on Neural Networks*, 15(2):318–328, 2004.

[82] N. H. Xiu and J. Z. Zhang. On finite convergence of proximal point algorithms for variational inequalities. *Journal of mathematical analysis and applications*, 312:148–158, 2005.

[83] Q. Z. Yang. On variable-step relaxed projection algorithm for variational inequalities. *Journal of mathematical analysis and applications*, 302:166–179, 2005.

[84] L. Yong, K. Lishan, and D. Evans. The annealing evolution algorithm as function optimizer. *Parallel Computing*, 21:389–400, 1995.

[85] X. M. Yuan. The improvement with relative errors of the He *et.al.*'s inexact alternating direction method for monotone variational inequalities. *Mathematical and computer modelling*, 42:1225–1236, 2005.

[86] S. H. Żak. *Systems and Control*. Oxford University Press, New York, 2003.

[87] P. J. Zufiria and R. S. Guttalu. On the role of singularities in the Branin's method from dynamic and continuation perspectives. *Applied mathematics and computation*, 130:593–618, 2002.