



IVAN XAVIER MOURA DO NASCIMENTO

OTIMIZAÇÃO SEM DERIVADAS: SOBRE A CONSTRUÇÃO E A
QUALIDADE DE MODELOS QUADRÁTICOS NA SOLUÇÃO DE
PROBLEMAS IRRESTRITOS

CAMPINAS
2014



UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística
e Computação Científica

IVAN XAVIER MOURA DO NASCIMENTO

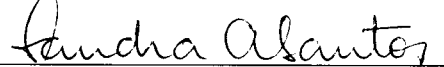
OTIMIZAÇÃO SEM DERIVADAS: SOBRE A CONSTRUÇÃO E A
QUALIDADE DE MODELOS QUADRÁTICOS NA SOLUÇÃO DE
PROBLEMAS IRRESTRITOS

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada.

Orientadora: Sandra Augusta Santos

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO IVAN XAVIER MOURA DO NASCIMENTO, E ORIENTADA PELA PROFA. DRA. SANDRA AUGUSTA SANTOS.

Assinatura da Orientadora


/

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

N17o Nascimento, Ivan Xavier Moura do, 1989-
Otimização sem derivadas : sobre a construção e a qualidade de modelos quadráticos na solução de problemas irrestritos / Ivan Xavier Moura do Nascimento. – Campinas, SP : [s.n.], 2014.

Orientador: Sandra Augusta Santos.

Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Otimização sem derivadas. 2. Otimização irrestrita. 3. Interpolação quadrática. 4. Lagrange, Funções de. 5. Posicionamento (Matemática). 6. Experimentos numéricos. I. Santos, Sandra Augusta, 1964-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Derivative-free optimization : on the construction and quality of quadratic models for unconstrained optimization problems

Palavras-chave em inglês:

Derivative-free optimization

Unconstrained optimization

Quadratic interpolation

Lagrange functions

Poisedness

Numerical experiments

Área de concentração: Matemática Aplicada

Titulação: Mestre em Matemática Aplicada

Banca examinadora:

Sandra Augusta Santos [Orientador]

Maria Aparecida Diniz Ehrhardt

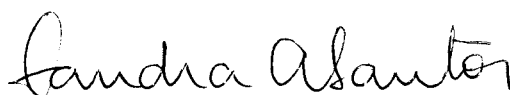
Lucas Garcia Pedroso

Data de defesa: 28-03-2014

Programa de Pós-Graduação: Matemática Aplicada

Dissertação de Mestrado defendida em 28 de março de 2014 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.



Prof.(a). Dr(a). SANDRA AUGUSTA SANTOS



Prof.(a). Dr(a). MARIA APARECIDA DINIZ EHRHARDT



Prof.(a). Dr(a). LUCAS GARCIA PEDROSO

Abstract

Trust-region methods are a class of iterative algorithms widely applied to nonlinear unconstrained optimization problems for which derivatives of the objective function are unavailable or inaccurate. One of the classical approaches involves the optimization of a polynomial model for the objective function, built at each iteration and based on a sample set.

In a recent work, Scheinberg and Toint [*SIAM Journal on Optimization*, 20 (6) (2010), pp. 3512–3532] proved that, despite being essential for convergence results, the improvement of the geometry (poisedness) of the sample set might occur only in the final stage of the algorithm. Based on these ideas and incorporating them into a theoretical algorithm framework, the authors investigate analytically an interesting self-correcting geometry mechanism of the interpolating set, which becomes evident at unsuccessful iterations. Global convergence for the new algorithm is then proved as a consequence of this self-correcting property.

In this work we study the positioning of the sample points within interpolation-based methods that rely on quadratic models and investigate the computational performance of the theoretical algorithm proposed by Scheinberg and Toint, whose parameters are based upon either choices of previous works or numerical experiments.

Keywords: derivative-free optimization; nonlinear programming; unconstrained optimization; quadratic interpolation; Lagrange functions; poisedness; numerical experiments.

Resumo

Métodos de região de confiança formam uma classe de algoritmos iterativos amplamente utilizada em problemas de otimização não linear irrestrita para os quais as derivadas da função objetivo não estão disponíveis ou são imprecisas. Uma das abordagens clássicas desses métodos envolve a otimização de modelos polinomiais aproximadores para a função objetivo, construídos a cada iteração com base em conjuntos amostrais de pontos.

Em um trabalho recente, Scheinberg e Toint [*SIAM Journal on Optimization*, 20 (6) (2010), pp. 3512–3532] mostram que apesar do controle do posicionamento dos pontos amostrais ser essencial para a convergência do método, é possível que tal controle ocorra de modo direto apenas no estágio final do algoritmo. Baseando-se nessas ideias e incorporando-as a um esquema algorítmico teórico, os autores investigam analiticamente uma curiosa propriedade de autocorreção da geometria dos pontos, a qual se evidencia nas iterações de insucesso. A convergência global do novo algoritmo é, então, obtida como uma consequência da geometria autocorretiva.

Nesta dissertação estudamos o posicionamento dos pontos em métodos baseados em modelos quadráticos de interpolação e analisamos o desempenho computacional do algoritmo teórico proposto por Scheinberg e Toint, cujos parâmetros são determinados com base em escolhas da literatura ou por meio de experimentos numéricos.

Palavras-chave: otimização sem derivadas; programação não linear; otimização irrestrita; interpolação quadrática; funções de Lagrange; posicionamento; experimentos numéricos.

Sumário

Dedicatória	ix
Agradecimentos	x
1 Introdução	1
2 Interpolação Polinomial	4
2.1 O uso de modelos em métodos de região de confiança	4
2.2 Aproximações de Taylor	5
2.3 Interpolação Polinomial - caso completo	6
2.4 Polinômios de Lagrange	11
2.5 Algumas propriedades dos polinômios de Lagrange	12
2.6 Λ -posicionamento	16
2.7 Propriedades do Λ -posicionamento	18
3 Um método de região de confiança	23
3.1 DFO-LP	25
3.2 Convergência global	27
4 Uma implementação para o DFO-LP	29
4.1 <i>Software</i> para a implementação computacional	29
4.2 Representação do conjunto amostral de pontos e polinômios de Lagrange	29
4.3 Inicialização do Algoritmo	31
4.3.1 O conjunto inicial de pontos amostrais	31
4.3.2 O modelo aproximador inicial	32
4.3.3 A base inicial de polinômios de Lagrange	34
4.4 O parâmetro ϵ_0	37
4.5 A construção de um modelo Λ -posicionado	38
4.5.1 Otimização Global de $ \ell_j(x) $	39
4.6 Atualização da matriz de coeficientes dos polinômios de Lagrange	39
4.7 O subproblema de região de confiança	40
4.8 O descarte de pontos em iterações de insucesso	40

4.9	Atualização do raio da região de confiança	41
5	Metodologia de comparação de <i>solvers</i> sem derivada	42
5.1	Teste de Convergência	42
5.2	Perfis de desempenho e informativos	43
5.2.1	Perfis de desempenho	44
5.2.2	Perfis informativos	44
6	Interface entre CUTer e MATLAB	46
6.1	Arquivos MEX	46
6.2	Instalando o CUTer	46
6.3	Procedimentos a partir do MATLAB	47
6.4	Determinando a dimensão dos problemas	49
6.4.1	Tratando de dimensões múltiplas	51
6.5	Dinâmica dos Testes Computacionais	51
7	Experimentos Computacionais	54
7.1	<i>Solvers</i> para otimização sem derivadas	54
7.1.1	DF0-LP	55
7.1.2	SID-PSM	55
7.1.3	Nelder-Mead	56
7.2	Problemas teste	57
8	Análise de Resultados	60
8.0.1	Tratamento <i>a posteriori</i> dos dados coletados	61
8.1	Primeiro bloco de testes: o parâmetro ϵ_0	62
8.1.1	Os efeitos do testes de criticalidade no orçamento	67
8.2	Segundo bloco de testes: a comparação dos <i>solvers</i> de \mathcal{S}	70
8.2.1	Problemas não resolvidos pelo DF0-LP1	77
9	Considerações finais e perspectivas futuras	81
	Referências	83

À minha amada mãe, Dona Sonia.

Agradecimentos

À Sandra, pelas inúmeras horas carinhosamente dedicadas a este projeto.

À Cheti, ao Lucas Pedroso e ao Luís Felipe, pela enorme contribuição com valiosas sugestões, orais e escritas, para a versão final desta dissertação.

Ao Abel e ao Ranieri, por toda a assitência com o CUTer e sua interface com o MATLAB.

Ao Gilli, à Margarida e ao Mujica, pela confiança depositada em mim através de suas cartas de recomendação para o ingresso no programa de mestrado.

À Dona Sonia, minha mãe, e à Karen, minha eterna irmãzinha, por estarem presentes durante a defesa.

Aos meus queridos amigos, em especial aos do Barraco e seus agregados, e também aos meus familiares, por todo o apoio e paciência que tiveram comigo nesses tempos de ausência e trabalho árduo.

Aos meus professores do fundamental e médio, pela coragem e perseverança como docentes do ensino público. Em especial, às professoras Neuza, Maria Laura, Sumara, Maria Angélica, Adenilda, Mônica, Rosângela, Deise, Rosa, Denise e Fernanda.

À Capes, pelo apoio financeiro.

Introdução

Desde os fundamentos mais elementares do Cálculo, a determinação de máximos e mínimos de funções reais de uma variável envolve, além de exigências relacionadas à continuidade e à diferenciabilidade da função, a análise de pontos nos quais a **derivada** da função se anula [28]. No caso mais geral de otimização não linear, as diversas versões das condições necessárias e suficientes para problemas de otimização também envolvem o uso de derivadas, seja da função objetivo ou das possíveis restrições do problema, na determinação de máximos e mínimos, tanto locais como globais [6]. Nesse sentido, a abordagem clássica de resolução desses problemas está apoiada no uso de derivadas, calculadas diretamente ou por aproximação.

Contudo, por uma série de motivos, algumas situações de interesse prático inviabilizam o uso das derivadas envolvidas no problema. A indisponibilidade, o custo e a presença de incerteza nas medidas das derivadas são os principais deles. Por exemplo, quando a função objetivo é cara de ser avaliada ou quando há interferência de ruídos em seus valores, o uso de métodos de aproximação de derivada através de diferenças finitas pode ser pouco útil.

Outro exemplo clássico é o problema em que a função objetivo é resultado de um processo complexo de simulação, computacional ou não. Nesses casos, é muito provável que não exista uma fórmula explícita para as derivadas da função e, dependendo do tempo e dos custos de simulação, qualquer método de aproximação de derivadas será considerado impraticável. Há, ainda, os casos em que a função objetivo está relacionada a códigos binários, proprietários ou que, simplesmente, não foram mantidos por seus autores.

Quando se abre mão de uma informação de tão grande valor como as derivadas, faz-se necessário repensar as maneiras de abordar os problemas. Esse cenário de indisponibilidade de derivadas da função objetivo e das restrições do problema será chamado resumidamente de **otimização sem derivadas**. Por concisão, métodos especialmente desenvolvidos com essas características e propósitos, nos quais as derivadas do problema não podem ser explicitamente usadas, serão chamados de **métodos ou solvers sem derivada**.

De acordo com um levantamento feito em [19], os primeiros trabalhos em otimização sem derivada datam da década de 1960 [29, 38, 52] e, portanto, considera-se que seu desenvolvimento tenha ocorrido de maneira rápida, estando frequentemente associado a aplicações práticas em problemas científicos [18, 58], médicos [34, 39], de engenharia [3, 4], etc.

É preciso destacar, também, as limitações da otimização sem derivadas. Em [14, Seção 1.3],

é ressaltado o caráter de pequeno porte dos problemas para os quais métodos sem derivada são adequados, ainda que com o advento da computação paralela tal limitação tenha sido amenizada. Além disso, nessa mesma seção, os autores afirmam não ser realista a expectativa por soluções acuradas. De fato, em um grande trabalho experimental que envolveu 22 dos melhores *solvers* sem derivadas e 502 problemas-teste, Rios e Sahinidis [45] concluíram que, mesmo para problemas de pequeno porte, a obtenção das melhores soluções foi um desafio para a maioria dos *solvers*. É consenso entre esses autores que a taxa de convergência local de métodos de otimização sem derivadas, mesmo quando são utilizados modelos de aproximação quadráticos, está mais próxima à linear do que à quadrática. Igualmente, apesar de apresentar bons resultados em problemas moderadamente com ruídos e descontinuidades, as demonstrações de convergência, em geral, assumem a hipótese de Lipschitz continuidade sobre as primeiras derivadas da função objetivo.

Portanto, de maneira geral, métodos de otimização sem derivadas são apropriados para funções com alto custo de avaliação, funções que produzam dados moderadamente imprecisos (com ruídos), funções razoavelmente suaves e de dimensão não muito maior do que algumas dezenas de variáveis. Ademais, tais métodos são adequados para casos em que não se ambicione uma taxa de convergência alta (principalmente quando comparada a *solvers* com derivadas) e, sobretudo, quando apenas alguns dígitos de precisão são requisitados. Existe, ainda, uma situação em que tais métodos podem ser bem aceitos: no caso de não familiaridade do usuário com métodos de otimização que exijam algum conhecimento matemático mais avançado. Métodos sem derivadas costumam ser simples, sob o ponto de vista do usuário, que tem que fornecer poucos dados de entrada para a execução da rotina.

Métodos de otimização sem derivada, *grosso modo*, podem ser separados em dois grandes grupos: o de **métodos de busca direta**, nos quais as direções de busca são determinadas pelo cálculo do valor da função objetivo diretamente; e o de **métodos baseados em modelos**, os quais se apóiam na construção de modelos para guiar o processo de busca de direções. Há um certo dualismo entre esses dois grupos, no seguinte sentido: métodos de busca direta primeiramente buscam uma direção (de descida) e depois determinam o tamanho do passo a ser tomado, enquanto que os métodos baseados em modelos limitam o tamanho do passo a uma região para, posteriormente, determinar a direção a ser seguida.

Muitos trabalhos recentes têm sido publicados sobre a convergência de métodos de otimização sem derivadas. Em [14, Seção 1.4], destacam-se três atributos que estão presentes em todos os algoritmos (sem derivadas) **globalmente convergentes**, isto é, algoritmos que, independentemente do ponto inicial, são capazes de gerar sequências de iterandos que se aproximam assintoticamente de pontos estacionários. O primeiro deles é a garantia de **decrécimo suficiente** no valor de função fora de pontos estacionários. Em seguida, aparece algum tipo de controle da **geometria dos pontos amostrais**, que garanta que a medida de estacionaridade seja, de fato, válida. Por último, algoritmos de otimização sem derivadas globalmente convergentes devem conduzir a uma diminuição gradual do tamanho do passo, em direção a zero.

Nesse contexto, o presente trabalho se propõe a investigar as propriedades de autocontrole da geometria dos pontos amostrais do algoritmo teórico de região de confiança sem derivadas, proposto por Scheinberg e Toint [47], chamado aqui de **DF0-LP**, e apresentar experimentos numéricos realizados com uma implementação própria desse algoritmo, comparando-a segundo a metodologia de Moré e Wild [37] com um conjunto \mathcal{S} de *solvers* sem derivada e um conjunto de problemas-teste

extraído de Fasano, Morales e Nocedal [21].

Esta dissertação pode ser dividida em duas partes. Na primeira, majoritariamente teórica, são introduzidos os principais conceitos sobre a construção de modelos de interpolação quadráticos e a relação de sua geometria com os polinômios de Lagrange associados ao conjunto de pontos de interpolação, no Capítulo 2, seguidas pela apresentação do algoritmo **DF0-LP** no Capítulo 3, inserido em seu cenário de região de confiança. Apenas no Capítulo 4 se definem as diretrizes da implementação computacional do algoritmo, antes de estabelecer, já no Capítulo 5, a metodologia de comparação de *solvers* sem derivada a ser empregada.

Na segunda parte, essencialmente experimental, é esclarecida a interface entre **CUTEr** e **MATLAB** (Capítulo 6) utilizada na geração dos dados experimentais do Capítulo 7. Por fim, nos Capítulos 8 e 9, são realizadas respectivamente as análises de resultados e as conclusões a respeito deste trabalho.

Sobre a notação utilizada

De um modo geral, as nomenclaturas adotadas neste trabalho são introduzidas à medida que se fazem necessárias e não se alteram no decorrer do texto. As exceções são o Capítulo 3 e as Seções 4.8 e 4.9, onde os superíndices passam a corresponder a iterações de algoritmos. Consequentemente, são introduzidas modificações no significado de alguns subíndices a fim de não sobrecarregar a notação. Entretanto, tais redefinições encontram-se sinalizadas e são esclarecidas por seu contexto. Ainda, as convenções adotadas e algumas definições fundamentais são apresentadas em forma de nota de rodapé no decorrer do trabalho.

Interpolação Polinomial

Este capítulo tem por finalidade introduzir o uso de modelos polinomiais completos na resolução do problema geral de otimização irrestrita e, a partir deles, estabelecer relações entre a geometria do conjunto amostral de pontos e os polinômios de Lagrange associados a esse conjunto.

2.1 O uso de modelos em métodos de região de confiança

Seja $f : \mathbb{R}^n \rightarrow \mathbb{R}$ uma função não linear e suponha que suas derivadas de primeira ordem existam e sejam Lipschitz contínuas. O principal interesse deste trabalho está na resolução do problema irrestrito

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} f(x) \quad (2.1.1)$$

em um contexto prático onde as derivadas da função não estão disponíveis ou não são precisas o suficiente para certa aplicação.

A estratégia de resolução a ser adotada aqui consiste, essencialmente, na substituição de (2.1.1) por sucessivos problemas restritos mais fáceis de serem resolvidos. Partindo de um ponto inicial $x^0 \in \mathbb{R}^n$ e de uma vizinhança de raio $\Delta_0 \in \mathbb{R}$ em torno dele, a função objetivo é trocada por um modelo que a aproxime *razoavelmente bem* na bola fechada¹ $\mathcal{B}(x^0, \Delta_0)$ para que, então, seja realizada a otimização do novo problema, agora restrito a essa região, em busca de um ponto que diminua *suficientemente* o valor da função original. Em seguida, a solução “local” passa a ser o novo ponto central e, com a alteração do raio da vizinhança e do modelo aproximador da função na nova região, o processo é iterativamente repetido até que, sob algum critério, o ponto corrente de alguma iteração seja decretado como solução final do problema original.

O esquema anterior descreve panoramicamente os chamados **métodos de região de confiança** e deixa em aberto diversas questões a respeito das etapas de sua execução. Desde a escolha do ponto e raio iniciais, passando pelo significado matemático de “diminuir suficientemente”, até a alteração do raio da região e o critério de terminação do método, todas as decisões terão implicações teóricas, na demonstração da convergência do método, e práticas, no desempenho do algoritmo.

¹ Dados $x \in \mathbb{R}^n$ e um raio $\Delta \in \mathbb{R}$, a bola fechada $\mathcal{B}(x, \Delta)$ é o conjunto de pontos que distam no máximo Δ do ponto x , na norma Euclidiana. Isto é, $\mathcal{B}(x, \Delta) = \{y \in \mathbb{R}^n; \|x - y\|_2 \leq \Delta\}$.

No Capítulo 3 serão abordados maiores detalhes sobre essa classe de métodos. Por ora, o interesse está direcionado para a construção do modelo que aproxima localmente a função objetivo. É a maneira de construí-lo que está relacionada com quão “razoavelmente boa” é a aproximação e quão mais fácil é sua otimização quando comparada à da função original.

2.2 Aproximações de Taylor

Sabe-se que uma função analítica é aquela que pode ser representada em determinada região através de sua expansão em série de Taylor. Uma maneira de obter uma aproximação para essa função é truncando a série de Taylor que a representa. De um modo geral, quanto mais termos da série são considerados, menor é o erro cometido na aproximação. Para funções continuamente deriváveis, também é possível obter uma aproximação (e fórmulas explícitas para seu resíduo) através da expansão em Taylor. Sendo a função f duas vezes diferenciável, uma aproximação de segunda ordem em torno de um ponto \bar{x} de seu domínio é dada por:

$$f(x) \approx f(\bar{x}) + \nabla f(\bar{x})^\top (x - \bar{x}) + \frac{1}{2}(x - \bar{x})^\top \nabla^2 f(\bar{x})(x - \bar{x}), \quad (2.2.1)$$

onde $\nabla f(x)$ e $\nabla^2 f(x)$ são, respectivamente, o gradiente e a Hessiana da função, e são definidos através das derivadas parciais de f com relação às componentes x_1, x_2, \dots, x_n de $x \in \mathbb{R}^n$ da seguinte maneira:

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \quad \text{e} \quad \nabla^2 f(x) := \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}. \quad (2.2.2)$$

É importante destacar que se f possui derivadas de segunda ordem contínuas em uma região \mathcal{D} então $\nabla^2 f(x)$ é simétrica em \mathcal{D} .

Observa-se que em (2.2.1) surge naturalmente uma forma polinomial de grau dois cujos coeficientes estão relacionados com as derivadas da função objetivo. Tipicamente, em otimização com derivadas, o modelo aproximador da função objetivo é construído com base nesse polinômio, cujos coeficientes são os valores exatos ou aproximados para o gradiente e a Hessiana. De fato, o modelo polinomial mais simples de ser tratado é o de grau um, dependente somente de $\nabla f(\bar{x})$. No entanto, por se tratar de uma forma linear, incapaz de captar as curvaturas da função que está aproximando, é comum a utilização de modelos polinomiais de grau dois, ainda que sua construção seja, em geral e em algum sentido, mais onerosa.

A teoria de aproximação de funções continuamente diferenciáveis através do truncamento de sua série de Taylor está munida de fórmulas para os limitantes do erro cometido nessas aproximações. Em geral, tais limitantes são dados em termos das derivadas da função utilizadas na construção dos modelos polinomiais.

Métodos sem derivada também podem se basear na concepção de modelos polinomiais, desde que nenhuma informação sobre as derivadas da função objetivo seja necessária para sua construção. Adicionalmente, a técnica de formação do modelo deve prover alguma estimativa sobre os limitantes para o erro associado à aproximação. O desafio que se apresenta consiste, portanto, em estudar técnicas de geração de modelos polinomiais de maneira que essas forneçam garantias teóricas sobre o erro cometido ou, em outras palavras, que garantam a construção e a qualidade desses objetos. Assim como em métodos com derivada, modelos polinomiais quadráticos são os mais comumente adotados para fins de aproximar a função objetivo. De acordo com Powell [40], a ideia de usar interpolação na construção de modelos quadráticos quando as derivadas da função objetivo não estão disponíveis foi proposta por Winfield [57] em 1973.

2.3 Interpolação Polinomial - caso completo

Considere, então, a tarefa de aproximar uma função objetivo $f : \mathbb{R}^n \rightarrow \mathbb{R}$ em uma região em torno de um ponto de interesse, através de um modelo $m : \mathbb{R}^n \rightarrow \mathbb{R}$ polinomial quadrático, da forma

$$m(x) = c + g^\top (x - \bar{x}) + \frac{1}{2}(x - \bar{x})^\top H(x - \bar{x}), \quad (2.3.1)$$

escrito em torno de \bar{x} , com $g \in \mathbb{R}^n$ e $H = H^\top \in \mathbb{R}^{n \times n}$, cuja construção deva depender somente dos valores de f avaliados em um conjunto amostral (discreto)

$$\mathcal{Y} = \{y^1, y^2, \dots, y^p\}$$

de pontos $y^j \in \mathbb{R}^n$, $j = 1, \dots, p$. A condição de que o modelo m deva satisfazer os valores da função f em \mathcal{Y} conduz a uma abordagem chamada de **interpolação polinomial**. O número $p = \text{card}(\mathcal{Y})$ é um inteiro positivo que define, portanto, o número de pontos no conjunto de interpolação².

Dessa forma, o modelo dado pela Equação (2.3.1) nada mais é do que um elemento pertencente ao espaço \mathcal{P}_n^2 de polinômios de grau menor ou igual a 2 nas variáveis x_1, x_2, \dots, x_n com coeficientes em \mathbb{R} . Se

$$\Phi = \{\phi_1(x), \phi_2(x), \dots, \phi_q(x)\}$$

é uma base ordenada para esse espaço, então, da definição de base, segue que o polinômio $m(x)$ pode ser escrito como combinação linear dos elementos de Φ com coeficientes $\alpha_j \in \mathbb{R}$, ou seja,

$$m(x) = \sum_{j=1}^q \alpha_j \phi_j(x). \quad (2.3.2)$$

² Sendo A um conjunto finito qualquer, denota-se por $\text{card}(A)$ a quantidade de elementos que ele possui, isto é, sua cardinalidade.

Um exemplo de base ordenada para \mathcal{P}_n^2 é aquela que surge *naturalmente* da expansão de Taylor em torno da origem,

$$\bar{\Phi} = \left\{ 1, x_1, x_2, \dots, x_n, x_1x_2, x_1x_3, \dots, x_{n-1}x_n, \frac{1}{2}x_1^2, \frac{1}{2}x_2^2, \dots, \frac{1}{2}x_n^2 \right\}, \quad (2.3.3)$$

com $x = (x_1, x_2, \dots, x_n)^\top \in \mathbb{R}^n$, por isso conhecida como *base natural*. Da cardinalidade de $\bar{\Phi}$ não é difícil concluir que a dimensão q de \mathcal{P}_n^2 é dada por

$$q = \frac{(n+1)(n+2)}{2}. \quad (2.3.4)$$

As condições de interpolação podem ser matematicamente expressas através da equação

$$m(y^j) = f(y^j), \quad j = 1, \dots, p, \quad (2.3.5)$$

ou, ainda, usando a Equação (2.3.2), através do sistema linear

$$\begin{bmatrix} \phi_1(y^1) & \phi_2(y^1) & \cdots & \phi_q(y^1) \\ \phi_1(y^2) & \phi_2(y^2) & \cdots & \phi_q(y^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(y^p) & \phi_2(y^p) & \cdots & \phi_q(y^p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_q \end{bmatrix} = \begin{bmatrix} f(y^1) \\ f(y^2) \\ \vdots \\ f(y^p) \end{bmatrix}. \quad (2.3.6)$$

Observe que a matriz do sistema linear da Equação (2.3.6) está intimamente relacionada com o conjunto amostral \mathcal{Y} e com a base Φ do espaço \mathcal{P}_n^2 . Portanto, por questão de clareza e concisão, a essa matriz será dado o nome de $M(\Phi, \mathcal{Y})$ e o sistema anterior será simplificadaamente escrito como

$$M(\Phi, \mathcal{Y})\alpha = f(\mathcal{Y}), \quad (2.3.7)$$

onde $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_q)^\top \in \mathbb{R}^q$ e $f(\mathcal{Y}) = (f(y^1), f(y^2), \dots, f(y^p))^\top \in \mathbb{R}^p$.

A menos que se imponham condições sobre g e H , no mínimo q pontos são necessários para determinar g e H unicamente. Por outro lado, se $p > q$, o sistema linear dado pela Equação (2.3.7) possivelmente não admite solução³. Nesse sentido, parece razoável fixar a cardinalidade de \mathcal{Y} de maneira que a matriz $M(\Phi, \mathcal{Y})$ seja quadrada, isto é, $p = q$. Isso motiva a primeira definição deste trabalho.

Definição 2.3.1. O conjunto $\mathcal{Y} = \{y^1, y^2, \dots, y^q\}$ é dito **posicionado para interpolação polinomial quadrática**⁴ em \mathbb{R}^n se a matriz $M(\Phi, \mathcal{Y})$ correspondente for não singular para alguma base Φ de \mathcal{P}_n^2 . Nesse sentido, o ato ou efeito de posicionar será chamado de **posicionamento**⁵.

³ De fato, se $p > q$, é possível considerar modelos não lineares de regressão [14, Capítulo 4]. Entretanto, tal abordagem está fora do escopo deste trabalho.

⁴ Ou 2-unisolvente, segundo Ciarlet e Raviart [9].

⁵ O substantivo “posicionamento” foi adotado como tradução da palavra *poisedness*, do inglês. A correspondência entre a tradução das famílias dessas palavras é análoga.

A fim de ilustrar certos conceitos, dois conjuntos de pontos serão tomados como exemplos e revisitados sempre que necessário: um para o caso unidimensional e outro para o bidimensional.

Exemplo 2.3.2. Um caso unidimensional ($n = 1$)

Considere o simples caso de interpolação polinomial no espaço \mathcal{P}_1^2 dos polinômios de grau menor ou igual a dois em uma variável. Como $n = 1$, então $q = 3$. A base natural para esse espaço é

$$\bar{\Phi} = \left\{1, x, \frac{x^2}{2}\right\},$$

onde $x \in \mathbb{R}$. Considere também o conjunto amostral (genérico)

$$\mathcal{Y} = \{a, b, c\} \subset \mathbb{R}.$$

Assim, a matriz de interpolação associada à base ordenada $\bar{\Phi}$ e ao conjunto \mathcal{Y} é dada por

$$M(\bar{\Phi}, \mathcal{Y}) \equiv M = \begin{bmatrix} 1 & a & \frac{a^2}{2} \\ 1 & b & \frac{b^2}{2} \\ 1 & c & \frac{c^2}{2} \end{bmatrix},$$

e não é difícil ver que seu determinante vale

$$\det M = \frac{1}{2}(a-b)(b-c)(c-a).$$

Portanto, M é não singular para quaisquer pontos a, b e c reais distintos entre si. Ou seja, da Definição 2.3.1, é possível concluir que quaisquer três pontos distintos (dois a dois) de \mathbb{R} formam um conjunto posicionado para interpolação polinomial quadrática em uma variável.

Note que o posicionamento de um conjunto independe da função que se quer aproximar. Supondo que $f(x) = e^x$, resolvendo o sistema linear dado em (2.3.7) e associando cada componente da solução ao elemento da base $\bar{\Phi}$ correspondente, obtém-se o modelo quadrático

$$m(x) = \frac{be^c - ce^b}{b-c} + \frac{(e^b - e^c)}{b-c}x + \left(\frac{e^a - e^b}{a-b} - \frac{e^a - e^c}{a-c}\right) \frac{(b-x)(c-x)}{b-c}. \quad (2.3.8)$$

Para efeitos ilustrativos, admita que $a = -1$, $b = 1$ e $c = 4$. Dessa forma, substituindo os valores em 2.3.8, é possível recuperar o modelo

$$m(x) \approx -1.7 + 1.2x + 3.2x^2, \quad (2.3.9)$$

cuja representação gráfica pode ser vista na Figura 2.1. Conforme imposto, o valor do modelo nos pontos de \mathcal{Y} coincidem com o valor da função aproximada. \diamond

Exemplo 2.3.3. Um caso bidimensional ($n = 2$)

No caso bidimensional, a dimensão do espaço \mathcal{P}_2^2 de polinômios de grau menor ou igual a dois em duas variáveis (x e y) é $q = 6$, e a base natural para esse espaço é

$$\bar{\Phi} = \left\{1, x, y, xy, \frac{x^2}{2}, \frac{y^2}{2}\right\}.$$

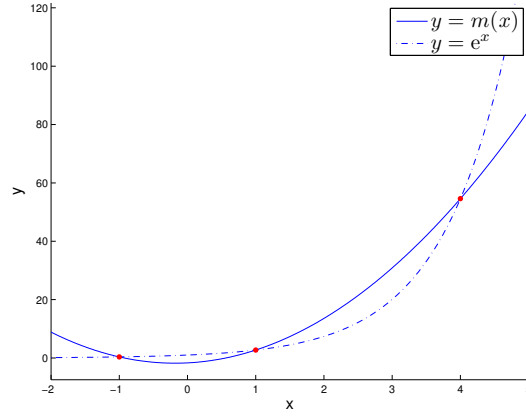


Figura 2.1: Aproximação de $f(x) = e^x$ através do modelo $m(x)$ construído sobre o conjunto \mathcal{Y} do Exemplo 2.3.2.

Considere o conjunto $\mathcal{Y} = \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\} \subset \mathbb{R}^2$, ilustrado na Figura 2.2. A matriz M de interpolação associada a \mathcal{Y} e $\bar{\Phi}$ é

$$M(\bar{\Phi}, \mathcal{Y}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0.5 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0.5 \\ 1 & -1 & 1 & -1 & 0.5 & 0.5 \\ 1 & 0 & -1 & 0 & 0 & 0.5 \\ 1 & 1 & -1 & -1 & 0.5 & 0.5 \end{bmatrix},$$

e $\det(M) = -1$, o que significa que \mathcal{Y} é um conjunto posicionado para interpolação quadrática.

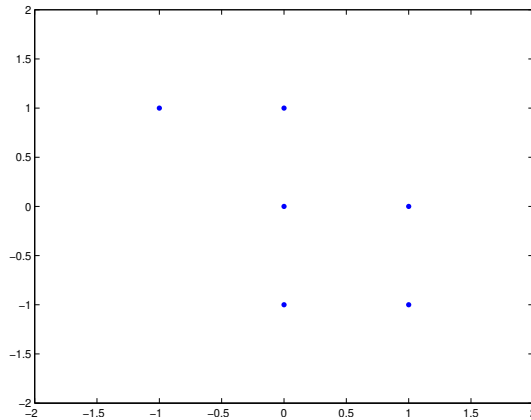


Figura 2.2: Representação do conjunto amostral \mathcal{Y} do Exemplo 2.3.3.

Introduzindo a função $f(x, y) = \sin xy$ no problema, tem-se

$$f(\mathcal{Y}) = (0, 0, 0, \sin(-1), 0, \sin(-1))^T$$

e o sistema de interpolação admite a solução (única) $\alpha = (0, 0, 0, \sin 1, 0, 0)^T$, correspondente ao modelo

$$m(x, y) = (\sin 1) xy \approx 0.84 xy. \quad (2.3.10)$$

A Figura 2.3 apresenta os gráficos da função f e do modelo m . ◇

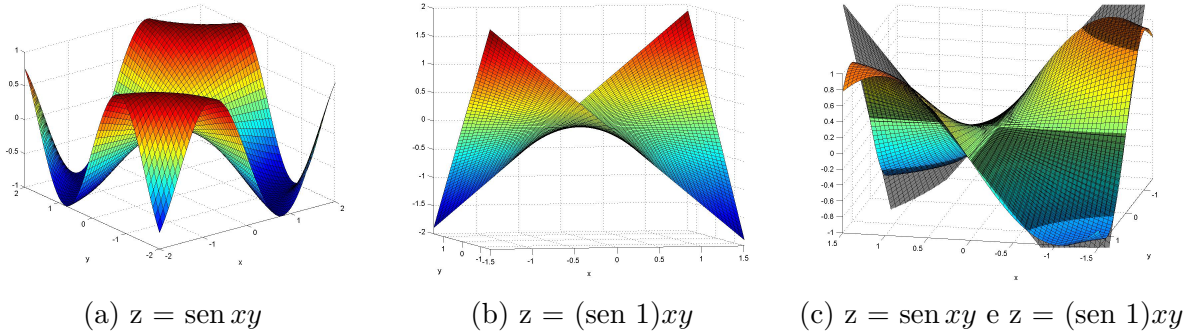


Figura 2.3: Gráficos das funções do Exemplo 2.3.3.

Dado que, em um espaço vetorial de dimensão finita, uma matriz de mudança de base é sempre não singular, é fácil ver que o posicionamento é invariante pela escolha da base. Além disso, se $\mathcal{Y} \subset \mathbb{R}^n$ é um conjunto posicionado, então o sistema linear dado pela Equação (2.3.7) admite solução $\alpha \in \mathbb{R}^q$ única. Em outras palavras, o posicionamento do conjunto $\mathcal{Y} \in \mathbb{R}^n$ garante a existência e a unicidade do polinômio interpolador $m \in \mathcal{P}_n^2$ para esse conjunto. Portanto, o posicionamento de \mathcal{Y} deve ser levado em consideração durante o processo de amostragem, caso se queira ter garantidas a existência e a unicidade do polinômio interpolador. A singularidade de $M(\Phi, \mathcal{Y})$ é a primeira característica a que se pode atribuir um significado essencialmente geométrico de \mathcal{Y} , mas que não serve como critério de comparação entre, por exemplo, dois conjuntos já posicionados. Isso evidencia uma necessidade em se medir o quão **bom** é o posicionamento de um dado conjunto.

É natural que o número de condição de $M(\Phi, \mathcal{Y})$ apareça como um palpite inicial para a maneira de mensurar o posicionamento de \mathcal{Y} . Entretanto, no caso geral, o condicionamento dessa matriz não serve para a caracterização do nível de posicionamento de um conjunto de pontos. Em particular, fixado \mathcal{Y} , é possível escolher a base Φ de maneira que o número de condição da matriz de interpolação correspondente assuma qualquer valor entre 1 e $+\infty$. Além disso, fixada a base Φ , o número de condição de $M(\Phi, \mathcal{Y})$ varia com a multiplicação de \mathcal{Y} por uma matriz de escalamento real. Ainda assim, é possível estabelecer algumas relações entre $\text{cond}(M(\Phi, \mathcal{Y}))$ e o (bom) posicionamento dos pontos amostrais, para o caso particular no qual \mathcal{Y} está escalado e para a adoção da base natural $\bar{\Phi}$. Maiores esclarecimentos podem ser encontrados em [12, Seção 3]. Aqui, entretanto, outras ferramentas serão adotadas para medir o **bom posicionamento** dos pontos.

Uma das maneiras de medi-lo está relacionada com os polinômios de Lagrange associados ao conjunto amostral. Powell, em um de seus trabalhos, justificou sua preferência por esses polinômios mencionando as diversas propriedades que eles satisfazem [40, pp. 293]. Assim, também em concordância com Scheinberg e Toint [47], a medida de *bom posicionamento* empregada neste trabalho se baseia nos polinômios de Lagrange associados ao conjunto \mathcal{Y} , os quais são introduzidos a seguir.

2.4 Polinômios de Lagrange

Definição 2.4.1. Uma base ordenada $\Phi_\ell = \{\ell_1(x), \ell_2(x), \dots, \ell_q(x)\}$ para \mathcal{P}_n^2 é chamada de base de polinômios de Lagrange associada ao conjunto amostral $\mathcal{Y} = \{y^1, y^2, \dots, y^p\}$ de cardinalidade p igual à dimensão do espaço q , se valem as relações

$$\ell_j(y^i) = \delta_{ij} = \begin{cases} 1, & \text{se } i = j \\ 0, & \text{se } i \neq j \end{cases}, \quad \text{para } i, j = 1, \dots, q. \quad (2.4.1)$$

Se \mathcal{Y} é posicionado, alterando o lado direito da Equação (2.3.7) apropriadamente, pode-se concluir que a base Φ_ℓ de polinômios de Lagrange existe e é unicamente definida. Basta observar que, da Definição 2.4.1, para cada j , $\ell_j(x) \in \Phi_\ell$ é um polinômio interpolador de uma função que se anula em todos os pontos de \mathcal{Y} exceto em y^j , no qual vale 1. Portanto, dada uma base ordenada $\Phi = \{\phi_1(x), \phi_2(x), \dots, \phi_q(x)\}$ para \mathcal{P}_n^2 , o sistema linear de interpolação equivalente ao problema de encontrar o polinômio $\ell_j(x)$ é dado por

$$M(\Phi, \mathcal{Y})\lambda^j = e^j, \quad (2.4.2)$$

onde $\lambda^j \in \mathbb{R}^q$ e e^j é o j -ésimo vetor canônico do \mathbb{R}^q . Consequentemente, a solução λ^j é o vetor cujas componentes λ_i^j são os coeficientes de $\ell_j(x)$ na base Φ , isto é,

$$\ell_j(x) = \sum_{i=1}^q \lambda_i^j \phi_i(x). \quad (2.4.3)$$

Considere, ainda, a matriz $L_\Phi = (l_{ij}) \in \mathbb{R}^{q \times q}$ de coeficientes dos polinômios de Lagrange na base Φ , cujas entradas podem ser definidas como

$$l_{ij} = \lambda_i^j, \quad i, j = 1, \dots, q. \quad (2.4.4)$$

Atentando para o fato de que as Equações (2.4.2) e (2.4.3) são válidas para $j = 1, 2, \dots, q$ e que $M(\Phi, \mathcal{Y})$ é não-singular, pode-se concluir que

$$M(\Phi, \mathcal{Y})L_\Phi^\top = I_q \iff M(\Phi, \mathcal{Y})^{-1} = L_\Phi^\top, \quad (2.4.5)$$

onde I_q é a matriz identidade de ordem q .

Se, por um lado, o posicionamento de \mathcal{Y} garante a existência e a unicidade da base de polinômios de Lagrange, por outro, a existência dos polinômios de Lagrange implica no posicionamento de \mathcal{Y} .

É fácil ver que se a base de polinômios de Lagrange existe e é única, então é possível construir a matriz L_Φ e que esta satisfaz, pela Definição 2.4.1, a equação $M(\Phi, \mathcal{Y})L_\Phi^\top = I_q$, significando que $M(\Phi, \mathcal{Y})$ é não singular e, portanto, que \mathcal{Y} é posicionado. Para provar que o conjunto de polinômios de Lagrange cuja existência foi garantida acima forma uma base de \mathcal{P}_n^2 , basta garantir que qualquer polinômio $m(x)$ nesse espaço pode ser escrito como combinação linear dos polinômios de Lagrange, o que será feito no Lema (2.5.2).

Antes de investigar as propriedades que fazem dos polinômios de Lagrange uma das ferramentas mais úteis de medida da qualidade da geometria dos pontos amostrais, serão dadas algumas definições equivalentes e algumas interpretações geométricas sobre esses polinômios.

2.5 Algumas propriedades dos polinômios de Lagrange

Conforme veremos no Lema 2.5.2, o modelo interpolador é trivialmente expresso em termos dos elementos da base Φ_ℓ de polinômios de Lagrange. Além disso, veremos também que os polinômios de Lagrange são úteis no estabelecimento de um critério de seleção de bons conjuntos de interpolação. Para um melhor entendimento da relação desses polinômios com a geometria do conjunto \mathcal{Y} , alguns resultados são apresentados e comentados a seguir.

Dado o espaço polinomial \mathcal{P}_n^2 e uma base ordenada Φ para esse espaço, seja

$$\Phi(x) = (\phi_1(x), \phi_2(x), \dots, \phi_q(x))^\top$$

um vetor do \mathbb{R}^q cujas componentes são os valores dos elementos da base polinomial Φ em x . É possível enxergar $\Phi(x)$ como uma transformação $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^q$. Com essa notação, pode-se expressar a matriz $M(\Phi, \mathcal{Y}) \in \mathbb{R}^{q \times q}$, associada ao conjunto $\mathcal{Y} = \{y^1, y^2, \dots, y^q\} \subset \mathbb{R}^n$, da seguinte maneira:

$$M(\Phi, \mathcal{Y}) = \begin{bmatrix} \phi_1(y^1) & \phi_2(y^1) & \cdots & \phi_q(y^1) \\ \phi_1(y^2) & \phi_2(y^2) & \cdots & \phi_q(y^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(y^q) & \phi_2(y^q) & \cdots & \phi_q(y^q) \end{bmatrix} = \begin{bmatrix} \Phi(y^1)^\top \\ \Phi(y^2)^\top \\ \vdots \\ \Phi(y^q)^\top \end{bmatrix}.$$

Se \mathcal{Y} é posicionado no sentido da Definição 2.3.1, então $M(\Phi, \mathcal{Y})$ é não singular e, consequentemente, suas linhas formam um conjunto linearmente independente de cardinalidade q . Portanto, é possível expressar o vetor $\Phi(x)$ unicamente em termos de $\Phi(y^i)$, $i = 1, 2, \dots, q$, como

$$\begin{aligned} \Phi(x) &= \sum_{i=1}^q \lambda_i(x) \Phi(y^i), \\ &= M(\Phi, \mathcal{Y})^\top \lambda(x) \end{aligned} \tag{2.5.1}$$

onde $\lambda(x) = (\lambda_1(x), \lambda_2(x), \dots, \lambda_q(x))^T$ é um vetor de polinômios em \mathcal{P}_n^2 .

Lema 2.5.1. Dado um conjunto \mathcal{Y} posicionado, o conjunto $\{\lambda_i(x), i = 1, \dots, q\}$ definido pela Equação (2.5.1) é o conjunto de polinômios de Lagrange associados a \mathcal{Y} .

A demonstração do lema anterior é direta. Substituindo $x = y^i$ na Equação (2.5.1) para cada $i = 1, 2, \dots, q$, da independência linear dos vetores $\Phi(y^i)$, conclui-se que $\lambda_j(y^i) = \delta_{ij}$, $\forall i, j = 1, 2, \dots, q$. Portanto, $\lambda_i(x)$ é o i -ésimo polinômio de Lagrange para o conjunto \mathcal{Y} .

O Lema 2.5.1, junto com a Equação (2.5.1), fornecem uma possível interpretação para o valor dos polinômios de Lagrange avaliados em determinado ponto: esse valor é o peso da contribuição do vetor $\Phi(y^i)$ na combinação linear que forma $\Phi(x)$.

Usando a Regra de Cramer para obter a solução $\lambda(x)$ do sistema linear dado pela Equação (2.5.1), tem-se que

$$\lambda_i(x) = \frac{\det(M(\Phi, \mathcal{Y}_i(x)))}{\det(M(\Phi, \mathcal{Y}))}, \quad (2.5.2)$$

onde $\mathcal{Y}_i(x)$ é o conjunto obtido a partir de \mathcal{Y} através da substituição do ponto y^i por $x \in \mathbb{R}^n$, isto é,

$$\mathcal{Y}_i(x) = \mathcal{Y} \setminus \{y^i\} \cup \{x\}.$$

Conn *et. al.* [14, pp. 41] destacam que, da Equação (2.5.2), é possível notar a independência de $\lambda(x)$ com relação à escolha de Φ , desde que o espaço de polinômios \mathcal{P}_n^2 se mantenha fixo.

Considere, agora, o conjunto $\Phi(\mathcal{Y}) := \{\Phi(y^i), i = 1, \dots, q\}$ em \mathbb{R}^q e seja $\text{vol}(\Phi(\mathcal{Y}))$ o volume da envoltória convexa q -dimensional⁶ de $\Phi(\mathcal{Y})$. Neste caso, a envoltória convexa é também o *simplex*⁷ cujos vértices são os elementos de $\Phi(\mathcal{Y})$. De acordo com Stein [53], o volume de tal simplex é dado por

$$\text{vol}(\Phi(\mathcal{Y})) = \frac{|\det(M(\Phi, \mathcal{Y}))|}{q!}. \quad (2.5.3)$$

Da expressão para o volume do conjunto $\Phi(\mathcal{Y}_i(x))$, análoga à da Equação (2.5.3), aplicada ao valor absoluto de $\lambda_i(x)$ proveniente da Equação (2.5.2), é possível estabelecer a relação

$$|\lambda_i(x)| = \frac{\text{vol}(\Phi(\mathcal{Y}_i(x)))}{\text{vol}(\Phi(\mathcal{Y}))}. \quad (2.5.4)$$

Observe que o valor absoluto do i -ésimo polinômio de Lagrange em um dado ponto $x \in \mathbb{R}^n$ pode ser visto como a porcentagem da alteração do volume da envoltória convexa de $\Phi(\mathcal{Y})$ quando y^i é substituído por x .

O seguinte resultado, extraído de [14, Seção 3.2], mostra como a recuperação do modelo interpolador pode ser diretamente realizada caso se disponha do conjunto de polinômios de Lagrange.

⁶ A envoltória convexa q -dimensional de um conjunto $A \subset \mathbb{R}^q$, denotada por $\text{conv}(A)$, é o menor conjunto convexo que contém A (no sentido de que é a interseção de todos os conjuntos convexos que contém A) [14, p. 30].

⁷ Um simplex em \mathbb{R}^q é a envoltória convexa de $q + 1$ pontos de \mathbb{R}^q . Os $q + 1$ pontos do simplex são chamados de vértices [7].

Lema 2.5.2. Para toda e qualquer função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ e um conjunto $\mathcal{Y} = \{y^1, y^2, \dots, y^q\}$ posicionado em \mathbb{R}^n , o único polinômio $m(x)$ que interpola $f(x)$ em \mathcal{Y} pode ser expresso como

$$m(x) = \sum_{i=1}^q f(y^i) \ell_i(x), \quad (2.5.5)$$

onde $\{\ell_i(x), i = 1, \dots, q\}$ é a base de polinômios de Lagrange para \mathcal{Y} .

O Lema 2.5.2 destaca uma importante propriedade dos polinômios de Lagrange. Uma vez calculados, eles podem ser usados para construir o modelo interpolador a um custo computacional (em operações) consideravelmente baixo. Ainda que para a construção do conjunto de polinômios de Lagrange seja necessário um número adicional de operações, a atualização e manutenção desse conjunto pode ser feita de maneira eficiente e prática, de forma que esse gasto adicional pode ser amenizado, dependendo da implementação do algoritmo.

Exemplo 2.5.3. Associada ao conjunto $\mathcal{Y} = \{-1, 1, 4\}$ do Exemplo 2.3.3 e à respectiva matriz

$$M(\bar{\Phi}, \mathcal{Y}) = \begin{bmatrix} 1 & -1 & \frac{1}{2} \\ 1 & 1 & \frac{1}{2} \\ 1 & 4 & 8 \end{bmatrix} \text{ está a matriz } L = \begin{bmatrix} \frac{2}{5} & -\frac{1}{2} & \frac{1}{5} \\ \frac{2}{3} & \frac{1}{2} & -\frac{1}{3} \\ -\frac{1}{15} & 0 & \frac{2}{15} \end{bmatrix}, \text{ inversa da transposta de } M(\bar{\Phi}, \mathcal{Y}).$$

Utilizando os elementos de L , é possível recuperar os polinômios de Lagrange associados a \mathcal{Y} :

$$\ell_1(x) = \frac{2}{5} - \frac{1}{2}x + \frac{1}{10}x^2; \quad \ell_2(x) = \frac{2}{3} + \frac{1}{2}x - \frac{1}{6}x^2; \quad \ell_3(x) = -\frac{1}{15} + \frac{1}{15}x^2.$$

A Figura 2.4 ilustra a condição (2.4.1), que estabelece que cada polinômio de Lagrange deve valer 1 no ponto ao qual corresponde e se anular nos demais pontos de \mathcal{Y} .

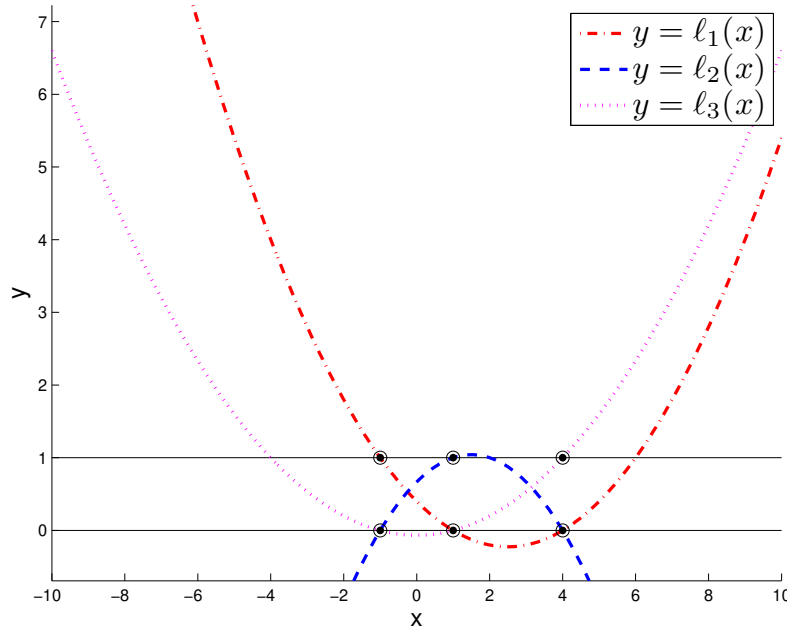


Figura 2.4: Gráficos dos polinômios de Lagrange do Exemplo 2.5.3. Os pontos destacados são aqueles em que os polinômios valem zero ou um, de acordo com a Definição 2.4.1.

Num contexto algorítmico onde os polinômios de Lagrange tenham sido calculados e a função objetivo tenha sido avaliada no conjunto de interpolação, a recuperação do modelo é imediatamente realizada através de (2.5.5). Portanto, retomando a função $f(x) = e^x$ e os polinômios de Lagrange deste exemplo, o modelo interpolador pode ser encontrado sem a necessidade de resolver o sistema linear (2.3.7):

$$\begin{aligned} m(x) &= f(-1)\ell_1(x) + f(1)\ell_2(x) + f(4)\ell_3(x) \\ &= e^{-1} \left(\frac{2}{5} - \frac{1}{2}x + \frac{1}{10}x^2 \right) + e \left(\frac{2}{3} + \frac{1}{2}x - \frac{1}{6}x^2 \right) + e^4 \left(\frac{-1}{15} + \frac{1}{15}x^2 \right) \\ &\approx -1.7 + 1.2x + 3.2x^2, \end{aligned}$$

o qual condiz com nossos cálculos anteriores (cf. (2.3.9)). \diamond

Exemplo 2.5.4. Analogamente ao caso unidimensional do exemplo anterior, os polinômios de Lagrange associados ao conjunto $\mathcal{Y} \subset \mathbb{R}^2$ do Exemplo 2.5.4 podem ser calculados a partir de sua matriz $M(\bar{\Phi}, \mathcal{Y})$, resultando em:

$$\begin{aligned} \ell_1(x, y) &= -x^2 - xy - y^2 + 1, & \ell_4(x, y) &= \frac{1}{2}x^2 - \frac{1}{2}x, \\ \ell_2(x, y) &= x^2 + xy, & \ell_5(x, y) &= \frac{1}{2}x^2 + xy + \frac{1}{2}y^2 - \frac{1}{2}x + \frac{1}{2}y, \\ \ell_3(x, y) &= -\frac{1}{2}x^2 + \frac{1}{2}y^2 + \frac{1}{2}x + \frac{1}{2}y, & \ell_6(x, y) &= -\frac{1}{2}x^2 - xy + \frac{1}{2}x. \end{aligned} \quad (2.5.6)$$

Com o aumento da dimensão de \mathcal{Y} , os gráficos dos polinômios de Lagrange tornam-se superfícies no \mathbb{R}^3 . A Figura 2.5 ilustra essas superfícies.

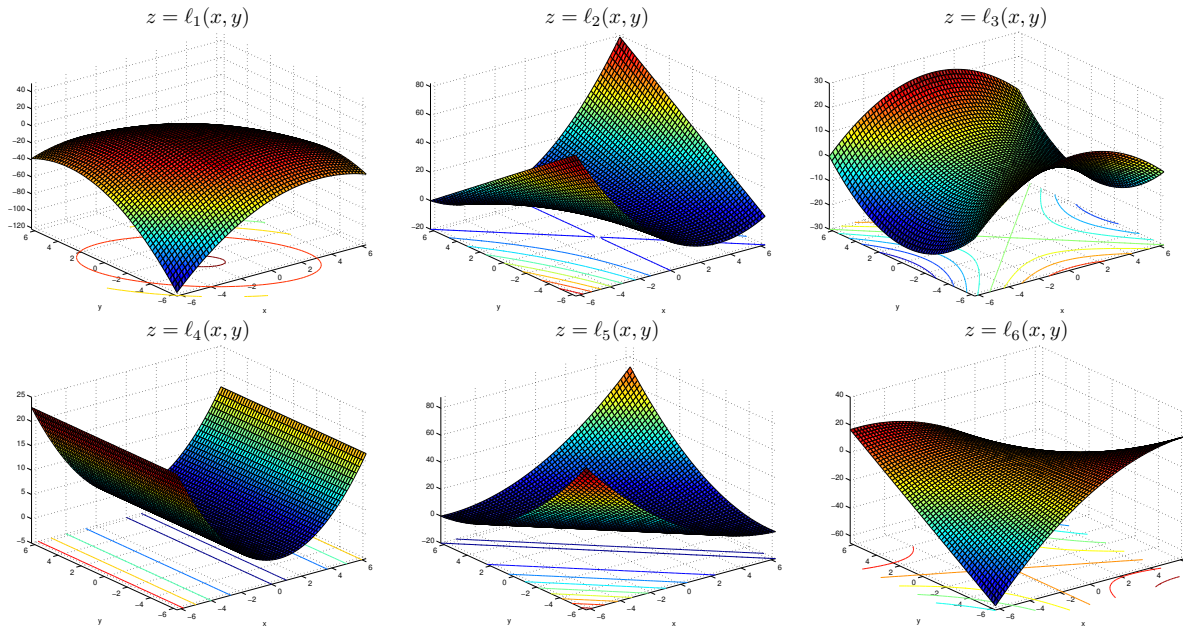


Figura 2.5: Gráficos dos polinômios de Lagrange associados ao conjunto \mathcal{Y} do Exemplo 2.5.4.

Apesar do aumento da dimensão dificultar a visualização gráfica, ainda é possível perceber que os polinômios $\ell_1, \ell_2, \dots, \ell_6$ se “coordenam” para valerem 1 no ponto do plano ao qual correspondem

e se anulam nos demais pontos de \mathcal{Y} . A Figura 2.6 ilustra simultaneamente os gráficos de tais polinômios.

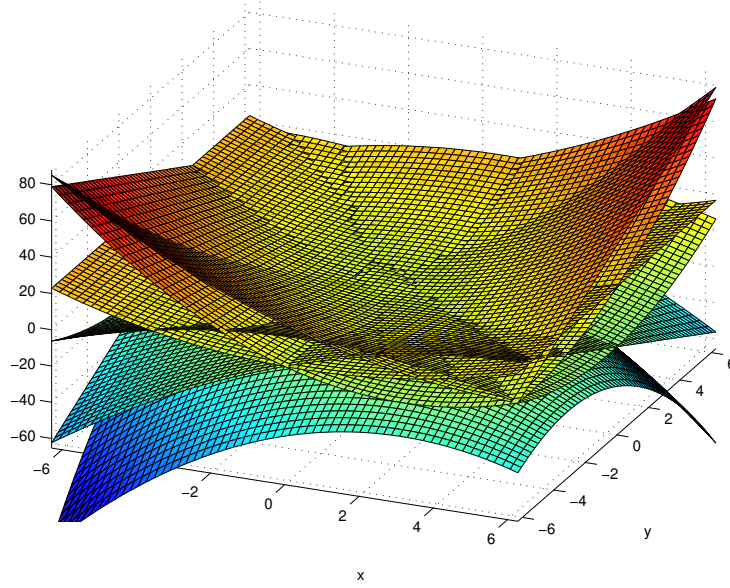


Figura 2.6: Sobreposição dos gráficos dos polinômios de Lagrange associados ao conjunto \mathcal{Y} do Exemplo 2.5.4.

2.6 Λ -posicionamento

Uma medida clássica de bom posicionamento de \mathcal{Y} em uma região $\mathcal{B} \in \mathbb{R}^n$ está relacionada com os valores absolutos dos elementos da base de polinômios de Lagrange nessa região \mathcal{B} . Adaptando o resultado de Ciarlet e Raviart [9, Teorema 2] ao caso da interpolação polinomial quadrática, obtém-se que, para qualquer x na envoltória convexa de \mathcal{Y} , vale a desigualdade

$$\|\mathcal{D}^r f(x) - \mathcal{D}^r m(x)\| \leq \frac{\nu}{6} \sum_{i=1}^q \|y^i - x\|^3 \|\mathcal{D}^r \ell_i(x)\|, \quad (2.6.1)$$

onde \mathcal{D}^r denota o operador derivada de ordem $r \in \{0, 1, 2\}$, $\|\cdot\|$ representa a norma Euclidiana e ν é dado por

$$\nu = \sup_{x \in \text{conv}(\mathcal{Y})} \|\mathcal{D}^3 f(x)\| < \infty, \quad (2.6.2)$$

ou, conseqüentemente, qualquer outro limitante superior para $\|\mathcal{D}^3 f(x)\|$.

Para $r = 0$, a Desigualdade (2.6.1) se converte numa estimativa para o limitante do erro que se comete ao aproximar a função objetivo f pelo modelo interpolatório quadrático m :

$$|f(x) - m(x)| \leq \frac{\nu}{6} \sum_{i=1}^q \|y^i - x\|^3 |\ell_i(x)|, \quad \forall x \in \text{conv}(\mathcal{Y}). \quad (2.6.3)$$

Um resultado bastante semelhante é demonstrado em Powell [40, pp. 299], onde x é tomado como qualquer vetor no \mathbb{R}^n . Nele, o autor argumenta que, mesmo quando f não possui derivadas de terceira ordem, experimentos numéricos mostram que é bastante útil assumir a Desigualdade (2.6.3) como válida para algum número real ν .

A desigualdade anterior pode ser ainda mais simplificada se um limitante superior para o valor absoluto dos polinômios de Lagrange em uma região que contenha a de interesse for estabelecido. Com essa finalidade, seja

$$\Lambda_{\mathcal{Y}} = \max_{i=1,2,\dots,q} \max_{x \in \mathcal{B}(\mathcal{Y})} |\ell_i(x)|, \quad (2.6.4)$$

onde $\mathcal{B}(\mathcal{Y})$ é a bola de menor raio na norma Euclidiana que contém o conjunto \mathcal{Y} . Observe que, como $\text{conv}(\mathcal{Y}) \subset \mathcal{B}(\mathcal{Y})$, então

$$\max_{x \in \mathcal{B}(\mathcal{Y})} |\ell_i(x)| \geq \max_{x \in \text{conv}(\mathcal{Y})} |\ell_i(x)|$$

para todo $i = 1, 2, \dots, q$. Assim, sendo Δ o diâmetro de $\mathcal{B}(\mathcal{Y})$, a Desigualdade (2.6.3) é simplificada em

$$|f(x) - m(x)| \leq \frac{q\nu\Lambda_{\mathcal{Y}}\Delta^3}{6}, \quad \forall x \in \text{conv}(\mathcal{Y}). \quad (2.6.5)$$

Note-se que, na Desigualdade (2.6.5), a constante $\Lambda_{\mathcal{Y}}$, junto com Δ , desempenham um papel muito importante na limitação do erro cometido ao se aproximar a função objetivo, principalmente porque q e ν são constantes intrínsecas ao problema. Num contexto algorítmico de região de confiança, o controle do raio Δ da região é feito diretamente a cada iteração. Por outro lado, o controle direto da constante $\Lambda_{\mathcal{Y}}$ requer certo cuidado especial. Isso motiva a seguinte definição.

Definição 2.6.1. Sejam $\Lambda > 0$, um conjunto $\mathcal{B} \subset \mathbb{R}^n$ e uma base $\Phi = \{\phi_1(x), \phi_2(x), \dots, \phi_q(x)\}$ de \mathcal{P}_n^2 . Um conjunto posicionado $\mathcal{Y} = \{y^1, y^2, \dots, y^q\}$ é dito **Λ -posicionado** em \mathcal{B} , no sentido da interpolação, se e somente se

1. para a base de polinômios de Lagrange associada a \mathcal{Y} vale

$$\Lambda \geq \max_{i=1,2,\dots,q} \max_{x \in \mathcal{B}} |\ell_i(x)|, \quad (2.6.6)$$

ou, equivalentemente,

2. para todo e qualquer $x \in \mathcal{B}$ existe $\lambda(x) \in \mathbb{R}^q$ tal que

$$\sum_{i=1}^q \lambda_i(x) \Phi(y^i) = \Phi(x) \quad \text{com} \quad \|\lambda(x)\|_{\infty} \leq \Lambda,$$

ou, equivalentemente,

3. a substituição de qualquer ponto de \mathcal{Y} por qualquer ponto $x \in \mathcal{B}$ pode aumentar o volume do conjunto $\{\Phi(y^1), \Phi(y^2), \dots, \Phi(y^q)\}$ no máximo por um fator Λ .

O ato ou efeito de ser Λ -posicionado será chamado de **Λ -posicionamento**.

A constante Λ está fortemente relacionada à definição da constante de Lebesgue na teoria da aproximação. Para saber mais, consulte [46] e [51]. Por concisão, em se tratando de Λ -posicionamento, serão omitidas as referências à base e ao espaço \mathcal{P}_n^2 adotados, pois estas deverão estar claras pelo contexto em que serão mencionadas.

Atente para o fato de que, na Definição 2.6.1, o conjunto \mathcal{Y} não necessariamente é um subconjunto da região \mathcal{B} onde o valor absoluto dos polinômios de Lagrange estão sendo maximizados. De fato, teoricamente, desde que se garanta que os polinômios de Lagrange sejam limitados em valor absoluto, \mathcal{Y} pode estar arbitrariamente distante de \mathcal{B} .

Das duas últimas versões da Definição 2.6.1 de Λ -posicionamento, é possível notar alguns outros aspectos geométricos relacionados a essa medida de posicionamento. Por exemplo, observa-se que a constante Λ é uma medida de quão bem $\Phi(\mathcal{Y})$ gera $\Phi(\mathcal{B})$. Ainda, se $\Phi(\mathcal{Y})$ gera $\Phi(\mathcal{B})$ bem, então não é possível aumentar substancialmente o volume da envoltória convexa de $\Phi(\mathcal{Y})$ através da troca de algum ponto de \mathcal{Y} por qualquer ponto de \mathcal{B} .

O lema a seguir, proposto por Conn *et. al.* [14, pp. 43], fornece alguns outros aspectos geométricos complementares.

Lema 2.6.2. Dado um conjunto amostral \mathcal{Y} , seja $\mathcal{B}(y^1, \Delta(\mathcal{Y}))$ a menor bola fechada centrada em $y^1 \in \mathcal{Y}$ e que contém \mathcal{Y} . Sem perda de generalidade, suponha que a base adotada seja a base $\bar{\Phi}$ natural, $y^1 = 0 \in \mathbb{R}^n$ e que o raio $\Delta(\mathcal{Y})$ seja igual a 1. Admita também que, para dado $\Lambda > 0$, \mathcal{Y} seja não posicionado em $\mathcal{B}(0, 1)$. Então, existem $z \in \mathcal{B}(0, 1)$ e $\gamma(z) \in \mathbb{R}^q$ tais que

$$\max_{x \in \mathcal{B}(0, 1)} \|\bar{\Phi}(x)\|_\infty \leq 1 \quad \text{e} \quad \left\| \sum_{i=1}^q \gamma_i(z) \bar{\Phi}(y^i) \right\|_\infty \leq \frac{1}{\Lambda} \quad \text{com} \quad \gamma_1(z) > 1. \quad (2.6.7)$$

Em outras palavras, de acordo com o Lema 2.6.2, o valor $1/\Lambda$, em certo sentido, limita a distância dos vetores $\bar{\Phi}(y^i)$, $i = 1, \dots, q$, à dependência linear. Conforme Λ aumenta, mais próximo de 0 fica o lado direito da Desigualdade (2.6.7). No limite, quando $1/\Lambda$ se aproxima de zero, o sistema representado por esses vetores se torna linearmente dependente. Nesse sentido, a constante Λ pode ser interpretada como uma medida da distância a um conjunto não posicionado.

2.7 Propriedades do Λ -posicionamento

Alguns resultados básicos sobre o conceito de Λ -posicionamento são apresentados a seguir, eventualmente com alguns comentários a respeito de suas demonstrações. Para maiores detalhes sobre os resultados e, em alguns casos, para as demonstrações, veja Conn *et. al.* [14, Seção 3.3].

1. Se $\mathcal{Y} \subset \mathbb{R}^n$ é Λ -posicionado em $\mathcal{B} \subset \mathbb{R}^n$ e se algum dos pontos de \mathcal{Y} pertence à região \mathcal{B} , então $\Lambda \geq 1$.

Da Definição 2.4.1 sabe-se que, a cada ponto $y^i \in \mathcal{Y}$ corresponde um polinômio $\ell_i(x) \in \Phi_\ell$, o qual satisfaz $\ell_i(y^i) = 1$. Se pelo menos um desses pontos (y^k , por exemplo) pertence à região \mathcal{B} , então $\Lambda \geq \max_{x \in \mathcal{B}} |\ell_k(x)| \geq 1$ e, portanto, vale a propriedade.

2. Se \mathcal{Y} é Λ -posicionado em um dado conjunto \mathcal{B} , então \mathcal{Y} é Λ -posicionado em qualquer subconjunto de \mathcal{B} .

Claramente, o máximo sobre um conjunto \mathcal{B} é maior do que o máximo sobre qualquer um dos subconjuntos de \mathcal{B} .

3. Se \mathcal{Y} é Λ -posicionado em \mathcal{B} e $\bar{\Lambda}$ é uma constante real que satisfaz $\bar{\Lambda} \geq \Lambda$, então \mathcal{Y} é $\bar{\Lambda}$ -posicionado em \mathcal{B} .

Essa propriedade é uma aplicação direta da desigualdade $\bar{\Lambda} \geq \Lambda$ na Desigualdade (2.6.6).

4. Se $\mathcal{Y} = \{y^1, y^2, \dots, y^q\} \subset \mathbb{R}^n$ é Λ -posicionado em $\mathcal{B}(\bar{x}, \Delta)$, então, para qualquer ξ real não nulo e qualquer $s \in \mathbb{R}^n$, $\hat{\mathcal{Y}} = \xi(\mathcal{Y} + s) = \{\xi(y^1 + s), \xi(y^2 + s), \dots, \xi(y^q + s)\}$ é Λ -posicionado em $\mathcal{B}(\xi(\bar{x} + s), \xi\Delta)$.

Essa última propriedade diz que o Λ -posicionamento é invariante com respeito a translações e escalamentos dos pontos de \mathcal{Y} .

Exemplo 2.7.1. O gráfico dos polinômios de Lagrange do conjunto de pontos do Exemplo 2.5.3 foi exibido na Figura 2.4. Agora, na Figura 2.7, são exibidos os valores absolutos de tais polinômios.

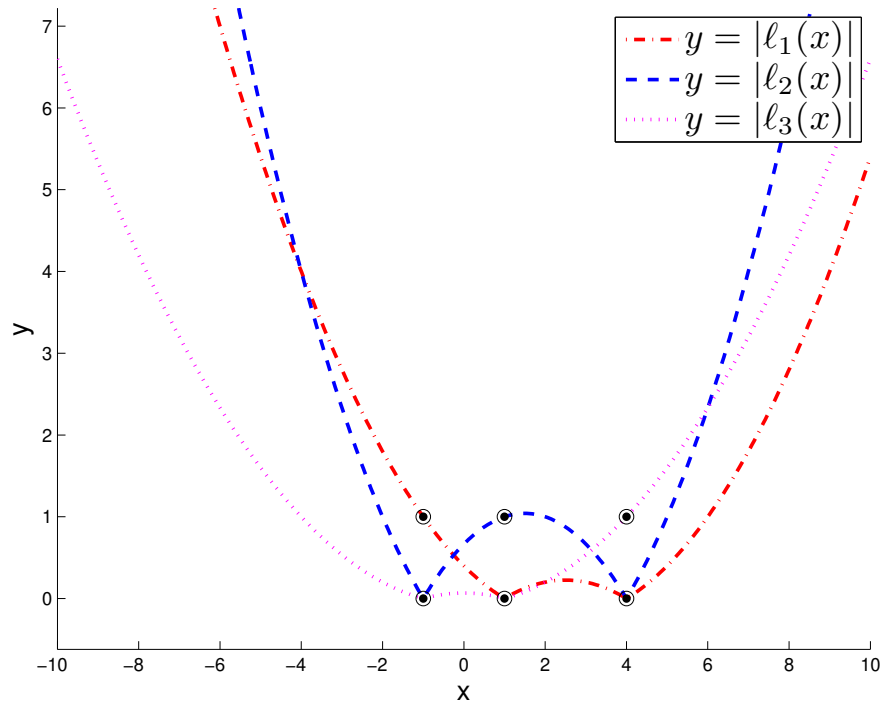


Figura 2.7: Gráficos do valor absoluto dos polinômios de Lagrange do Exemplo 2.7.1, com destaque para três níveis de posicionamento.

Novamente, são ilustrados os valores absolutos dos polinômios nos pontos de $\mathcal{Y} = \{-1, 1, 4\}$. A ideia aqui é encontrar intervalos em \mathbb{R} (eixo x) nos quais os valores absolutos dos três polinômios

possam ser limitados por uma constante Λ . De acordo com (2.6.6), nesses intervalos, o conjunto \mathcal{Y} será dito Λ -posicionado.

Observando a reta horizontal $y = 3$ na Figura 2.7 é possível concluir, por exemplo, que \mathcal{Y} é 3-posicionado no intervalo $[-2, 6]$. Isto porque, nesse intervalo, o valor absoluto de cada um dos polinômios não ultrapassa a magnitude 3. De fato, o intervalo de maior comprimento onde \mathcal{Y} é 3-posicionado é ainda mais abrangente do que este. Ele vai desde o ponto $a < 0$ tal que $|\ell_1(a)| = 3$ até o ponto $b > 0$ tal que $|\ell_2(b)| = 3$. Ainda, visualmente, \mathcal{Y} continuará sendo 3-posicionado em qualquer subconjunto do intervalo $[a, b]$ e que \mathcal{Y} é também $\bar{\Lambda}$ -posicionado em $[a, b]$ para qualquer $\bar{\Lambda} \geq 3$, como por exemplo $\bar{\Lambda} = 6$. Tais constatações estão de acordo com as Propriedades 2 e 3, respectivamente.

Deste caso simples, é possível entender como um conjunto \mathcal{Y} pode ser Λ -posicionado com uma constante $\Lambda < 1$: basta ver que existe um intervalo imediatamente à esquerda da origem onde o módulo dos polinômios pode ser limitado por um valor menor do que um, tal como 0.8. Mais do que isso: existem infinitos intervalos com tais características. Outro deles pode ser encontrado entre 2 e 4, por exemplo. \diamond

Exemplo 2.7.2. Na Figura 2.8 encontram-se os gráficos do valor absoluto de cada polinômio de Lagrange do Exemplo 2.5.4. A sobreposição desses gráficos é ilustrada pela Figura 2.9.

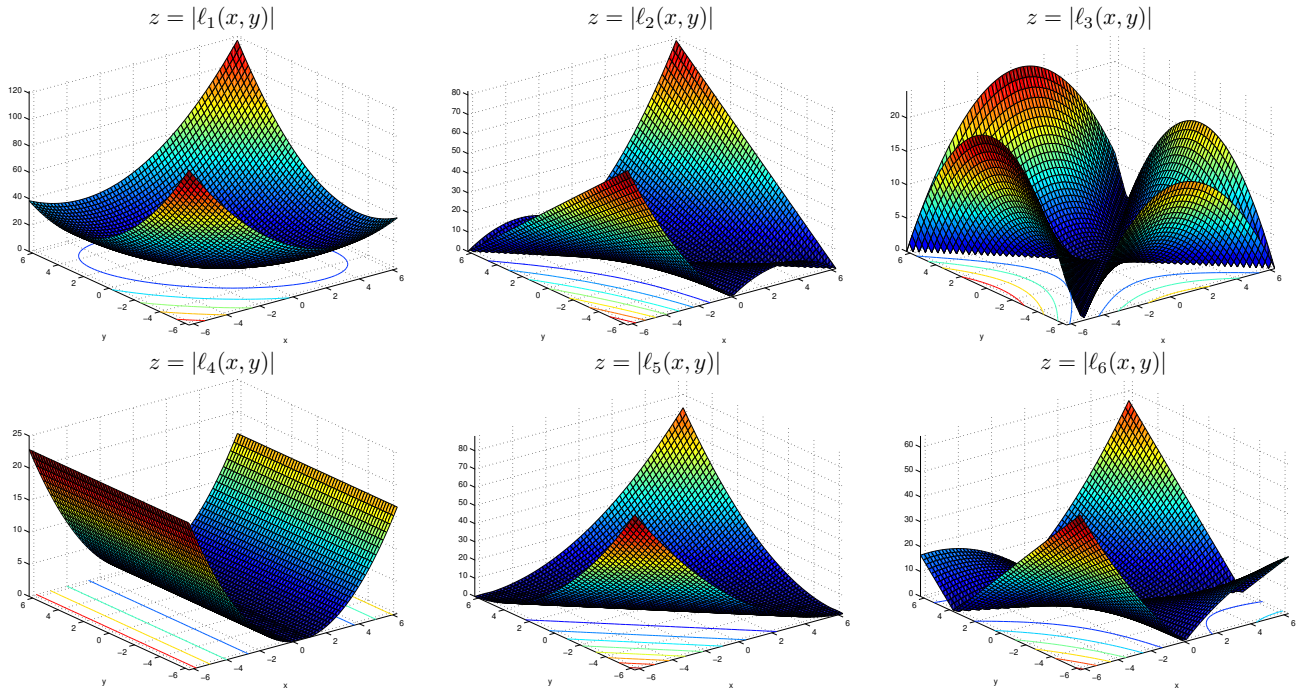


Figura 2.8: Gráficos das funções valor absoluto dos polinômios de Lagrange associados ao conjunto \mathcal{Y} do Exemplo 2.5.4.

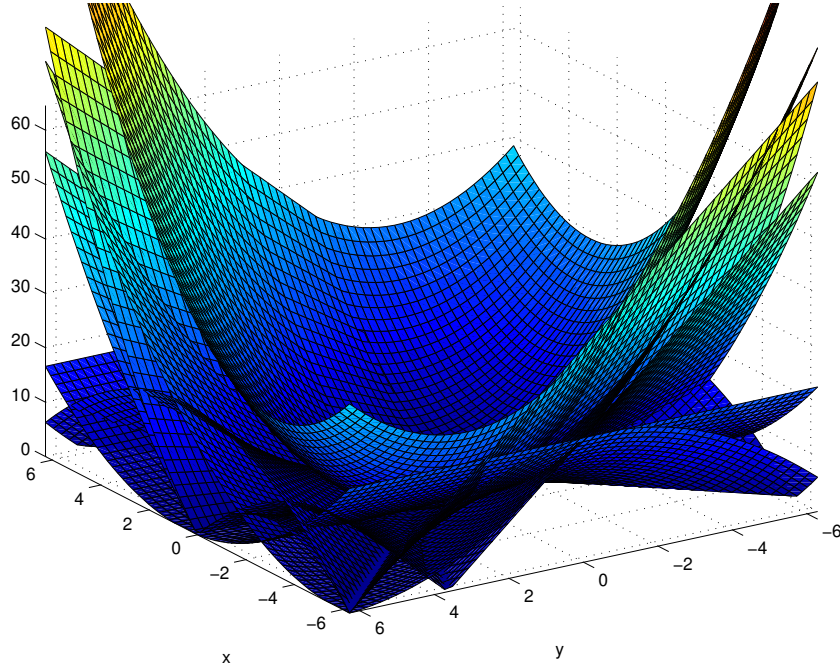


Figura 2.9: Sobreposição dos gráficos das funções valor absoluto dos polinômios de Lagrange associados ao conjunto \mathcal{V} do Exemplo 2.5.4.

O equivalente a procurar por regiões onde \mathcal{V} seja Λ -posicionado, como feito no exemplo anterior, seria traçar planos paralelos ao plano xy na altura $z = \Lambda$ e analisar graficamente em qual região \mathcal{B} do plano xy o valor do módulo dos polinômios de Lagrange pode ser limitado por Λ . Entretanto, a fim de facilitar a visualização dos gráficos, optou-se por traçar as curvas de nível $|\ell_i(x)| = \Lambda$ e, através delas, demarcar a maior região do plano onde \mathcal{V} é Λ -posicionado. A Figura 2.10 ilustra três casos particulares: $\Lambda = 1$, $\Lambda = 2$ e $\Lambda = 50$. \diamond

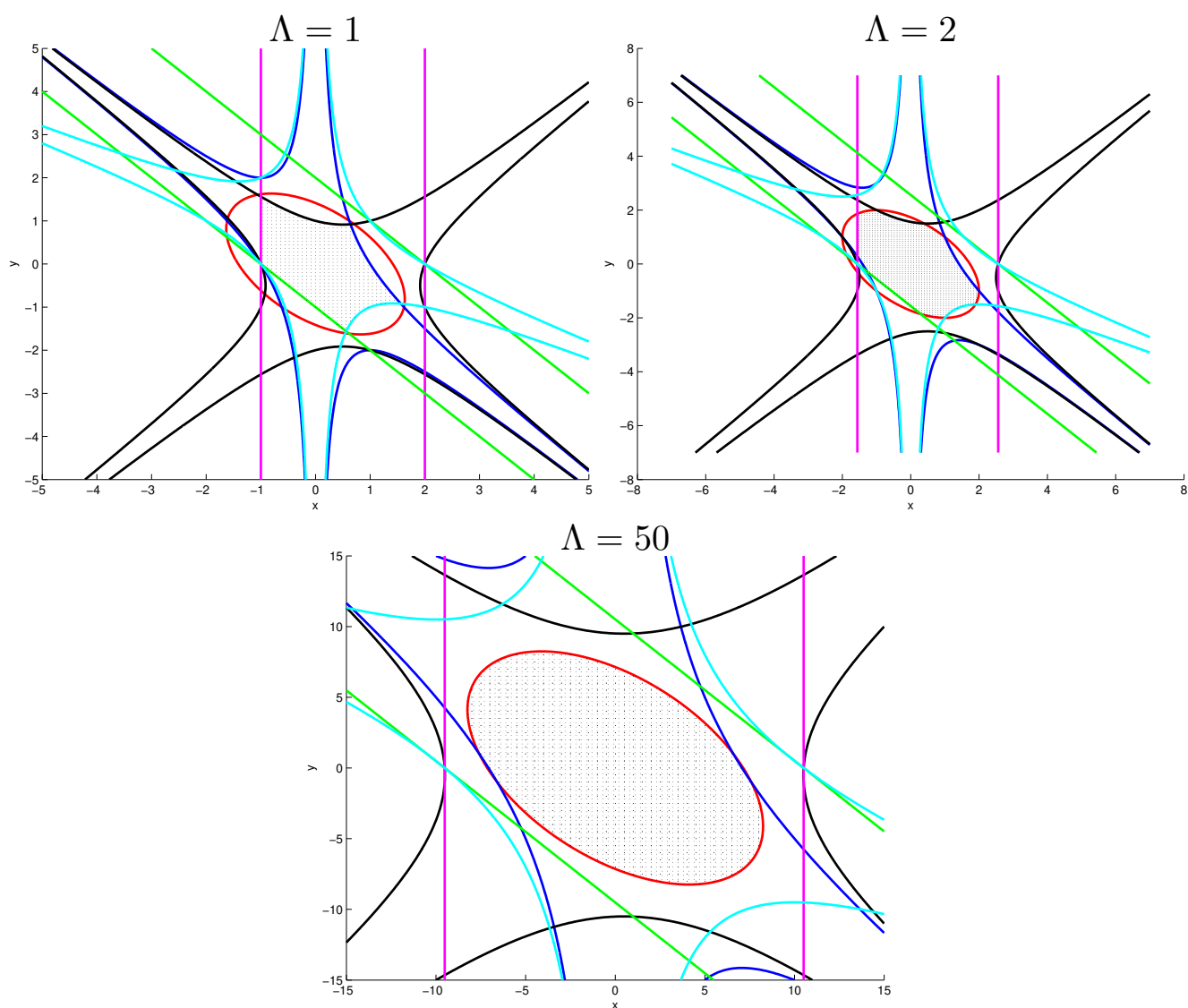


Figura 2.10: Curvas de nível Λ dos polinômios de Lagrange associados ao conjunto \mathcal{V} do Exemplo 2.5.4. Em cada um dos casos, a parte hachurada indica a região de maior área na qual \mathcal{V} é Λ -posicionado.

Esquema algorítmico de um método de região de confiança

Seguindo rumo ao objetivo de introduzir o algoritmo central deste trabalho, proposto originalmente por Scheinberg e Toint [47], agora com o estabelecimento dos modelos polinomiais a serem adotados e com a exploração dos aspectos geométricos que permeiam a sua utilização como aproximadores para a função objetivo, este capítulo se propõe a contextualizar o uso desses modelos num cenário de região de confiança.

Retomando o que foi dito na introdução do capítulo anterior, a aproximação local da função objetivo f por um modelo interpolador polinomial quadrático m está inserida no contexto algorítmico dos chamados **métodos de região de confiança**[11]. Tais métodos são iterativos¹ e, a cada iteração k , constroem em torno do iterando x^k um modelo $m_k(x^k + s)$ que aproxima a função objetivo *suficientemente bem* em uma região $\mathcal{B}(x^k, \Delta_k)$, chamada de região de confiança de raio Δ_k . Um passo s^k é determinado através da resolução de um subproblema de região de confiança dado por

$$\underset{s \in \mathcal{B}(0, \Delta_k)}{\text{minimizar}} \ m_k(x^k + s). \quad (3.0.1)$$

Com isto, o ponto tentativo da iteração é definido como $x_+^k = x^k + s^k$ e o valor $f(x_+^k)$ é calculado. Em seguida, a redução no valor de f efetivamente alcançada ($ared_k = f(x^k) - f(x^k + s^k)$) é comparada com a redução prevista pelo modelo ($pred_k = m_k(x^k) - m_k(x^k + s^k)$). Se a razão $\rho_k = ared_k / pred_k$ é grande o bastante, isto é, se $\rho_k \geq \eta \in (0, 1)$, então x_+^k é aceito como novo iterando, o modelo é atualizado e o raio da região de confiança é (possivelmente) aumentado. Iterações como essa são chamadas de iterações de sucesso. Por outro lado, se $\rho < \eta$, o ponto tentativo é rejeitado como iterando (ainda que possa ser aceito como ponto amostral de \mathcal{Y}_{k+1}), a região de confiança é reduzida e a iteração é dita ser de insucesso.

Essa estrutura geral está permeada de detalhes que, crucialmente, influenciam na análise de convergência e no desempenho desses métodos. Por exemplo, apesar de tipicamente se realizar

¹ Por conta das iterações, os superíndices deste capítulo indicam a iteração a qual se refere, assim como ocorre nas Seções 4.8 e 4.9. Assim, os índices que antes ocupavam essas posições passam a ser subíndices. Nesses casos, por exemplo, $\ell_j^k(x)$, y_j^k e x_+^k correspondem, respectivamente, ao j -ésimo polinômio de Lagrange e aos vetores y_j e x_+ , todos da k -ésima iteração.

a otimização do modelo m na região (de confiança) de interesse, a análise de convergência exige apenas que a seguinte condição, chamada de Condição de decréscimo de Cauchy, seja satisfeita:

$$m_k(x^k) - m_k(x_+^k) \geq \kappa_C \|g^k\| \min \left\{ \frac{\|g^k\|}{1 + \|H^k\|}, \Delta_k \right\}, \quad (3.0.2)$$

onde $g^k \stackrel{\text{def}}{=} \nabla m_k(x^k)$ e $H^k \stackrel{\text{def}}{=} \nabla^2 m_k(x^k)$. Outro ponto não esclarecido acima é o significado do termo “aproxime suficientemente bem” usado para se referir à maneira como o modelo aproxima a função objetivo. Na prática, alguma definição quantitativa precisa ser adotada na implementação do método.

Como destacam Bandeira *et. al.* [5, Seção 5], as propriedades da convergência global para esses métodos dependem fortemente da exigência de que o modelo se torne mais acurado conforme o raio da região de confiança diminui. Em particular, isso implica que o raio da região de confiança deva estar suficientemente longe de zero, desde que não se chegue a um ponto estacionário. Modelos baseados em aproximações de Taylor com o uso de derivadas satisfazem esse requerimento naturalmente.

Na ausência de derivadas, no entanto, cuidados especiais devem ser tomados com os modelos e pontos amostrais. Conn *et. al.* [13] demonstram a convergência global a pontos estacionários de primeira e segunda ordem desde que os modelos polinomiais (lineares ou quadráticos) satisfaçam limitantes do tipo dos existentes para modelos de Taylor em otimização com derivada. Para manter o posicionamento dos pontos bom o suficiente para a convergência, passos especiais de melhora na geometria do conjunto são geralmente incluídos.

Nesse cenário, surge o trabalho de Fasano, Morales e Nocedal [21], no qual os autores abrem mão dessas iterações especiais e do cuidado com a geometria dos pontos, e surpreendentemente obtêm resultados de robustez e eficiência competitivos, mesmo quando comparados ao estado-da-arte de otimização sem derivada irrestrita, o algoritmo NEWUOA, de Powell [42, 43]. Esses autores levantam, então, a hipótese de existência de um mecanismo de auto-correção no método testado.

Inspirados por esses resultados, Scheinberg e Toint [47] mostram que, de fato, existe um mecanismo de geometria autocorretiva por trás dessa classe de algoritmos. Entretanto, a conclusão é de que, sem o controle efetivo do posicionamento dos pontos, não é possível ter garantias de convergência para o método partindo de pontos iniciais arbitrários. Ainda nesse trabalho, um novo algoritmo é desenvolvido visando reduzir substancialmente a necessidade dos passos geométricos, isto é, passos em que se controla a geometria dos pontos, através da exploração da propriedade de auto-correção. E, assim, surge o algoritmo central desta dissertação, originalmente proposto em [47, Algoritmo 2] como uma tentativa teoricamente exitosa de abrir mão das iterações especiais de controle da geometria dos pontos amostrais sem perder a garantia de convergência global a pontos estacionários de primeira ordem. No presente trabalho, optou-se pelo nome **DFO-LP** para se referir a esse algoritmo, destacando a adoção dos polinômios de Lagrange (*Lagrange polynomials*, em inglês) como ferramenta principal de controle da geometria dos pontos.

Sobre o novo algoritmo, Scheinberg e Toint argumentam que, quando o gradiente do modelo é muito pequeno, não se deve dar prosseguimento às iterações sem antes garantir algum posicionamento para o conjunto amostral de pontos. Nessas situações, faz-se necessário um **teste de criticalidade**, que tem como atribuição verificar se a pequena norma do gradiente do modelo é

indicativo de estacionaridade ou de baixa qualidade dos pontos de interpolação. Assim, se recorre ao passo geométrico somente quando o gradiente do modelo é *suficientemente* pequeno. Entretanto, tal liberdade deve vir acompanhada do mecanismo de atualização dos pontos que leva em consideração sua geometria [47, pp. 3513], conforme a proposta **DFO-LP**.

A seguir, o **DFO-LP** é apresentado ainda em sua forma teórica, seguido dos principais aspectos da demonstração de sua convergência global a pontos estacionários de primeira ordem. Dentro dos propósitos desta dissertação, nos capítulos seguintes será descrita uma implementação computacional para esse algoritmo, chamada de **DFO-LP**, e um estudo numérico de seu desempenho.

3.1 DFO-LP

Passo 0: Inicialização. Um raio de região de confiança inicial Δ_0 e uma constante inicial de acurácia ϵ_0 são dados. Um conjunto amostral inicial posicionado para interpolação \mathcal{Y}_0 é conhecido e contém um ponto inicial x_0 . Esse conjunto de interpolação define um modelo interpolador inicial m_0 em torno de x_0 e os polinômios de Lagrange $\{\ell_j^0\}_{j=1}^q$ associados. Constantes $\eta \in (0, 1)$, $0 < \kappa_C < 1$, $0 < \gamma_1 \leq \gamma_2 < 1$, $\mu \in (0, 1)$, $\theta > 0$, $\beta \geq 1$ e $\Lambda > 1$ são também dadas. Escolha $v^0 \neq x^0$ e faça $k = 0$ e $i = 0$.

Passo 1: Teste de Criticalidade.

Passo 1a: Defina $\hat{m}_i = m_k$.

Passo 1b: Se $\|\nabla \hat{m}_i(x^k)\| < \epsilon_i$, faça $\epsilon_{i+1} = \mu \|\nabla \hat{m}_i(x^k)\|$, calcule um modelo² \hat{m}_{i+1} Λ -posicionado em $\mathcal{B}(x^k, \epsilon_{i+1})$, incremente i em uma unidade, e recomece o Passo 1b.

Passo 1c: Se um novo modelo foi calculado, faça $m_k = \hat{m}_i$, $\Delta_k = \theta \|\nabla m_k(x^k)\|$, e defina $v^i = x^k$.

Passo 2: Calcule um ponto tentativo. Calcule x_+^k de maneira que (3.0.2) valha e que

$$\|x_+^k - x^k\| \leq \Delta_k.$$

Passo 3: Avalie a função objetivo no ponto tentativo. Calcule $f(x_+^k)$ e ρ_k .

Passo 4: Defina o próximo iterando

Passo 4a: Iteração de sucesso. Se $\rho_k \geq \eta$, defina $x^{k+1} = x_+^k$, escolha $\Delta_{k+1} \geq \Delta_k$, e defina $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y_r^k\} \cup \{x_+^k\}$, para

$$y_r^k \in \arg \max_{y_j^k \in \mathcal{Y}_k} \|y_j^k - x_+^k\|^2 |\ell_j^k(x_+^k)|. \quad (3.1.1)$$

Passo 4b: Substitua um ponto de interpolação distante. Se $\rho_k < \eta$, $x^k \neq v^i$ ou $\Delta_k \leq \epsilon_i$, e o conjunto

$$\mathcal{F}_k := \{y_j^k \in \mathcal{Y}_k ; \|y_j^k - x^k\| > \beta \Delta_k \text{ e } \ell_j^k(x_+^k) \neq 0\} \quad (3.1.2)$$

² A rigor, o que se pede é a construção de um modelo \hat{m}_{i+1} que resulte de um conjunto de pontos, este sim, Λ -posicionado em $\mathcal{B}(x^k, \epsilon_{i+1})$. No entanto, preferiu-se manter a expressão da versão original do algoritmo (ver [47]).

é não vazio, então faça $x^{k+1} = x^k$, $\Delta_{k+1} = \Delta_k$, e defina $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y_r^k\} \cup \{x_+^k\}$, onde r é um índice de um ponto qualquer em \mathcal{F}_k , por exemplo, tal que

$$y_r^k \in \arg \max_{y_j^k \in \mathcal{F}_k} \|y_j^k - x_+^k\|^2 |\ell_j^k(x_+^k)|. \quad (3.1.3)$$

Passo 4c: Substitua um ponto de interpolação próximo. Se $\rho_k < \eta$, $x^k \neq v^i$ ou $\Delta_k \leq \epsilon_i$, o conjunto \mathcal{F}_k é vazio, e o conjunto

$$\mathcal{C}_k := \left\{ y_j^k \in \mathcal{Y}_k \setminus \{x^k\} ; \|y_j^k - x^k\| \leq \beta \Delta_k \text{ e } |\ell_j^k(x_+^k)| > \Lambda \right\} \quad (3.1.4)$$

é não vazio, então faça $x^{k+1} = x^k$, $\Delta_{k+1} = \Delta_k$, e defina $\mathcal{Y}_{k+1} = \mathcal{Y}_k \setminus \{y_r^k\} \cup \{x_+^k\}$, onde r é um índice de um ponto qualquer em \mathcal{C}_k , por exemplo, tal que

$$y_r^k \in \arg \max_{y_i^k \in \mathcal{C}_k} \|y_j^k - x_+^k\|^2 |\ell_j^k(x_+^k)|. \quad (3.1.5)$$

Passo 4d: Reduza o raio de região de confiança. Se $\rho_k < \eta$ e, ou $x^k = v^i$ e $\Delta_k > \epsilon_i$, ou $\mathcal{F}_k \cup \mathcal{C}_k = \emptyset$, então faça $x^{k+1} = x^k$, $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$ e defina $\mathcal{Y}_{k+1} = \mathcal{Y}_k$.

Passo 5: Atualize o modelo e os polinômios de Lagrange. Se $\mathcal{Y}_{k+1} \neq \mathcal{Y}_k$, calcule o modelo interpolador m_{k+1} em torno de x^{k+1} usando \mathcal{Y}_{k+1} e os polinômios de Lagrange $\{\ell_j^{k+1}\}_{j=1}^q$ associados. Incremente k em uma unidade, e volte para o Passo 1.

Comentários sobre o algoritmo

O algoritmo **DFO-LP**, da maneira como foi apresentado em 3.1, está em sua forma teórica, uma vez que não foram atribuídos valores às constantes envolvidas em sua execução nem tampouco foi definido um critério de parada, dentre outras especificações de caráter prático que certamente influenciam em seu desempenho. Tais esclarecimentos serão feitos no Capítulo 4 e na Seção 7.1.1, antes da apresentação dos resultados computacionais. Algumas ideias gerais sobre o algoritmo são retomadas com mais detalhes a seguir.

O Passo 1 foi elaborado para funcionar como um teste para a criticalidade do ponto corrente. Conforme assinalam os autores do **DFO-LP**, a criticalidade é confirmada, na prática, quando ϵ_i se torna menor do que certa tolerância definida pelo usuário.

Após o cálculo do valor da função objetivo no ponto tentativo obtido através da resolução do subproblema de região de confiança, a iteração é classificada no Passo 4 como *sucesso* ou *insucesso*. O ponto tentativo será aceito como iterando seguinte somente em caso de sucesso na iteração atual (Passo 4a). Entretanto, ainda que o ponto tentativo não tenha resultado em uma diminuição *satisfatória* do valor da função objetivo, é possível que este seja incluído no conjunto amostral, o que necessariamente implica a substituição de algum ponto desse conjunto, já que sua cardinalidade deve ser mantida.

O ponto tentativo x_+^k será inserido no conjunto amostral apenas se as condições estabelecidas pelos Passos 4b e 4c forem satisfeitas. Essencialmente, no Passo 4b, verifica-se a existência de pontos que estejam suficientemente distantes da região de confiança e cuja substituição (por x_+^k) mantenha o conjunto posicionado. No algoritmo, o conjunto de tais pontos é denominado \mathcal{F}_k . Em seguida, no Passo 4c, caso as condições de 4b não sejam satisfeitas, o método tenta identificar pontos que estejam *suficientemente* próximos ao ponto corrente e cuja substituição (por x_+^k) apresente melhora *relevante* no posicionamento do conjunto resultante. Tais pontos constituem o conjunto \mathcal{C}_k . Somente então, se as condições de 4b e 4c não forem satisfeitas, isto é, se o Λ -posicionamento for garantido pelo teste de criticalidade ou se $\mathcal{F}_k \cup \mathcal{C}_k$ for vazio, o ponto tentativo será finalmente descartado e o raio da região de confiança será diminuído, no Passo 4d.

Observe que, da maneira como as etapas foram definidas, apenas uma das condições estipuladas pelos quatro itens do Passo 4 pode ocorrer. Destaque-se, também, o uso do polinômio de Lagrange associado ao ponto candidato a ser substituído, avaliado em x_+^k , como ferramenta de análise do efeito da troca de pontos no posicionamento do conjunto resultante (ver (2.5.4)).

Em cada uma das situações de substituição de pontos expressas em 4a, 4b e 4c, a única exigência teórica sobre o novo conjunto de interpolação é a de que este seja mantido posicionado. No entanto, em havendo mais de um ponto apto a ser descartado, Scheinberg e Toint sugerem que sejam adotados critérios envolvendo distância e posicionamento (3.1.1), (3.1.3), (3.1.5), de maneira que o procedimento seja melhorado [47, p. 3521]. Conforme será visto na Seção 4.8, a versão implementada para este trabalho utiliza uma variante desses critérios.

A chave da teoria desenvolvida para o **DFO-LP** consiste em mostrar que as condições dos Passos 4b e 4c só podem ser satisfeitas finitas vezes antes que uma iteração de sucesso ocorra ou o modelo se torne bem posicionado. Para isso, os autores do método destacam que a troca de pontos nesses dois passos é impedida caso o iterando ainda não tenha saído do ponto v_i , em torno do qual um modelo bem posicionado é conhecido e a região de confiança é maior do que o raio ϵ_i de posicionamento do modelo. Observe, no entanto, que o Passo 4a permite que o posicionamento do conjunto amostral se deteriore durante iterações de sucesso. Todos esses aspectos são levados em consideração pelos autores na demonstração do Teorema 3.2.2.

3.2 Convergência global

Os detalhes da demonstração da convergência global do **DFO-LP** podem ser conferidos em [47, Seção 5]. A seguir, enunciam-se apenas os resultados mais notáveis que culminam no teorema final sobre a convergência. As hipóteses sob as quais a demonstração de convergência de [47] foi feita são:

- H1) A função objetivo é continuamente diferenciável em um conjunto aberto \mathcal{V} contendo todos os iterandos gerados pelo algoritmo, e seu gradiente ∇f é Lipschitz contínuo em \mathcal{V} com constante ν .
- H2) Existe uma constante κ_f tal que $f(x) \geq \kappa_f$ para todo x em \mathcal{V} .
- H3) Existe uma constante $\kappa_H \geq \nu$ tal que $1 + \|H^k\| \leq \kappa_H$ para todo e qualquer $k \geq 0$, onde H^k é a Hessiana do modelo m_k .

Os autores comentam que a Hipótese H3 pode ser excessivamente restritiva e que uma outra, um pouco mais realística que essa e potencial substituta, exige apenas que a Hessiana do modelo não cresça muito rapidamente ao longo das iterações. Entretanto, por simplicidade das demonstrações, assumem H3 como verdadeira e destacam que as informações sobre a segunda derivada não são essenciais para a análise de convergência de primeira ordem.

A principal propriedade do **DFO-LP** é estabelecida no seguinte lema.

Lema 3.2.1. Suponha que as hipóteses de H1 a H3 são válidas e que m_k é de grau maior que ou igual a um. Então, para qualquer constante $\Lambda > 1$, se a iteração k é de insucesso, $\mathcal{F}_k = \emptyset$ e

$$\Delta_k \leq \min \left\{ \frac{1}{\kappa_H}, \frac{(1 - \eta)\kappa_C}{2\kappa_{ef}(\beta + q)^2(q\Lambda + 1)} \right\} := \kappa_\Lambda \|g^k\|,$$

onde κ_{ef} é uma certa constante associada ao limitante do erro cometido ao se aproximar f por m^k , então

$$\mathcal{C}_k \neq \emptyset.$$

Do Lema 3.2.1, é possível extrair a seguinte interpretação: se o raio da região de confiança é pequeno o suficiente, essencialmente com relação ao inverso da norma da Hessiana do modelo, e se todos os pontos de interpolação significativos estão contidos na região de confiança, então todas as iterações de insucesso devem resultar em uma melhora na geometria do conjunto de interpolação. E é justamente por isso que se diz que a geometria dos pontos é autocorretiva nas iterações de insucesso desse tipo. Adicionalmente, o valor da melhora na geometria depende somente de Λ .

Em seguida, são demonstrados lemas mais típicos de métodos de região de confiança. O primeiro deles assegura que o limitante Δ_k para o tamanho do passo não pode diminuir arbitrariamente longe de um ponto crítico. Depois, demonstra-se que, em algum momento, o gradiente do modelo deve se tornar menor do que ϵ_0 . Quando isso acontece, o algoritmo é reiniciado com um modelo resultante de um conjunto amostral Λ -posicionado em uma bola suficientemente menor que a corrente.

Finalmente, demonstra-se o seguinte resultado de convergência global:

Teorema 3.2.2. Se as hipóteses de H1 a H3 são satisfeitas e se o modelo é de grau maior que ou igual a um para todo k suficientemente grande, então

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| = 0, \tag{3.2.1}$$

onde

$$\liminf_{k \rightarrow \infty} \|\nabla f(x^k)\| \stackrel{\text{def}}{=} \lim_{k \rightarrow \infty} \left(\inf_{j \geq k} \|\nabla f(x^j)\| \right).$$

Uma implementação para o DFO-LP

A fim de comparar o desempenho de uma instância do algoritmo **DFO-LP** nos moldes da metodologia a ser introduzida no Capítulo 5, os principais aspectos da implementação computacional são apresentados a seguir. Neste capítulo, caracteriza-se a versão **DFO-LP** a partir de decisões práticas a respeito de aspectos que o algoritmo original **DFO-LP**, por seu caráter teórico, deixa em aberto.

4.1 *Software* para a implementação computacional

O código **DFO-LP** e suas subrotinas foram implementados em **MATLAB**[®], um ambiente computacional desenvolvido pela empresa americana MathWorks[®][2] para cálculos numéricos e que provê, também, ferramentas de computação simbólica, geração de gráficos e interface com diferentes linguagens de programação. A capacidade de processamento do **MATLAB** vem evoluindo satisfatoriamente em suas versões mais recentes, sem perder a tradicional amigabilidade para com o usuário nem a estrutura familiar à escrita matemática. Principalmente para problemas do porte dos que são tratados aqui, tais características unidas à disponibilização institucional, resultaram na centralização dos testes nesse *software*. Diversos trabalhos importantes de otimização sem derivadas corroboram, através de seus experimentos computacionais, a razoabilidade da adoção do **MATLAB** [5, 15, 17, 21, 26, 32, 55].

4.2 Representação do conjunto amostral de pontos e polinômios de Lagrange

Polinômios de Lagrange desempenham o papel de indicar quando a troca de determinado ponto do conjunto amostral \mathcal{Y} preserva a não singularidade das equações de interpolação e ainda provêm um limitante para o erro do modelo quadrático. Dada sua importância (e custo computacional), se fazem necessárias maneiras eficientes de construí-los e atualizá-los, assim como ocorre com o conjunto amostral e o modelo aproximador. A representação desses objetos matemáticos através

de matrizes do **MATLAB** se dá com o armazenamento de seus coeficientes em bases apropriadas, seja para o \mathbb{R}^n ou para o \mathcal{P}_n^2 .

Já no Passo 0 de **DFO-LP**, subentende-se a existência de um conjunto \mathcal{Y}_0 e da base de polinômios de Lagrange associada, além do próprio modelo aproximador inicial. Em uma iteração típica do algoritmo **DFO-LP**, a atualização de \mathcal{Y} corresponde à simples troca de um de seus elementos, acompanhada da atualização de seus polinômios de Lagrange e, possivelmente, do modelo aproximador. Há ainda as iterações do Teste de Criticalidade, que podem resultar na troca de grande parte do conjunto de pontos, e que precisam estar implementadas de maneira cuidadosa e tão eficiente quanto possível.

Nesse sentido, considere o modelo polinomial quadrático $m \in \mathcal{P}_n^2$ do Capítulo 2, representado em sua forma matricial em torno de um ponto $\bar{x} \in \mathbb{R}^n$, como em (2.3.1):

$$m(x) = c + g^\top(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^\top H(x - \bar{x}), \quad (4.2.1)$$

onde $g \in \mathbb{R}^n$ e $H = H^\top \in \mathbb{R}^{n \times n}$. Fixado o ponto \bar{x} , o modelo m pode ser totalmente representado pelos parâmetros c , g e H . Da mesma forma, o algoritmo estudado requer que os polinômios de Lagrange (2.4.1) associados a um conjunto $\mathcal{Y} \in \mathbb{R}^n$ de q pontos amostrais sejam explicitamente calculados. Isso significa que, considerando o j -ésimo polinômio de Lagrange

$$\ell_j(x) = c_j + g_j^\top(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^\top H_j(x - \bar{x}), \quad (4.2.2)$$

analogamente ao que ocorre com o modelo m , o número $c_j \in \mathbb{R}$, o vetor $g_j \in \mathbb{R}^n$ e a matriz simétrica $H_j \in \mathbb{R}^{n \times n}$ devem ser explicitamente calculados para cada $j \in \{1, 2, \dots, q\}$.

Como cada matriz H_j é simétrica, para determiná-la completamente são necessários apenas os elementos de uma de suas partes triangulares¹, resultando em $1 + 2 + \dots + n = (n+1)n/2$ elementos para cada matriz. Assim, o número de coeficientes necessários para construir um polinômio $\ell_j(x)$ é de

$$\underbrace{1}_{c_j} + \underbrace{n}_{g_j} + \underbrace{\frac{1}{2}(n+1)n}_{H_j} = \frac{1}{2}(n+1)(n+2)$$

elementos, que é igual a q . Portanto, para determinar todos os q polinômios de Lagrange associados a um conjunto \mathcal{Y} é necessário que um total de q^2 coeficientes sejam determinados.

Seja, agora, a base natural (2.3.3) centrada em $\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)^\top \in \mathbb{R}^n$:

$$\bar{\Phi}_{\bar{x}} = \left\{ 1, z_1, z_2, \dots, z_n, z_1 z_2, z_1 z_3, \dots, z_{n-1} z_n, \frac{1}{2} z_1^2, \frac{1}{2} z_2^2, \dots, \frac{1}{2} z_n^2 \right\}, \quad (4.2.3)$$

onde $z \in \mathbb{R}^n$ é tal que $z = (z_1, z_2, \dots, z_n)^\top = x - \bar{x}$.

Computacionalmente, optou-se por representar os polinômios de Lagrange através de uma matriz $L \in \mathbb{R}^{q \times q}$ cujas entradas são seus coeficientes na base (ordenada) $\bar{\Phi}_{\bar{x}}$ centrada em torno de algum \bar{x} conveniente. O mesmo foi feito para o modelo aproximador da função objetivo, que também é um polinômio em \mathcal{P}_n^2 e necessita de um vetor $m \in \mathbb{R}^q$ de coeficientes reais para representá-lo nessa mesma base. O conjunto \mathcal{Y} , por sua vez, foi tratado como uma matriz cujas colunas

¹ A parte triangular superior (resp., inferior) de uma matriz $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ é composta por seus elementos a_{ij} tais que $i \leq j$ (resp., $j \leq i$).

representam seus elementos. Note-se que L e m estão totalmente vinculados à base $\bar{\Phi}_{\bar{x}}$, da mesma maneira que o conjunto \mathcal{Y} está vinculado à base canônica do \mathbb{R}^n .

Os coeficientes dos polinômios de Lagrange foram armazenados na matriz L de maneira que cada $\ell_i(x)$ fosse alocado na i -ésima linha de L , segundo uma ordem (fixa) associada à dos pontos y^j na matriz que representa \mathcal{Y} . Ainda, a ordenação da base $\bar{\Phi}_{\bar{x}}$ determina a alocação dos coeficientes de cada polinômio $\ell_i(x)$ na linha que o representa. Sendo L_{ij} a entrada da linha i e coluna j da matriz L , tem-se, portanto, que L_{ij} é o coeficiente que multiplica o j -ésimo elemento da base ordenada $\bar{\Phi}_{\bar{x}}$ quando da recuperação do polinômio $\ell_i(x)$ através de L .

Vale recordar que, uma vez que a base de polinômios de Lagrange esteja disponível, o modelo aproximador da função objetivo pode ser calculado a partir da Equação (2.5.5). Portanto, a disponibilidade desses polinômios permite que m seja construído com q^2 operações adicionais, o que mantém a ordem $\mathcal{O}(q^2)$ no número de operações².

Mais adiante, será descrito um procedimento proposto por Powell [40] de atualização da base de polinômios de Lagrange quando apenas um ponto do conjunto \mathcal{Y} é substituído por outro. Essa procedimento foi desenvolvido para ser feito em $\mathcal{O}(q^2)$ operações, o que culmina em $\mathcal{O}(q^4)$ operações por iteração típica do algoritmo.

4.3 Inicialização do Algoritmo

Sobre o Passo 0 de inicialização do DFO-LP, optou-se pelo procedimento descrito por Powell [40, Seção 5] que trata de um mecanismo perspicaz e conveniente de construção dos conjuntos \mathcal{Y}_0 (posicionado) e L_0 , além do modelo aproximador inicial, a partir de um único ponto y^1 dado e que resulta em estruturas iniciais consideravelmente esparsas. Nesta seção, espera-se contribuir para uma melhor compreensão das sugestões e resultados de Powell, apresentadas de maneira sucinta possivelmente devido ao contexto de sua publicação. Para a inicialização, considerou-se $\bar{x} = x^1$ nas Equações (4.2.1) e (4.2.2). Serão determinados, portanto, o conjunto amostral, o polinômio interpolador e os polinômios de Lagrange, nessa ordem.

4.3.1 O conjunto inicial de pontos amostrais

Seja $\mathcal{Y}_0 = \{y^1, y^2, \dots, y^q\}$ o conjunto de $q = (n+1)(n+2)/2$ pontos a ser determinado e $\Delta_0 > 0$ o raio da região de confiança inicial. Admitindo o ponto inicial padrão do problema como y^1 , a primeira parte do procedimento sugerido por Powell [40] está encarregada de determinar outros $2n$ pontos do conjunto a partir de múltiplos apropriados dos vetores canônicos do \mathbb{R}^n . Sem perda de generalidade, considere o subconjunto $\{y^2, y^3, \dots, y^{2n+1}\} \subset \mathcal{Y}_0$. Assim, sejam estabelecidos os pontos

$$y^{2j} = y^1 + \Delta_0 e^j \quad \text{para } j = 1, 2, \dots, n, \quad (4.3.1)$$

onde e^j é o j -ésimo vetor canônico no \mathbb{R}^n . A tática, agora, consiste em avaliar o valor da função objetivo nesses n pontos e, baseando-se nos decréscimos de f proporcionados por eles, determinar

² Uma função $h(n)$ é dita ser de ordem $\mathcal{O}(g(n))$, ou simplesmente, $h(n) = \mathcal{O}(g(n))$, se e somente se existirem $M > 0$ e $n_0 \in \mathbb{R}$ tais que $|h(n)| \leq M|g(n)|$ para todo e qualquer $n > n_0$.

os n pontos adicionais. Mais especificamente, serão incorporados os pontos

$$y^{2j+1} = y^1 + \xi_j e^j, \text{ para } j = 1, 2, \dots, n, \quad (4.3.2)$$

onde

$$\xi_j = \begin{cases} -\Delta_0, & \text{se } f(y^{2j}) < f(y^1) \\ 2\Delta_0, & \text{se } f(y^{2j}) \geq f(y^1) \end{cases}. \quad (4.3.3)$$

Na segunda parte do procedimento de construção de \mathcal{Y}_0 , propõe-se a utilização não de um, mas dois, vetores canônicos para determinar cada um dos pontos extras, agora indexados através de pares ordenados, da seguinte maneira:

$$y^{i(j,k)} = y^1 + \rho_j e^j + \rho_k e^k, \text{ para } 1 \leq j < k \leq n, \quad (4.3.4)$$

onde

$$\rho_j = \begin{cases} \Delta_0, & \text{se } f(y^{2j}) < f(y^1) \\ -\Delta_0, & \text{se } f(y^{2j}) \geq f(y^1) \end{cases}, \text{ para } j = 1, 2, \dots, n. \quad (4.3.5)$$

Note que, novamente, os valores da função objetivo nos pontos $y^{2j}, j = 1, 2, \dots, n$, são utilizados para direcionar a construção dos demais pontos. Observe, também, que estabelecendo a relação

$$i(j, k) = 2n + 1 + j + \frac{1}{2}(k - 1)(k - 2), \text{ para } 1 \leq j < k \leq n, \quad (4.3.6)$$

fica claro que os pontos determinados pela Equação (4.3.4) são os y^i para $i = 2n + 1, 2n + 2, \dots, q$, ou seja, os pontos que faltavam para completar \mathcal{Y}_0 .

Vale destacar que esse esquema de escolha de pontos, da maneira como foi proposto por Powell, visa explorar os locais onde a função objetivo parece estar decrescendo.

4.3.2 O modelo aproximador inicial

Por conveniência, considere o modelo inicial $m_0(x)$ caracterizado pelos parâmetros c_0 , g^0 e H^0 , e escrito em torno de $\bar{x} = y^1$, conforme a Equação (4.2.1):

$$m_0(y^i) = c_0 + (g^0)^\top (y^i - y^1) + \frac{1}{2} (y^i - y^1)^\top H^0 (y^i - y^1), \text{ para } i = 1, 2, \dots, q. \quad (4.3.7)$$

Fazendo $i = 1$ na expressão anterior, obtém-se que

$$c_0 = m_0(y^1). \quad (4.3.8)$$

Para os demais valores de i , convém que se analise separadamente o termo $y^i - y^1$. Assim, para $i = 2, 3, \dots, 2n + 1$, da definição dada pelas Equações (4.3.1) e (4.3.2), tem-se que

$$\begin{aligned} y^{2j} - y^1 &= (y^1 + \Delta_0 e^j) - y^1 = \Delta_0 e^j \\ y^{2j+1} - y^1 &= (y^1 + \xi_j e^j) - y^1 = \xi_j e^j \end{aligned}, \text{ para } j = 1, 2, \dots, n. \quad (4.3.9)$$

e, portanto, utilizando as Equações (4.3.9) e (4.3.7), junto com a notação de subíndices para as compontes de g^0 e entradas de H^0 ,

$$\begin{aligned} m_0(y^{2j}) &= c_0 + (g^0)^\top (\Delta_0 e^j) + \frac{1}{2} (\Delta_0 e^j)^\top H^0 (\Delta_0 e^j) \\ &= c_0 + \Delta_0 g_j^0 + \frac{1}{2} \Delta_0^2 H_{jj}^0 \\ m_0(y^{2j+1}) &= c_0 + (g^0)^\top (\xi_j e^j) + \frac{1}{2} (\xi_j e^j)^\top H^0 (\xi_j e^j) \\ &= c_0 + \xi_j g_j^0 + \frac{1}{2} \xi_j^2 H_{jj}^0 \end{aligned} \quad , \quad (4.3.10)$$

para $j = 1, 2, \dots, n$.

Das condições de interpolação dadas pela Equação (2.3.5), sabe-se que o modelo m_0 deve satisfazer o valor da função objetivo em todos os pontos de \mathcal{Y}_0 . Em particular, isso significa que $m_0(y^1)$ é igual a $f(y^1)$ e, da Equação (4.3.8), obtemos o valor da constante do modelo inicial:

$$c_0 = f(y^1).$$

Seguindo essa linha de raciocínio, $m_0(y^{2j}) = f(y^{2j})$ e $m_0(y^{2j+1}) = f(y^{2j+1})$ e, portanto, utilizando os resultados da Equação (4.3.10), obtém-se o seguinte conjunto de sistemas lineares 2×2 nas variáveis g_j e H_{jj} :

$$\begin{cases} f(y^1) + \Delta_0 g_j^0 + \frac{1}{2} \Delta_0^2 H_{jj}^0 &= f(y^{2j}) \\ f(y^1) + \xi_j g_j^0 + \frac{1}{2} \xi_j^2 H_{jj}^0 &= f(y^{2j+1}) \end{cases}, \quad j = 1, 2, \dots, n. \quad (4.3.11)$$

As soluções desses sistemas não são difíceis de serem encontradas e são dadas por

$$\begin{aligned} g_j^0 &= \frac{1}{\xi_j - \Delta_0} \left(\frac{a_j \xi_j}{\Delta_0} - \frac{b_j \Delta_0}{\xi_j} \right) \\ H_{jj}^0 &= \frac{1}{\xi_j - \Delta_0} \left(\frac{-2a_j}{\Delta_0} + \frac{2b_j}{\xi_j} \right) \end{aligned}, \quad j = 1, 2, \dots, n, \quad (4.3.12)$$

onde $a_j = [f(y^{2j}) - f(y^1)]$ e $b_j = [f(y^{2j+1}) - f(y^1)]$. Observe que $\xi_j \neq \Delta_0$ e que $\xi_j \Delta_0 \neq 0$, de modo que a solução do sistema anterior está bem definida.

Com isso, o vetor g^0 está completamente determinado, enquanto que, até o momento, apenas a diagonal da matriz H^0 foi estabelecida.

Agora, a fim de determinar os demais elementos de H^0 , considere os pontos $y^{i(j,k)}$ dados pela Equação (4.3.4) e os termos

$$y^{i(j,k)} - y^1 = (y^1 + \rho_j e^j + \rho_k e^k) - y^1 = \rho_j e^j + \rho_k e^k. \quad (4.3.13)$$

Dessa forma, também da Equação (2.3.5), segue que

$$\begin{aligned} m_0(y^{i(j,k)}) &= c_0 + (g^0)^\top (\rho_j e^j + \rho_k e^k) + \frac{1}{2} (\rho_j e^j + \rho_k e^k)^\top H^0 (\rho_j e^j + \rho_k e^k) \\ &= c_0 + \rho_j g_j^0 + \rho_k g_k^0 + \frac{1}{2} \rho_j^2 H_{jj}^0 + \rho_j \rho_k H_{jk}^0 + \frac{1}{2} \rho_k^2 H_{kk}^0 \end{aligned} \quad , \quad (4.3.14)$$

para $j = 1, 2, \dots, n$. Das condições de interpolação (2.3.5), chega-se às equações independentes

$$c_0 + \rho_j g_j^0 + \rho_k g_k^0 + \frac{1}{2} \rho_j^2 H_{jj}^0 + \rho_j \rho_k H_{jk}^0 + \frac{1}{2} \rho_k^2 H_{kk}^0 = f(y^{i(j,k)}), \quad 1 \leq j < k \leq n, \quad (4.3.15)$$

cujas únicas incógnitas são H_{jk}^0 , dado que c_0 , g^0 e a diagonal de H^0 já foram estabelecidas anteriormente pelas Equações (4.3.8) e (4.3.12). Dessa maneira,

$$H_{jk}^0 = \frac{1}{\rho_j \rho_k} \left[f(y^{i(j,k)}) - \left(c_0 + \rho_j g_j^0 + \rho_k g_k^0 + \frac{1}{2} \rho_j^2 H_{jj}^0 + \frac{1}{2} \rho_k^2 H_{kk}^0 \right) \right]. \quad (4.3.16)$$

Novamente, a solução está bem definida pois $\rho_j \rho_k \neq 0$.

4.3.3 A base inicial de polinômios de Lagrange

Ainda seguindo as sugestões de Powell [40], o conjunto $\{\ell_i(x), i = 1, 2, \dots, q\}$ será determinado na ordem inversa de seus índices, por blocos, no sentido de que os polinômios com menores índices i serão determinados por último.

Recorde que, na Seção 2.4, destacou-se que os polinômios de Lagrange ℓ_j da base Φ_ℓ podem ser vistos como os polinômios quadráticos cujos coeficientes são dados pela solução do Sistema (2.3.7) com o lado direito substituído pelo equivalente a uma função que se anula em todos os pontos de \mathcal{Y} , exceto no respectivo y^j , no qual vale 1. De fato, esse sistema foi explicitado na Equação (2.4.2). Reforça-se essa analogia porque, através dela, é possível enxergar nos procedimentos descritos na Seção 4.3.2 uma metodologia de determinação dos coeficientes de cada um dos q polinômios ℓ_j . Basta substituir os valores de $f(y^i)$ apropriadamente por 0 ou 1 e, baseados neles, fixar as constantes ρ_j e ξ_j .

O processo sugerido é tedioso e árduo, uma vez que deve ser repetido diversas vezes, separando-se uns casos de outros, e também em grupos, até que todos os q polinômios possam ser caracterizados completamente por seus coeficientes. Entretanto, não há maior dificuldade do que essa, tendo em vista que o todo o processo se trata de repetir os procedimentos da Seção 4.3.2 múltiplas vezes.

Em seu trabalho, Powell [40] se apoia em polinômios auxiliares $\hat{\ell}(x)$ e, definindo os polinômios de Lagrange na ordem decrescente de seus coeficientes, simplifica imensamente o processo. Seus resultados são exibidos a seguir.

Dado o conjunto \mathcal{Y}_0 construído na Seção 4.3.1, um número $n(n-1)/2$ dos polinômios de Lagrange associados a \mathcal{Y}_0 pode ser determinado pela expressão

$$\ell_i(x) = \frac{1}{\rho_j \rho_k} (x_j - y_j^1) (x_k - y_k^1), \quad 1 \leq j < k \leq n, \quad (4.3.17)$$

onde $i = i(j, k) = 2n+1+j+\frac{1}{2}(k-1)(k-2)$, $x = (x_1, x_2, \dots, x_n)^\top \in \mathbb{R}^n$ e $y^j = (y_1^j, y_2^j, \dots, y_n^j)^\top \in \mathbb{R}^n$.

É uma tarefa minuciosa mostrar que esses polinômios são, de fato, polinômios de Lagrange associados ao conjunto \mathcal{Y}_0 . Entretanto, realizá-la agora fará com que outras demonstrações análogas possam ser omitidas mais adiante, sem prejuízo de compreensão. Por isso, o resultado é formalizado a seguir.

Lema 4.3.1. Os polinômios definidos pela Equação (4.3.17) satisfazem as condições de Lagrange

$$\ell_i(y^t) = \delta_{ti} = \begin{cases} 1 & \text{se } i = t \\ 0 & \text{se } i \neq t \end{cases} \quad (4.3.18)$$

para $i = i(j, k) = 2n + 1 + j + \frac{1}{2}(k - 1)(k - 2)$ ou, equivalentemente, para $i = 2n + 2, 2n + 3, \dots, q$ e $t = 1, 2, \dots, q$, sendo $\mathcal{Y}_0 = \{y^1, y^2, \dots, y^q\}$ o conjunto construído na Seção 4.3.1.

Demonstração. Considere os seguintes termos presentes na definição dos polinômios dada pela Equação (4.3.17): $(x_j - y_j^1)$ e $(x_k - y_k^1)$. Por conveniência, sejam os pontos de \mathcal{Y}_0 separados em quatro subconjuntos disjuntos, de acordo com suas definições.

I) O subconjunto $\{y^1\}$ unitário.

Como $(y_j^1 - y_j^1) = (y_k^1 - y_k^1) = 0$, então $\ell_i(y^1) = 0$ para todo $i = 2n + 2, 2n + 3, \dots, q$.

II) O subconjunto $\{y^t; t = 2r, r = 1, 2, \dots, n\}$ definido pela Equação (4.3.1).

Aqui, tem-se que

$$\begin{aligned} (y_j^t - y_j^1) &= y_j^1 + \Delta_0 e_j^r - y_j^1 = \Delta_0 \delta_{rj} \\ (y_k^t - y_k^1) &= y_k^1 + \Delta_0 e_k^r - y_k^1 = \Delta_0 \delta_{rk} \end{aligned}$$

e, portanto,

$$\ell_i(y^t) = \frac{1}{\rho_j \rho_k} \Delta_0^2 \delta_{rj} \delta_{rk} = 0, \quad (4.3.19)$$

pois $j \neq k$ quando $i = 2n + 2, 2n + 3, \dots, q$ e $t = 2, 4, \dots, 2n$.

III) O subconjunto $\{y^t; t = 2r + 1, r = 1, 2, \dots, n\}$ definido pela Equação (4.3.2).

Analogamente,

$$\begin{aligned} (y_j^t - y_j^1) &= y_j^1 + \xi_r e_j^r - y_j^1 = \xi_r \delta_{rj} \\ (y_k^t - y_k^1) &= y_k^1 + \xi_r e_k^r - y_k^1 = \xi_r \delta_{rk} \end{aligned}$$

e, portanto,

$$\ell_i(y^t) = \frac{1}{\rho_j \rho_k} \xi_r^2 \delta_{rj} \delta_{rk} = 0, \quad (4.3.20)$$

pois $j \neq k$ quando $i = 2n + 2, 2n + 3, \dots, q$ e $t = 3, 5, \dots, 2n + 1$.

IV) O subconjunto $\{y^t; t = t(u, v), 1 \leq u < v \leq n\}$ definido pela Equação (4.3.4).

Por fim,

$$\begin{aligned} (y_j^t - y_j^1) &= y_j^1 + \rho_u e_j^u + \rho_v e_j^v - y_j^1 = \rho_u \delta_{uj} + \rho_v \delta_{vj} \\ (y_k^t - y_k^1) &= y_k^1 + \rho_u e_k^u + \rho_v e_k^v - y_k^1 = \rho_u \delta_{uk} + \rho_v \delta_{vk} \end{aligned}$$

e, daí,

$$\begin{aligned}\ell_i(y^t) &= \frac{1}{\rho_j \rho_k} (\rho_u^2 \delta_{uj} \delta_{uk} + \rho_u \rho_v \delta_{uj} \delta_{vk} + \rho_u \rho_v \delta_{uk} \delta_{vj} + \rho_v^2 \delta_{vj} \delta_{vk}) \\ &= \frac{1}{\rho_j \rho_k} \rho_u \rho_v \delta_{uj} \delta_{vk} = \delta_{ti},\end{aligned}\tag{4.3.21}$$

pois, se $j = u$ e $k = v$, ademais de $t = j$, vale que $\delta_{uj} \delta_{vk} = 1$, levando em consideração que $j \neq k$, $j < k$ e $u < v$.

Portanto, dos itens I) a IV), concluímos que $\ell_i(y^t) = \delta_{ti}$ para $i = 2n + 2, 2n + 3, \dots, q$, e $t = 1, 2, \dots, q$. □

Considere, agora, os seguintes polinômios auxiliares $\hat{\ell} \in \mathcal{P}_n^2$:

$$\begin{aligned}\hat{\ell}_{2j}(x) &= \frac{1}{\Delta_0(\Delta_0 - \xi_j)} (x_j - y_j^1) (x_j - y_j^{2j+1}) \\ \hat{\ell}_{2j+1}(x) &= \frac{1}{\xi_j(\xi_j - \Delta_0)} (x_j - y_j^1) (x_j - y_j^{2j})\end{aligned}, \quad j = 1, 2, \dots, n, \quad x \in \mathbb{R}^n.\tag{4.3.22}$$

Avaliando esse conjunto de polinômios nos pontos dados pelas Equações (4.3.1) e (4.3.4), de maneira semelhante ao que foi feito na demonstração do Lema 4.3.1, tem-se que

$$\hat{\ell}_i(y^t) = \delta_{ti}, \quad \text{para } t = 1, 2, \dots, 2n + 1.\tag{4.3.23}$$

Das Equações (4.3.18) e (4.3.23), é possível concluir, após uma análise cautelosa, que os polinômios dados por

$$\ell_i(x) = \hat{\ell}_i(x) - \sum_{t=2n+2}^m \hat{\ell}_i(y^t) \ell_t(x), \quad i = 2, 3, \dots, n + 1, \quad x \in \mathbb{R}^n\tag{4.3.24}$$

também satisfazem as condições de Lagrange $\ell_i(y^t) = \delta_{ti}$, $t = 1, 2, \dots, q$.

Portanto, o polinômio restante, $\ell_1(x)$, expresso por

$$\ell_1(x) = 1 - \sum_{t=2}^q \ell_t(x), \quad x \in \mathbb{R}^n\tag{4.3.25}$$

é, também, um dos polinômios de Lagrange associados a \mathcal{Y}_0 .

Contagem dos coeficientes dos polinômios de Lagrange

A quantidade de coeficientes (não nulos) necessários para a completa determinação da base inicial de polinômios de Lagrange é um bom indicativo do esforço computacional envolvido nesse processo. Vale lembrar que, desde o princípio, a escolha dos pontos de \mathcal{Y}_0 foi feita já visando tal economia. Outro ponto importante a se observar é o de que a escolha da base (4.2.3) centrada em $\bar{x} = y^1$ é conveniente e simplifica bastante os cálculos.

Primeiramente, observe que cada um dos polinômios $\ell_i(x)$ definidos pela Equação (4.3.17), quando representado na base natural centrada em y^1 (cf. 4.2.3), depende apenas do coeficiente $(\rho_j \rho_k)^{-1}$, associado ao termo misto $(x_j - y_j^1)(x_k - y_k^1)$. Se escritos em sua forma matricial (4.2.2), os parâmetros $c \in \mathbb{R}$ e $g \in \mathbb{R}^n$ de cada um desses polinômios seriam nulos. Ainda, sua matriz Hessiana H (simétrica) possuiria todas entradas nulas, exceto as de posição (j, k) e (k, j) , que valeriam $H_{j,k} = H_{k,j} = (\rho_j \rho_k)^{-1}$. Alternativamente, se utilizados os vetores $e^j \in \mathbb{R}^n$ canônicos, H poderia ser representada por $H = (\rho_j \rho_k)^{-1} (e^j (e^k)^\top + e^k (e^j)^\top)$. De qualquer maneira, apenas um elemento de cada um desses polinômios é necessário para caracterizá-lo, dada a simetria de H .

Seguindo com a contagem, considere os polinômios dados pela Equação (4.3.22). Substituindo y^{2j+1} e y^{2j} pelos pontos que equivalem, obtém-se

$$\begin{aligned}\hat{\ell}_{2j}(x) &= \frac{(x_j - y_j^1)(x_j - y_j^1 - \xi_j)}{\Delta_0(\Delta_0 - \xi_j)} = \frac{1}{\Delta_0(\Delta_0 - \xi_j)} (x_j - y_j^1)^2 - \frac{\xi_j}{\Delta_0(\Delta_0 - \xi_j)} (x_j - y_j^1) \text{ e} \\ \hat{\ell}_{2j+1}(x) &= \frac{(x_j - y_j^1)(x_j - y_j^1 - \Delta_0)}{\xi_j(\xi_j - \Delta_0)} = \frac{1}{\xi_j(\xi_j - \Delta_0)} (x_j - y_j^1)^2 - \frac{\Delta_0}{\xi_j(\xi_j - \Delta_0)} (x_j - y_j^1),\end{aligned}\tag{4.3.26}$$

para $j = 1, 2, \dots, n$. Portanto, para cada $\hat{\ell}_i$, $i = 2, 3, \dots, 2n + 1$, são necessários apenas dois coeficientes para construí-lo. Novamente, caso estivesse escrito em sua forma matricial, $\hat{\ell}_{2j}$ possuiria coeficientes

$$c = 0, \quad g = -\frac{\xi_j}{\Delta_0(\Delta_0 - \xi_j)} e^j \text{ e } H = \frac{2}{\Delta_0(\Delta_0 - \xi_j)} e^j (e^j)^\top,$$

enquanto que $\hat{\ell}_{2j+1}$ possuiria

$$c = 0, \quad g = -\frac{\Delta_0}{\xi_j(\xi_j - \Delta_0)} e^j \text{ e } H = \frac{2}{\xi_j(\xi_j - \Delta_0)} e^j (e^j)^\top.$$

Powell observa que, para calcular o valor $\hat{\ell}_i(y^t)$ para um dado i e para $t = 2n + 2, 2n + 3, \dots, q$, são necessários $n - 1$ pontos. Isso pode ser visto através da substituição dos pontos y^t , definidos via Equação (4.3.4), nas Equações (4.3.22). Nesse caso, adotando uma indexação $t = t(u, v)$, com $1 \leq u < v \leq n$, e $i = 2j$, para $j = 1, 2, \dots, n$, é possível enxergar que quando se fixa j , os únicos casos em que $\hat{\ell}_{2j}(y^{t(u,v)})$ não se anula são aqueles em que $u = j$ ou $v = j$, o que ocorre para $n - 1$ pontos y^t . O mesmo vale para $\hat{\ell}_{2j+1}$.

Dessas observações, pode-se concluir que, fixado i , o polinômio resultante da Equação (4.3.24) possui não mais do que $n + 1$ parâmetros não nulos (2 provenientes de $\hat{\ell}_i(x)$ e $n - 1$, de $\hat{\ell}_i(y^t)$), e que todos os coeficientes de $\ell_1(x)$ podem ser não nulos. Isso significa que a construção do conjunto inicial de polinômios de Lagrange pode ser realizada em ordem $\mathcal{O}(n^2)$ de operações computacionais.

4.4 O parâmetro ϵ_0

Originalmente presente no Algoritmo **DFO-LP** 3.1, o parâmetro ϵ_0 determina o momento em que, pela primeira vez na execução do método proposto, será exigida a construção de um modelo Λ -posicionado. Junto com outras constantes, o valor de ϵ_0 não é sugerido pelos autores, ficando a critério do programador o valor que ele assume. Entretanto, curiosamente e à diferença das demais constantes, nenhuma menção explícita é feita sobre a faixa de valores que esse parâmetro pode assumir. Obviamente, devido à sua aplicação, fica subentendido que ϵ_0 deve assumir algum

valor positivo. Pela sua natureza, fica claro que ϵ_0 deva depender do problema tratado. Por exemplo, se ϵ_0 é maior ou igual à norma do gradiente inicial do problema, já na primeira iteração um modelo Λ -posicionado será desnecessariamente solicitado. Portanto, a determinação do valor desse parâmetro será feita através de experimentos computacionais, descritos no Capítulo 8.

4.5 A construção de um modelo Λ -posicionado

Como a proposta deste trabalho é a adoção dos polinômios de Lagrange como ferramentas para o controle do posicionamento do conjunto amostral, optou-se pela utilização de um subalgoritmo que faz uso desses polinômios para construir modelos Λ -posicionados, no Passo 1 do Algoritmo 3.1 DFO-LP. Uma instância do subalgoritmo, originalmente proposto por Conn, Scheinberg e Vicente [14, Algoritmo 6.2] é essencialmente reproduzida a seguir. A tarefa consiste em verificar se o conjunto é Λ -posicionado baseado no valor dos polinômios de Lagrange na região corrente de interesse. Caso não seja, o ponto associado ao polinômio cujo valor absoluto ultrapassa Λ é substituído por outro que maximize esse polinômio na mesma região. Os polinômios de Lagrange são atualizados, então, ao fim de cada iteração.

A demonstração de que esse procedimento resulta num conjunto Λ -posicionado após um número finito e uniformemente limitado de passos pode ser consultada em Conn, Scheinberg e Vicente [14, Teorema 6.3].

Algoritmo de correção do posicionamento dos pontos de \mathcal{Y}

Passo 0: Inicialização. Seja um conjunto \mathcal{Y} de cardinalidade q dado e seus respectivos polinômios de Lagrange $\ell_i(x)$, $i = 1, 2, \dots, q$, associados. Dada uma região $\mathcal{B} \subset \mathbb{R}^n$ e um ponto corrente $\bar{x} \in \mathbb{R}^n$, escolha uma constante $\Lambda > 1$ e faça $k = 1$.

Passo 1: Determinação de Λ_{k-1} .

$$\text{Calcule } \Lambda_{k-1} = \max_{1 \leq i \leq q} \max_{x \in \mathcal{B}} |\ell_i(x)|.$$

Passo 2: Atualização de \mathcal{Y} . Se $\Lambda_{k-1} \leq \Lambda$, decrete o conjunto \mathcal{Y} como Λ -posicionado em \mathcal{B} e pare. Senão, prossiga com os demais passos.

Passo 2a: Determine um ponto a ser eliminado de \mathcal{Y} .

Seja $\mathcal{I} = \{i_1, i_2, \dots\} \subset \{1, 2, \dots, q\}$ o conjunto de índices i_k para os quais

$$\max_{x \in \mathcal{B}} |\ell_{i_k}(x)| > \Lambda.$$

Tome j tal que

$$j \in \arg \max \{\|y^i - \bar{x}\|; i \in \mathcal{I}\}$$

como índice do ponto a ser eliminado de \mathcal{Y} .

Passo 2b: Determine o ponto a ser incluído em \mathcal{Y} . Tome $y_*^j = \arg \max_{x \in \mathcal{B}} |\ell_j(x)|$ como novo ponto amostral.

Passo 2c: Atualização de \mathcal{Y} . Realize a troca de pontos

$$\mathcal{Y} \leftarrow \mathcal{Y} \cup \{y_*^j\} \setminus \{y^j\}.$$

Passo 3: Atualização dos coeficientes dos polinômios de Lagrange.

Sendo j o mesmo do passo anterior, faça

$$\begin{aligned} \ell_j(x) &\leftarrow \ell_j(x) / \ell_j(y_*^j) \\ \ell_\sigma(x) &\leftarrow \ell_\sigma(x) - \ell_\sigma(y_*^j) \ell_j(x), \quad \sigma \in \{1, 2, \dots, q\} \setminus \{j\} \end{aligned}$$

Incremente k em uma unidade e volte para o Passo 1.

Junto com os procedimentos padrão desse algoritmo, foi necessário implantar um sistema de rastreamento dos pontos substituídos, uma vez que somente nesses pontos é que a função objetivo precisa ser avaliada.

4.5.1 Otimização Global de $|\ell_j(x)|$

O subproblema que surge no Passo 1 do DFO-LP com a função de avaliar o posicionamento do conjunto de pontos foi resolvido “exatamente” com o auxílio da rotina `trust.m` do **MATLAB**, aplicada duas vezes por subproblema: em ℓ_j e em $-\ell_j$. É de conhecimento dos autores deste trabalho que, para problemas maiores do que os testados aqui, essa prática pode ser inconcebível e não recomendável sob o ponto de vista do esforço computacional investido. Entretanto, assumindo um contexto em que, de fato, o maior custo de resolução esteja associado ao número de avaliações da função objetivo (e não ao da álgebra linear computacional), o uso do `trust.m` se mostra bastante razoável.

Também cientes do custo envolvido nessa otimização global, Conn, Scheinberg e Vicente [14] recomendam alguns procedimentos que substituem a resolução “exata” por uma resolução baseada em limitantes para o valor dos polinômios na região \mathcal{B} de interesse. Entretanto, a substituição do subproblema original por outros que apenas determinam limitantes inferiores e/ou superiores faz com que o Teorema 6.3 [14] não seja mais diretamente aplicável ao algoritmo. Para o leitor interessado em alternativas à resolução desse problema, Powell descreve em seu trabalho com o *solver* UOBYQA [41] uma maneira de obter um problema que aproxime o subproblema em questão.

4.6 Atualização da matriz de coeficientes dos polinômios de Lagrange

Após a construção do conjunto inicial de polinômios de Lagrange, todas as alterações subsequentes que envolvam a troca de um único ponto do conjunto \mathcal{Y} serão realizadas da maneira que se descreve a seguir, conforme consta em Powell [40, Seção 2].

Considere a substituição de um ponto de interpolação $y^j \in \mathcal{Y}$ por outro $x^+ \in \mathbb{R}^n$. Essa simples troca faz com que todos os coeficientes dos polinômios de Lagrange precisem ser atualizados.

Sejam $\{\ell_i(x)\}_{i=1}^q$ os polinômios de Lagrange antes da realização da troca de pontos e $\{\tilde{\ell}_i(x)\}_{i=1}^q$, os atualizados. Ambos os conjuntos são bases para o espaço \mathcal{P}_n^2 . Fica claro ao tentar representar o polinômio $\tilde{\ell}_j(x)$ em termos da base $\{\ell_i(x)\}_{i=1}^q$ que ele deve ser um múltiplo de $\ell_j(x)$. Caso contrário, o novo polinômio $\tilde{\ell}_j(x)$ não se anularia nos pontos que foram mantidos em \mathcal{Y} . Portanto, da condição de que $\tilde{\ell}_j(x)$ deve valer 1 quando avaliado em y^j , segue que

$$\tilde{\ell}_j(x) = \ell_j(x)/\ell_j(x^+), \quad x \in \mathbb{R}^n. \quad (4.6.1)$$

Além disso, notando que $\tilde{\ell}_i$ e ℓ_i , para $i = 1, 2, \dots, q$, assumem os mesmos valores nos pontos inalterados de \mathcal{Y} , a diferença $\tilde{\ell}_i - \ell_i$ é um múltiplo de $\tilde{\ell}_j$, isto é,

$$\tilde{\ell}_i(x) - \ell_i(x) = \theta \tilde{\ell}_j(x), \quad x \in \mathbb{R}^n, \quad \theta \in \mathbb{R}.$$

Portanto, para $i \neq j$, como $\tilde{\ell}_i(x^+) = 0$ e $\tilde{\ell}_j(x^+) = 1$, conclui-se que $\theta = -\ell_i(x^+)$ e, assim,

$$\tilde{\ell}_i(x) = \ell_i(x) - \ell_i(x^+) \tilde{\ell}_j(x), \quad x \in \mathbb{R}^n. \quad (4.6.2)$$

Atentando para o fato de que, para todo $i = 1, 2, \dots, q$, $\ell_i(x^+)$ são números reais, o procedimento de atualização dos coeficientes dos polinômios de Lagrange descrito acima é de fácil implementação computacional. Não é difícil ver que tal atualização de coeficientes é da ordem de $\mathcal{O}(q^2)$ operações. Da Equação (2.5.5) sabe-se que o custo de calcular o modelo m se reduz ao custo de atualizar o conjunto de polinômios de Lagrange e, portanto, ambos compartilham a mesma complexidade. Repare, porém, que com os polinômios de Lagrange, tem-se à disposição uma série de recursos para o controle da geometria dos pontos.

4.7 O subproblema de região de confiança

O **Passo 2** do DFO-LP foi resolvido através do problema de região de confiança dado pela Equação (3.0.1), que garante que a condição de Cauchy (3.0.2) seja satisfeita, como pode ser visto em Conn, Gould e Toint [11, Seção 6.3]. Para tanto, utilizou-se a função `trust.m` do **MATLAB**, que corresponde essencialmente ao algoritmo de Moré e Sorensen [36], só que ao invés de empregar fatorações de Cholesky, utiliza a decomposição espectral completa da matriz Hessiana para resolver a equação secular do problema (ver [11, p. 182]), e por isso só é recomendado para problemas de pequeno porte.

4.8 O descarte de pontos em iterações de insucesso

Outra decisão em aberto no Algoritmo **DFO-LP** é a escolha do ponto y_r^k a ser descartado em iterações³ de insucesso (**Passo 4a** e **Passo 4b**). A proposta é a de que y_r^k seja escolhido

³ Por conta das iterações, os superíndices desta e da seção seguinte indicam a iteração a qual se refere, da mesma maneira como ocorre no Capítulo 4. Assim, os índices que antes ocupavam essas posições passam a ser subíndices. Nesses casos, por exemplo, $\ell_j^k(x)$, y_j^k e x_+^k correspondem, respectivamente, ao j -ésimo polinômio de Lagrange e aos vetores y_j e x_+ , todos da k -ésima iteração.

de maneira que maximize o produto do quadrado da distância entre ele e o ponto tentativo x_+^k com o valor absoluto do polinômio de Lagrange avaliado em x_+^k . Sobre esse critério, Gratton, Toint e Tröltzsch [26, Seção 3.5] ressaltam ser mais conveniente na prática a utilização de dois critérios distintos: se os candidatos a descarte estão distantes do ponto corrente (**Passo 4a**), que sejam escolhidos exclusivamente pelo critério distância; se, por outro lado, os candidatos a descarte estão próximos do ponto corrente (**Passo 4b**), é mais natural que sejam descartados apenas pelo critério do valor absoluto do polinômio de Lagrange em x_+^k . Portanto, baseando-se nessa proposta, as expressões (3.1.3) e (3.1.5) foram substituídas respectivamente por

$$y_r^k \in \arg \max_{y_i^k \in \mathcal{C}_k} \|y_j^k - x_+^k\|^2 \quad (4.8.1)$$

e

$$y_r^k \in \arg \max_{y_i^k \in \mathcal{C}_k} |\ell_j^k(x_+^k)|. \quad (4.8.2)$$

4.9 Atualização do raio da região de confiança

Seguindo o direcionamento de Conn, Gould e Toint [11, pp. 782], o procedimento de atualização do raio Δ da região de confiança adotado foi o seguinte:

$$\Delta_{k+1} = \begin{cases} \min\{\max\{\gamma_3 \|s^k\|, \Delta_k\}, \Delta_{max}\}, & \text{se } \rho_k \geq \eta_2 \\ \Delta_k, & \text{se } \eta_1 \leq \rho_k < \eta_2 \\ (\gamma_1 + \gamma_2)/2, & \text{se } \rho_k < \eta_1 \text{ e } x^k = v^i \text{ e } \Delta_k > \epsilon_i, \text{ ou,} \\ \Delta_k, & \text{se } \rho_k < \eta_1 \text{ e } \mathcal{F}_k \cup \mathcal{C}_k = \emptyset, \\ \Delta_k, & \text{caso contrário,} \end{cases}$$

onde x^k , v^i , ϵ_i , \mathcal{F}_k e \mathcal{C}_k são os mesmos do Algoritmo 3.1 e

$$0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3,$$

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{e}$$

$$\Delta_{max} > 0$$

são alguns de seus parâmetros.

Metodologia de comparação de *solvers* sem derivada

Após o embasamento teórico e os direcionamentos sobre a implementação do algoritmo DF0-LP, é preciso estabelecer uma metodologia e uma estrutura de testes computacionais que permitam tirar conclusões a respeito dessa instância do algoritmo. Em geral, define-se um conjunto \mathcal{P} de problemas a serem submetidos a cada um dos elementos do conjunto \mathcal{S} de *solvers* comparados. Portanto, o desempenho de um método é avaliado comparativamente. Está claro que o principal objetivo de um algoritmo de otimização é *resolver* um maior número de problemas realizando o menor *esforço* possível. Fazem parte da metodologia de testes a definição de um **critério de convergência** que estipule o significado prático de “resolver um problema” e, também, o estabelecimento de uma unidade de medida desse esforço. Não há um consenso a respeito dos procedimentos que devem ser adotados nas comparações em otimização sem derivada. Sendo assim, neste capítulo, é apresentada a metodologia proposta por Moré e Wild em [37], a qual será implantada nos experimentos computacionais do Capítulo 8.

5.1 Teste de Convergência

Em otimização sem derivadas, a comparação entre *solvers* usualmente se baseia no número de avaliações de função objetivo necessário para que se satisfaça determinado critério de convergência [21, 33, 42]. O *solver* que conseguir satisfazer esse critério com menor número de avaliações de função será considerado como aquele que melhor resolve o problema proposto. Costuma-se estabelecer um limite μ_f de avaliações de função que tem um caráter orçamentário e, portanto, não pode ser ultrapassado. Quando não se alcança satisfazer o critério de convergência dentro do orçamento μ_f , diz-se que o *solver* não é capaz de resolver o problema.

O critério ou teste de convergência em geral está relacionado a um ou mais parâmetros do algoritmo, responsáveis por indicar, ainda que indiretamente, a estacionariedade do ponto corrente. Por exemplo, um teste de convergência bastante utilizado em otimização via modelos de aproximação é aquele que verifica se a norma do gradiente do modelo é menor do que uma dada tolerância, como ocorre em [5].

Sobre isso, Moré e Wild [37] argumentam que a utilização de um teste de convergência que depende de uma certa tolerância parte do princípio de que a alteração dessa tolerância não afeta

significativamente o desempenho do *solver* e que o interesse do usuário concentra-se invariavelmente em soluções de alta precisão. No entanto, na prática, quando a avaliação de função é custosa, uma solução de baixa precisão é tudo o que se pode obter com um orçamento computacional modesto. Nesses casos, o interesse do usuário frequentemente reside no maior decréscimo no valor da função objetivo que se pode obter sem estourar o orçamento. Ainda, no caso da imprecisão dos dados disponíveis, não é realista esperar que a solução obtida seja muito acurada.

Seguindo essa linha de raciocínio, os autores de [37] propõem o seguinte teste de convergência para de um método $s \in \mathcal{S}$, na resolução de um problema $p \in \mathcal{P}$ que consiste na minimização irrestrita de uma função f :

$$f(x^0) - f(x) \geq (1 - \tau)(f(x^0) - f_L^p), \quad (5.1.1)$$

onde x^0 é o ponto inicial para o problema, $\tau \in [0, 1)$ é uma tolerância e f_L^p é o menor valor de função objetivo dentre os obtidos pelos *solvers* de \mathcal{S} para o problema p com μ_f avaliações de função disponíveis.

Da maneira como f_L^p está sendo definido, fica claro que o teste de convergência depende não somente do *solver* em questão mas também do desempenho dos demais elementos de \mathcal{S} . Para obter o valor f_L^p , é necessário submeter o problema p a todos os *solvers* de \mathcal{S} de maneira que o único critério de parada adotado seja o do número máximo de avaliações de função. Ou seja, a determinação da convergência é feita *a posteriori* e é totalmente dependente do desempenho de \mathcal{S} como um todo. Por exemplo, não há garantia de que algum dos *solvers* conseguirá obter o valor mínimo (local ou global) de f .

Observa-se que a Equação (5.1.1) exige que a redução $f(x^0) - f(x)$ obtida com o ponto x seja de, pelo menos, $(1 - \tau)$ vezes a melhor redução que se pode obter quando se dispõe dos *solvers* de \mathcal{S} . Esse mesmo teste de convergência foi adotado por Marazzi e Nocedal [33] mas com f_L^p sendo uma estimativa acurada para o valor de f em um minimizador local, obtida por um *software* com derivada. Moré e Wild apontam que, num contexto onde a avaliação de função é custosa, não é apropriado basear o referencial f_L^p em resultados de *solvers* com derivadas, pois é possível que nenhum *solver* seja capaz de satisfazer o teste de convergência com o orçamento computacional do usuário.

5.2 Perfis de desempenho e informativos

A fim de apresentar visualmente os resultados numéricos para todos os problemas e métodos considerados neste trabalho, foram adotadas duas ferramentas bastante úteis para tal propósito: os chamados perfis de desempenho e perfis informativos¹. Os perfis informativos foram recentemente propostos por Moré e Wild [37] para auxiliar na comparação de *solvers* sem derivada e servem como uma complementação aos perfis de desempenho de Dolan e Moré [20], estes amplamente adotados em toda a área de otimização. Ambos os perfis são detalhados a seguir.

¹ Os termos *perfil de desempenho* e *perfil informativo* são traduções dos termos em inglês *performance profile* e *data profile*.

5.2.1 Perfis de desempenho

Perfis de desempenho são funções de distribuição acumulada $\rho_s(\alpha)$ que representam o desempenho de determinado método $s \in \mathcal{S}$. Essas funções buscam captar quão bom é o desempenho do *solver* com relação aos outros elementos de \mathcal{S} , seus “concorrentes”, na resolução dos problemas de \mathcal{P} .

Seja $t_{p,s}$ uma medida de desempenho do *solver* $s \in \mathcal{S}$ no problema $p \in \mathcal{P}$, de maneira que menores valores de $t_{p,s}$ indiquem um melhor desempenho. Para comparar a medida $t_{p,s}$ com o melhor desempenho obtido para o problema p com os *solvers* de \mathcal{S} , define-se a taxa de desempenho para o par (p, s) da seguinte maneira:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}; s \in \mathcal{S}\}}.$$

Assim, o melhor *solver* para um problema p em particular obtém $r_{p,s} = 1$, o menor valor possível para essa taxa. Note que mais de um *solver* pode obter tal taxa de desempenho; é o caso em que o melhor desempenho (dentre os demais desempenhos de \mathcal{S}) para determinado problema tenha sido atingido por mais de um *solver* ou, equivalentemente, quando $\text{card}(\arg \min\{t_{p,s}; s \in \mathcal{S}\}) > 1$. Por convenção, quando um *solver* s não consegue satisfazer o teste de convergência quando da resolução de um problema p , faz-se $r_{p,s} = \infty$.

Dessa forma, definindo o **perfil de desempenho** $\rho_s(\alpha)$ do *solver* s por

$$\rho_s(\alpha) = \frac{1}{\text{card}(\mathcal{P})} \text{card}(\{p \in \mathcal{P} ; r_{p,s} \leq \alpha\}), \quad (5.2.1)$$

$\rho(\alpha)$ pode ser interpretado como o percentual de problemas de \mathcal{P} para os quais a taxa de desempenho é menor ou igual a α . Portanto, em caráter comparativo, a preferência deve ser por *solvers* com maior valor $\rho_s(\alpha)$.

A medida de desempenho adotada varia com a natureza dos métodos envolvidos na comparação. Por exemplo, é comum adotar a medida “tempo de execução” em otimização com derivadas. No presente contexto, onde se supõe que as informações sobre derivada não estão disponíveis ou não são confiáveis, assume-se que o custo dominante por iteração esteja atribuído ao número de avaliações de função, e não ao tempo de execução do código.

Como consequência da Equação (5.2.1), $\rho_s(1)$ é a probabilidade de que o *solver* s tenha a melhor performance entre todos os *solvers* e, para α suficientemente grande, $\rho(\alpha)$ é a fração de problemas que s conseguiu resolver com o orçamento disponível e dentro de determinada acurácia. Se o interesse estiver voltado à determinação de qual *solver* é o mais **eficiente** (no sentido de mais econômico na maior parte dos problemas), deve-se comparar o valor $\rho(1)$ para todos os *solvers*. Por outro lado, *solvers* com maiores valores de $\rho(\alpha)$ para α grande são aqueles que conseguiram resolver o maior número de problemas de \mathcal{P} , sendo por isso ditos serem mais **robustos**.

5.2.2 Perfis informativos

Por terem sido pensados para um contexto geral de otimização, os perfis de desempenho são ferramentas que provêm uma boa visão comparativa dos *solvers* para um usuário que disponha

efetivamente de um orçamento μ_f de avaliações de função, mas não fornecem informações para usuários com orçamentos distintos de μ_f . A hipótese implícita é a de que a maior importância reside no comportamento do algoritmo a longo prazo, o que não se verifica na prática quando f é uma função cara de se avaliar. Em geral, um usuário de otimização sem derivadas está interessado no desempenho dos *solvers* em termos do número de avaliações de função. Ainda assim, é preciso levar em conta a dimensão dos problemas de \mathcal{P} porque, na prática, observa-se que o gasto necessário para satisfazer determinado teste de convergência aumenta com o número de variáveis do problema. Para suprir a demanda por esse tipo de informação, Moré e Wild [37] introduzem os chamados perfis informativos.

Ainda considerando a medida de desempenho $t_{p,s}$ como sendo o número de avaliações de função necessário para satisfazer o teste de convergência (5.1.1), o **perfil informativo** de um *solver* $s \in \mathcal{S}$ é definido como

$$d_s(\alpha) = \frac{1}{\text{card}(\mathcal{P})} \text{card} \left(\left\{ p \in \mathcal{P}; \frac{t_{p,s}}{n_p + 1} \leq \alpha \right\} \right), \quad (5.2.2)$$

onde n_p é a dimensão do problema $p \in \mathcal{P}$. Moré e Wild [37] observam que a escala $n_p + 1$ presente na Equação (5.2.2) é, de certa forma, arbitrária. A interpretação dada pelos autores é a de que $n_p + 1$ é o número de avaliações de função gasto para calcular um gradiente simplex [7, p. 79]. Portanto, $d_s(\kappa)$ pode ser interpretado como o percentual de problemas que pode ser resolvido com o equivalente ao gasto para se calcular κ gradientes simplex². Perfis informativos são adequados para usuários com restrições orçamentárias que precisam tomar decisões a respeito de qual *solver* escolher para seu problema prático e, ainda, que estão interessados na redução do valor de sua função objetivo. Nesses casos, ele precisa expressar seu orçamento τ em termos de gradientes simplex, examinar os valores de d_s para os diferentes *solvers* e, então, verificar qual deles resolve mais problemas com um orçamento equivalente ao seu.

Assim como no caso dos perfis de desempenho, os perfis informativos são funções de distribuição acumulada e maiores valores de $d(\alpha)$ indicam melhores desempenhos. Entretanto, destaca-se que os perfis de desempenho possuem um caráter estritamente comparativo de *solvers*, enquanto que o perfil informativo d_s de um dado $s \in \mathcal{S}$ é independente de qualquer outro *solver*.

Para maiores caracterizações entre essas duas ferramentas e, ainda, para suas interessantes relações com testes de convergência de otimização com derivadas, consulte [37].

² Ou, por simplicidade, “o equivalente a κ gradientes simplex”.

Interface entre CUTer e MATLAB

Neste capítulo, serão tratados alguns detalhes referentes à configuração do ambiente de testes utilizado na composição dos problemas-teste e dos resultados computacionais deste trabalho. O objetivo não é servir como única referência para o assunto mas, sim, direcionar o leitor a fontes confiáveis e amigáveis, além de preencher algumas lacunas nas instruções-chave que possibilitam uma eventual reprodução dos resultados aqui obtidos, a partir do **MATLAB**.

6.1 Arquivos MEX

O **MATLAB** possui uma **Interface de Programação de Aplicativos (API)** que permite a execução de subrotinas escritas em diferentes linguagens e definidas por usuários externos. Isso significa que ele é capaz de chamar funções escritas na linguagem **C** ou **FORTRAN**, por exemplo, e possibilita o acesso à biblioteca e demais funcionalidades do **CUTer**, por possuir compiladores internos dessas linguagens.

Essa interface é provida por subrotinas intermediadoras carregadas dinamicamente e chamadas de *arquivos MEX* (do inglês *MATLAB Executable*). Quando compilados, os arquivos **MEX** permitem que códigos escritos em outras linguagens sejam executados como se fossem funções nativas do programa. Para maiores informações sobre arquivos **MEX**, consulte o manual do **MATLAB** [1].

6.2 Instalando o CUTer

Atualmente, o **CUTer** está disponível para diversas plataformas **UNIX** (incluindo **LINUX** e **MAC OS/X**), como também para a plataforma **WINDOWS** [23]. Para ter acesso às funcionalidades do ambiente de testes, é necessário que ele esteja corretamente instalado e devidamente configurado de acordo com a plataforma do computador.

Para tanto, de um modo geral, precisam ser instalados o decodificador **SIF** (*Standard Input Format* [10]) e o **CUTer** propriamente dito. Esse procedimento envolve *download*, descompactação e instalação de arquivos, e deve vir acompanhado de alterações simples porém específicas em algumas variáveis de ambiente.

Variáveis de ambiente são usadas para armazenar informações sobre o sistema operacional, caminhos de diretórios, configurações etc. Elas podem ser usadas tanto em *scripts* como em linhas de comando. Por exemplo, em **UNIX**, uma variável de ambiente comum é a **HOME**, que contém nela o caminho para o diretório raiz do usuário atual. Em **WINDOWS**, a **TEMP** guarda o caminho para o diretório temporário do usuário logado. Em geral, caracteres especiais são usados antes ou em volta delas para referenciá-las. Em **LINUX**, como é o caso do sistema adotado para os testes, as variáveis são precedidas pelo símbolo **\$**.

Durante a instalação do **CUTer**, as alterações em variáveis de ambiente consistem em modificar determinadas linhas de arquivos de configuração através de editores de texto comuns. Instruções detalhadas de instalação podem ser encontradas em manuais como o [44], em inglês, e o [50], em português.

Seguindo a nomenclatura desses manuais, ao fim da instalação, a variável de ambiente **CUTer** estará associada ao diretório no qual ocorreu a instalação do **CUTer**. **MYCUTer**, por sua vez, será um subdiretório de **CUTer** relacionado à instância da instalação concluída e, portanto, dependerá da máquina, do sistema operacional, do compilador **FORTRAN** instalado, da precisão de ponto flutuante e do porte dos problemas. Para ilustrar, suponha que a instalação tenha sido feita em um computador pessoal (PC) com sistema operacional **LINUX**, cujo compilador **FORTRAN** seja o **g95** e que se tenha optado por dupla precisão e problemas de grande porte. Nesse caso, a atribuição das variáveis de ambiente seria equivalente a:

Código 6.1: Exemplo de atribuição de variáveis de ambiente.

```
CUTer = "$HOME/CUTer/cuter2"
MYCUTer = "$CUTer/CUTer.large.pc.lnx.g95"
```

Portanto, ambas variáveis de ambiente dependem fortemente da configuração realizada durante a instalação do **CUTer**. Daí a importância de seguir as orientações de algum dos manuais disponíveis.

6.3 Procedimentos a partir do **MATLAB**

Uma vez que o **CUTer** esteja instalado e pronto para uso, é necessário, ainda, adicionar os caminhos de dois diretórios relacionados a ele ao **path** do **MATLAB**, para que este *saiba* onde buscar os dados dos problemas. Ainda na nomenclatura dos manuais mencionados, esse procedimento pode ser feito ao executar as seguintes intruções na janela de comando do **MATLAB**.

```
>> addpath $CUTer/common/src/matlab
>> addpath $MYCUTer/bin
```

Devido à dinâmica de geração de arquivos durante o acesso aos problemas, é recomendável a criação de uma pasta (aqui, chamada de **Test**) a partir da qual os testes serão conduzidos. Nesse momento, já deverá ser possível acionar o problema desejado a partir da janela de comando do **MATLAB**. Acessando a pasta **Test** recém-criada e tomando como exemplo a função Rosenbrock (**ROSENBR**), a execução do trecho do Código 6.2 poderá indicar se a configuração foi feita

adequadamente.

Código 6.2: Carregando o problema ROSENBR a partir do MATLAB.

```
1 >> system('runcuter --package mx --decode ROSENBR ');  
2 >> prob = cuter_setup()
```

Se tudo ocorrer bem, o resultado deverá ser semelhante a:

```
prob =  
      n: 2  
      m: 0  
    nnzh: 3  
    nnzj: 0  
      x: [2x1 double]  
      bl: [2x1 double]  
      bu: [2x1 double]  
      v: [0x1 double]  
      cl: [0x1 double]  
    equatn: [0x1 logical]  
    linear: [0x1 logical]  
    name: 'ROSENBR  '
```

Na primeira linha do Código 6.2, a função `system` executa no sistema operacional um comando que decodifica o problema (no caso, o de Rosenbrock) de **SIF** para **FORTRAN** e, em seguida, o associa ao arquivo de interface do **MATLAB**. Esse processo resulta na criação de arquivos em **FORTRAN** (*.f), do arquivo **MEX** (*.mexext¹), e de arquivos auxiliares gerados pelo decodificador (*.d).

Já na linha 2, as informações sobre o problema são carregadas pela função `cuter_setup` e armazenadas numa estrutura do **MATLAB** chamada, aqui, de `prob`.

Os cinco primeiros campos de `prob` são:

- a dimensão `n` do problema;
- a quantidade `m` de restrições;
- o número `nnzh` de elementos não nulos de uma parte triangular da Hessiana do Lagrangiano;
- o número `nnzj` de elementos não nulos na matriz Jacobiana das restrições do problema;
- o ponto inicial `x` padrão para *solvers* de otimização.

Para acessar as informações contidas nessa estrutura, basta digitar `prob.<campo>` na linha de comando do **MATLAB**, substituindo `<campo>` pelo nome do campo desejado. Ainda para a função **ROSENBR**, o ponto inicial padrão pode ser obtido da seguinte forma:

¹A extensão do arquivo **MEX** varia com a plataforma.

Código 6.3: Obtendo o ponto inicial da função **ROSENB**R.

```
>> prob.x

prob.x =
      -1.2
       1.0
```

É possível alterar os valores de quaisquer campos dessa estrutura, como acontece com qualquer outro objeto desse tipo no **MATLAB**. Para saber mais sobre os campos restantes, acesse [24].

As funcionalidades do **CUTer** vão além disso. O valor da função objetivo e seu gradiente em um ponto **x** de dimensão compatível com o problema carregado podem ser extraídos através da função `cuter_obj`. Entretanto, reforça-se que nossos experimentos não foram beneficiados em nenhum momento com as funcionalidades relacionadas às derivadas da função objetivo.

Código 6.4: Obtenção do valor da função e do gradiente no ponto inicial para o problema **ROSENB**R.

```
>> [f,g] = cuter_obj([1;1])
f =
      0
g =
      0.0
      0.0
```

A variável **f** agora contém o valor da função Rosenbrock no ponto (1, 1) e o vetor **g**, o gradiente dessa função no mesmo ponto.

Além da `cuter_obj`, existe uma série de outras funções disponíveis pelo **CUTer**, destinadas ao cálculo, em determinado ponto, do valor das restrições, dos gradientes, da Hessiana e da Jacobiana do Lagrangiano, etc. A chamada dessas funções é semelhante às de `cuter_setup` e `cuter_obj`. Todas começam com o prefixo `cuter_` para evitar conflitos com outras funções do **MATLAB**. Uma lista completa de funções e suas funcionalidades pode ser obtida em [24].

6.4 Determinando a dimensão dos problemas

Do mesmo modo que, no exemplo anterior, a dimensão da função Rosenbrock veio predeterminada, isto é, em nenhum momento o usuário precisou entrar com a dimensão (dois) do problema, as demais funções da biblioteca já possuem dimensões padrão estabelecidas dentro de seus arquivos **SIF**. Isso significa que para alterar ou consultar a dimensão de um problema, o arquivo correspondente deve ser aberto e, se necessário, editado. Esse procedimento pode ser feito com qualquer editor de texto (incluindo o próprio **MATLAB**) e deve ser realizado com cautela, principalmente se não houver familiaridade com o formato.

O diretório padrão onde os arquivos **SIF** são armazenados é o `$HOME/CUTer/mastsif`. Por *default*, a biblioteca do **CUTer** vem com um arquivo desse tipo para cada problema. Em geral, tais documentos compartilham uma estrutura semelhante que começa com uma breve descrição

e uma referência sobre a origem do problema, seguidas da declaração de suas variáveis e demais caracterizações.

Seguindo a linha do exemplo dado, com a função de Rosenbrock, o Código 6.5 corresponde ao primeiro trecho do arquivo `EXTROSNB.SIF`².

Código 6.5: Trecho inicial de `EXTROSNB.SIF`

```

1 *****
2 * SET UP THE INITIAL DATA *
3 *****
4
5 NAME          EXTROSNB
6
7 *   Problem :
8 *   -----
9
10 *   The extended Rosenbrock function (nonseparable version).
11
12 *   Source: problem 10 in
13 *   Ph.L. Toint,
14 *   "Test problems for partially separable optimization and
15 *   results for the routine PSPMIN",
16 *   Report 83/4, Department of Mathematics, FUNDP (Namur, B),
17 *   1983.
18
19 *   See also Buckley#116. Note that MGH#21 is the separable
20 *   version.
21 *   SIF input: Ph. Toint, Dec 1989.
22
23 *   classification SUR2-AN-V-0
24
25 *   Number of variables
26
27 *IE N          2          $-PARAMETER
28   IE N          5          $-PARAMETER
29 *IE N          10         $-PARAMETER
30 *IE N          20         $-PARAMETER
31 *IE N          100        $-PARAMETER
32 *IE N          1000       $-PARAMETER

```

De acordo com o padrão da linguagem [10], para anular o efeito de uma linha, isto é, para comentá-la, basta que ela comece com o símbolo asterisco (*). Todo o conteúdo da linha será

²O problema `EXTROSNB` é a versão estendida da função de Rosenbrock, enquanto que `ROSENBR` é a original com dimensão fixa e igual a dois.

desprezado durante a leitura do arquivo. Portanto, no Código 6.5, apenas as linhas 5 e 26 serão interpretadas pelo decodificador. Consequentemente, devido à vigésima sexta linha, a dimensão ativa será de $n = 5$.

Dessa forma, para alterar a dimensão do problema do exemplo, basta comentar a linha 26 e descomentar alguma das linhas consecutivas à vigésima-quarta. Caso a dimensão desejada não apareça na lista, é possível inseri-la no final do trecho exibido, desde que se mantenha o padrão de espaçamentos e que a dimensão seja compatível com o problema. É comum que o valor padrão esteja indicado pela marcação de *original value*.

Nem todos os problemas possuem flexibilidade na atribuição da dimensão. Em geral, aqueles que permitem mudanças desse tipo apresentam alguma informação em sua descrição.

6.4.1 Tratando de dimensões múltiplas

Quando é necessário realizar testes com mais de uma dimensão para um mesmo problema, a ideia de alterar o arquivo **SIF** via editor de texto se torna tediosa e pouco produtiva. Uma solução que se propõe neste trabalho e que se mostrou eficaz durante os testes é a duplicação do arquivo que define a função, associada à mudança de nome do mesmo.

Por exemplo, é possível copiar e colar o arquivo **EXTROSNB.SIF** no mesmo diretório, mas com um nome distinto que possa caracterizar a diferença entre eles - digamos, **EXTROSNB20.SIF**. Em seguida, através dos procedimentos explicados anteriormente, deixa-se ativada a dimensão 20 em **EXTROSNB20.SIF** e o problema passa a ser tratado como se fosse diferente do original. Ou seja, é como se o acervo da biblioteca tivesse sido aumentado com um arquivo exclusivo para o problema de Rosenbrock com dimensão 20.

Isso abre diversas possibilidades de dinamização dos testes e também serve para diminuir os riscos de confusão com a dimensão ativa do arquivo, uma vez que ela passa a estar explícita desde o nome do documento.

6.5 Dinâmica dos Testes Computacionais

A dificuldade de adotar a biblioteca do **CUTEr**, em **FORTRAN**, e acessá-la exclusivamente a partir do **MATLAB** pode ser contornada com o uso das ferramentas de interface conforme detalhado nas seções anteriores. Nesta seção, será ilustrada a dinâmica de acesso ao **CUTEr**, apoiada na existência de uma função centralizadora **main.m** e da separação da chamada da função objetivo através de um arquivo **funcf.m**, ao utilizar um *solver* $s \in \mathcal{S}$ para tentar resolver os problemas de \mathcal{P} .

Os procedimentos descritos aqui pretendem ser gerais e independentes do *solver* e do conjunto \mathcal{P} . O papel da rotina principal **main.m** é o de caracterizar e executar os testes computacionais. Nela, são carregadas a lista de problemas-teste, os respectivos pontos iniciais, os níveis de acurácia a serem exigidos e o orçamento computacional disponível. Em seguida, inicia-se um laço que percorre a lista de problemas e submete-os, um a um, ao *solver* em questão.

Para isso, a cada item da lista percorrida, **main.m** aciona o problema correspondente do **CUTEr** através da execução de um comando de sistema análogo ao do Código 6.2. Portanto, uma instrução

contendo o nome do problema na biblioteca é enviada ao console de comandos (no **shell**, por exemplo, no caso das plataformas **UNIX**) para que o problema possa ser ativado.

Ao acionar o *solver*, a função principal *indica funcf.m* como a responsável por *calcular* os dados dos problemas (no caso, o valor da função objetivo em determinado ponto ou conjunto de pontos). Dessa forma, quando necessita do valor da função em algum ponto, o pacote de otimização recorre à *funcf*, pois esta retorna informações sobre o problema que estiver atualmente ativado, através de *cuter_obj* e conforme o Código 6.4.

Após o processo de otimização, passa-se ao item seguinte da lista de problemas e o procedimento se repete até que esta seja esgotada. Os dados retornados durante as otimizações são dinamicamente guardados em arquivos do **MATLAB** para tratamento posterior. A Figura 6.1 mostra um esquema para essa dinâmica.

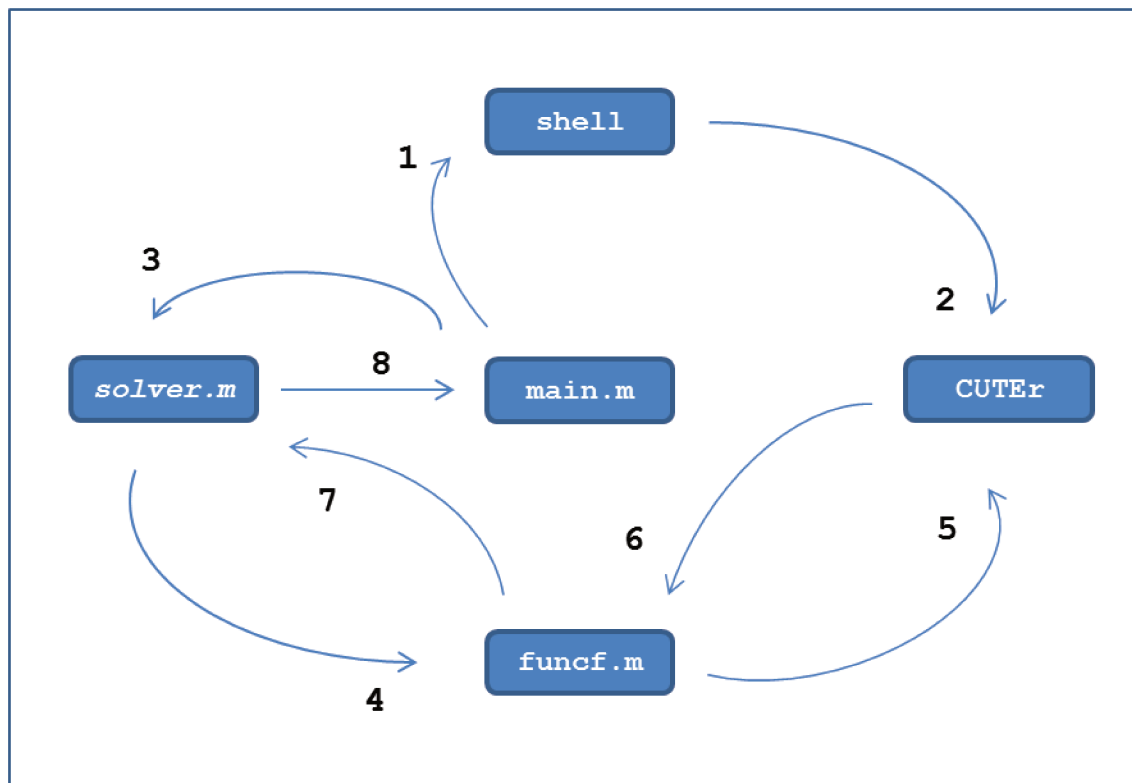


Figura 6.1: Fluxo de chamadas e dados na configuração de testes. A função principal executa um comando no sistema operacional (1) que ativa determinado problema do CUTer (2). Em seguida, aciona o *solver* para resolver o problema de otimização (3). Por sua vez, o resolvidor sistematicamente solicita informações sobre a função objetivo do problema e as consegue através da função intermediadora *funcf* (4 e 7), que obtém os dados direto da biblioteca **CUTer**, estabelecendo comunicação entre **MATLAB** e **FORTRAN** através dos arquivos **MEX** (5 a 6). Após sua execução, o *solver* retorna para a função principal um relatório do processo de otimização e seus resultados.

O Código 6.6 contém uma possível implementação do arquivo *funcf.m*. Em outros contextos,

ele poderia também retornar dados de derivadas de ordem superior da função objetivo e, inclusive, do Lagrangiano do problema [24].

Código 6.6: Arquivo funcf.m

```
1 function [varargout] = funcf( x )
2     global prob;
3     if nargout > 2
4         error('funcf admite no maximo dois argumentos de saida.')
5     end
6     %
7     if nargout == 1
8         % Calcula o valor da funcao objetivo nos m pontos de entrada
9         m = size(x,2);
10        aux = zeros(m,1);
11        for i=1:m
12            aux(i) = cuter_obj(x(:,i));
13        end
14        varargout{1} = aux;
15    end
16    %
17    if nargout == 2
18        % Calculo do gradiente da funcao nos m pontos de entrada
19        m = size(x,2);
20        n = size(x,1);
21        auxf = zeros (m,1);
22        auxg = zeros(n,m);
23        for i=1:m
24            [fun,gra] = cuter_obj(x(:,i));
25            auxf(i) = fun;
26            auxg(:,i) = gra;
27        end
28        varargout{1} = auxf;
29        varargout{2} = auxg;
30    end
31    end % end of funcf
```

Experimentos Computacionais

Os experimentos descritos nesta seção foram realizados em uma máquina com sistema operacional **LINUX** 64 bits e processador **Intel® Core i7 – 2600K**, de 3.40 GHz de frequência e com memória *cache* L3 de 8 Mb. A versão do **MATLAB** utilizada foi a 7.10.0.499 (*release R2010a*) para computadores de 64 bits.

Sob a hipótese de que o custo de avaliar a função objetivo dos problemas tratados seja mais relevante do que os da álgebra linear computacional e do tempo de processamento do código envolvidos, os resultados são apresentados em termos da unidade de medida **número de avaliações de função**.

A fim de comparar o desempenho da implementação **DF0-LP** do Algoritmo 3.1 de Scheinberg e Toint [47], foram escolhidos os métodos **SID-PSM**, representante da classe de busca padrão, e o **Nelder-Mead**, da classe de busca direta. Duas implementações do **SID-PSM** e uma do **Nelder-Mead** foram incorporadas ao conjunto \mathcal{S} de *solvers*, totalizando quatro métodos a serem confrontados.

Os testes consistiram em submeter 57 problemas, explicitados mais adiante, a cada um dos *solvers* de \mathcal{S} , armazenando os dados envolvidos no processo. De acordo com as metodologias de comparação de desempenho adotadas, as informações essenciais sobre os problemas são o decréscimo no valor da função objetivo atingido a cada iteração junto com o esforço gasto para atingi-lo, medido em quantidade de avaliações de função. O orçamento disponibilizado para cada *solver* foi o de 2400 avaliações de função objetivo por problema $p \in \mathcal{P}$.

A seguir, neste capítulo, são feitos o detalhamento do conjunto \mathcal{S} , uma breve apresentação e a configuração dos elementos de \mathcal{P} , assim como informações sobre a definição do orçamento computacional disponível.

7.1 *Solvers* para otimização sem derivadas

A definição do conjunto \mathcal{S} de *solvers* para otimização sem derivada se deu, essencialmente, por questão da facilidade de utilização de seus códigos, todos acessíveis pelo **MATLAB**. Além disso, a escolha de representantes dos métodos de busca padrão e de busca direta enriquece a comparação com a abordagem de região de confiança do **DF0-LP**.

7.1.1 DF0-LP

O algoritmo central deste trabalho, o **DF0-LP**, foi implementado de acordo com as especificações feitas na Seção 4. Os parâmetros de região de confiança do **DF0-LP** foram fixados nos valores $\eta_1 = 0.001$ e $\eta_2 = 0.5$, além de $\gamma_1 = 0.01$, $\gamma_2 = 0.5$ e $\gamma_3 = 2$. O raio Δ_0 da região de confiança inicial foi igualado a 1, conforme sugerido na Seção 17.2 de [11]. O vetor v^0 foi tomado com valor $v^0 = 10x^0$.

Seguindo Custódio e Vicente [17, Seção 8], adotou-se $\Lambda = 100$ nos experimentos. Além disso, considerou-se como *próximo* todo ponto que estivesse a uma distância de até $2\Delta_k$ de x^k , isto é, optou-se por $\beta = 2$. Após a realização de alguns testes preliminares, preferiu-se definir $\mu = 0.25$ e $\theta = 1.25$.

Para impor como único critério de parada o número máximo de avaliações de função, além de se estabelecer μ_f como limitante superior para o número de avaliações de função, zeraram-se as tolerâncias para o raio Δ , para o tamanho do passo s e para norma do gradiente g do modelo.

Com a configuração apresentada, foram submetidos ao **DF0-LP** os problemas-teste de \mathcal{P} , provenientes do **CUTEr**, conforme os procedimentos descritos na Seção 6.5. Os parâmetros de saída armazenados foram o vetor solução e o valor da função objetivo ótimos, além do *array histout*, formado por duas colunas e por tantas linhas quanto o número de iterações realizadas. Da maneira como *histout* foi criado, na iteração k , sua posição $(k, 1)$ contém a quantidade de avaliações de função realizadas até aquela iteração, e a posição $(k, 2)$, o menor valor de função objetivo obtido com tal gasto.

7.1.2 SID-PSM

O *Simplex Derivative in Pattern Search Method* (**SID-PSM**), de Custódio e Vicente [17, 15], é um método de otimização não linear sem derivadas pertencente à classe de métodos de busca padrão. De fato, o **SID-PSM** tem se colocado como um dos algoritmos mais competitivos de sua classe [45, 17]. Em seus passos de busca e pesquisa de direções potencialmente de descida são utilizadas informações das derivadas simplex do problema, além de modelos de norma de Frobenius mínima, estes mais recentemente incorporados à estratégia.

O pacote **SID-PSM 1.2**, de novembro de 2010, é a versão mais recente da implementação feita pelos próprios autores do método, em **MATLAB**, e é capaz de tratar de problemas com ou sem restrições [16]. Na situação geral restrita, as derivadas das restrições devem ser fornecidas pelo usuário. No entanto, apenas as funcionalidades para o caso irrestrito serão necessárias neste trabalho.

Através do arquivo `parameters.m`, o **SID-PSM** permite que diversos ajustes sejam realizados, durante sua configuração. Diferentes formas de definir o conjunto de pesquisa, de atualizar o tamanho do passo e de podar o conjunto de direções permitem a caracterização de múltiplas versões do **SID-PSM**. Em um trabalho recente [8], Cervelin comparou algumas dessas versões e destacou duas dentre elas: uma por sua robustez e outra por sua eficiência. A primeira foi chamada de **SID-PSM1** e esta última, de **SID-PSM2**. Durante as comparações de desempenho, elas serão tratadas como sendo dois *solvers* distintos devido às suas características contrastantes.

A versão **SID-PSM1** é obtida quando os valores 1, 2 e 1 são atribuídos respectivamente às

variáveis `mesh_option`, `pss` e `prunning` do arquivo `parameters.m`. Por sua vez, a versão SID-PSM2 é caracterizada quando os valores 3, 3 e 0 são atribuídos a essas mesmas variáveis, ainda na ordem supracitada.

Além dessas alterações, o arquivo de configuração de parâmetros foi modificado para que o único critério de parada fosse o número máximo μ_f de avaliações de função. Para tanto, foram inibidas, com valor zero, as variáveis `stop_alfa`, `stop_grad`, `stop_iter`, `tol_alfa` e `tol_grad`; acionada, com valor 1, a variável `stop_fevals`; e, finalmente, alterada para receber o valor μ_f , a variável `fevals_max`.

A comunicação do SID-PSM 1.2 com o **CUTEr** é realizada da maneira descrita no Capítulo 6. O arquivo através do qual se aciona a rotina de otimização é o `sid_psm.m`. Seus parâmetros de entrada, além do ponto inicial, consistem de duas *flags*: uma que indica a presença de restrições e outra que controla a quantidade de informação exibida na janela de comando durante a otimização.

Ao término da execução, são retornadas a solução, o valor ótimo associado e o *array histout* - este último, com a mesma estrutura e funcionamento que o do DFO-LP.

7.1.3 Nelder-Mead

O método de Nelder-Mead, também conhecido como método simplex ou método da ameoba, é um algoritmo de otimização não linear, irrestrita e sem uso de derivadas proposto na década de 60 por John Nelder e Roger Mead [38]. Ele pertence à classe de métodos de busca direta [30] e, essencialmente, realiza uma sequência de transformações no simplex da iteração corrente, na tentativa de obter, por comparação, um decréscimo no valor da função objetivo quando esta é avaliada nos vértices do simplex. Caso a redução desejada não seja atingida, diminui-se o tamanho do simplex na iteração seguinte.

Tipicamente, cada iteração requer apenas uma ou duas avaliações de função, o que é pouco se comparado a muitos outros métodos de busca direta. É um algoritmo relativamente simples e de fácil aplicação, o que também justifica sua popularidade em diversas áreas da ciência.

Apesar de se tratar de uma heurística clássica, é possível que sua convergência seja decretada em um ponto não estacionário [35]. Para evitar esse comportamento indesejado, isto é, para garantir sua convergência global, o método original precisaria ser melhorado (ver, p.ex., [54]).

Desde sua criação, o método sofreu algumas modificações e foi programado em diferentes linguagens. A versão em **MATLAB** utilizada neste trabalho está implementada nativamente sob o nome de `fminsearch.m` [31] e recebe como parâmetros de entrada a função objetivo, o ponto inicial e uma estrutura que contém a configuração dos parâmetros utilizados durante a execução da rotina, respectivamente. A lista de parâmetros de saída contém a solução e o valor ótimos encontrados para o problema, além de duas outras estruturas que armazenam informações relacionadas ao processo de otimização (número de avaliações de função, critério de parada satisfeito etc.). O `fminsearch` corresponde à versão original, proposta por Nelder e Mead, com um simplex inicial adequado, que inibe o contra-exemplo de McKinnon [35].

Ainda, o *release* **R2010b** permite o acesso ao código fonte de `fminsearch`. Com isso, foi possível modificá-lo para que também retornasse como parâmetro de saída um *array histout* equivalente aos do SID-PSM e do DFO-LP, com a toda a precisão do *software* de cálculos numéricos. Ressalta-se, entretanto, que tais modificações foram cautelosamente realizadas e não houve nenhuma alteração

na lógica do algoritmo.

O seguinte trecho de código **MATLAB** é responsável por configurar o conjunto de parâmetros utilizado pelo `fminsearch` durante a otimização.

```
% muf definido anteriormente
>> options = optimset('TolX',0,
                      'TolFun',0,
                      'MaxIter', Inf,
                      'MaxFunEvals', muf);
```

Os parâmetros `TolX` e `TolFun` representam, respectivamente, a tolerância na variável (multi-dimensional) do problema e na função objetivo relacionada. Ambas foram zeradas com a ideia de inibir os testes de parada associados a elas. Para que o número de iterações não fosse um critério efetivo de terminação do algoritmo, o parâmetro `MaxIter` foi associado a `Inf`, representação numérica da linguagem para o infinito (positivo). O número máximo de avaliações de função recebeu o valor `muf`, versão computacional de μ_f . Assim, na prática, o único critério de parada ativo foi o do número máximo μ_f de avaliações de função objetivo.

7.2 Problemas teste

A fim de analisar o desempenho dos *solvers* comparados neste trabalho, optou-se pelo uso do **CUTEr** (*Constrained and Unconstrained Testing Environment, revisited*) [22], um dos mais tradicionais ambientes de testes da área de otimização e álgebra linear computacional¹.

O **CUTEr** é composto por um conjunto de problemas teste bastante abrangente. Esse conjunto possui representantes de diversas classes, desde problemas de pequeno porte, irrestritos e diferenciáveis, passando por problemas restritos de igualdade e desigualdade, densos e esparsos, e alcançando, inclusive, sistemas de equações não-lineares e problemas de redes.

Os problemas componentes do **CUTEr** estão construídos sob o formato **SIF** (*Standard Input Format* [10]), um padrão de representação de problemas de programação não-linear criado para facilitar a modelagem matemática e computacional. Através de um decodificador, tais arquivos são convertidos para a linguagem **FORTRAN** e, assim, o ambiente **CUTEr** é capaz de prover ferramentas de interface com diversos programas e pacotes de otimização matemática, incluindo o tradicional **MATLAB**.

Para o presente trabalho, tomou-se como base a seleção de 60 problemas irrestritos feita em [21], cujos nomes e dimensões estão explicitados na Tabela 7.2. Dois deles² já não fazem parte da coleção e não se encontram disponíveis na biblioteca mantida pelos autores do ambiente de testes [22]. Adicionalmente, um dos demais problemas da lista possui variáveis limitadas³ e precisou ser descartado. Dessa forma, o conjunto \mathcal{P} de testes foi definitivamente determinado com os 57 problemas restantes.

¹Já existe uma versão posterior ao **CUTEr**, chamada CUTEst [25]. Entretanto, na fase de desenvolvimento dos experimentos desta dissertação, o CUTEst ainda não estava tão solidificado e estabelecido como o **CUTEr**.

²**SINGULAR** e **NASTY**

³**CHEBYQAD**

dimensão	2	3	4	5	6	8	10	11	15
ocorrências	16	11	4	1	2	4	12	1	16

Tabela 7.1: Distribuição das dimensões em \mathcal{P}

Para cada problema p em \mathcal{P} , sua dimensão n_p pertence ao intervalo $[2, 15]$. Mais especificamente,

$$n_p \in \{2, 3, 4, 5, 6, 8, 10, 11, 15\}.$$

A distribuição das dimensões n_p dos elementos de \mathcal{P} pode ser conferida na Tabela 7.1. Tal distribuição possui, portanto, média aproximada em 6.2 e mediana igual a 4.

Conforme visto na Capítulo 5, a metodologia de comparação de desempenho adotada aqui não se baseia em valores ótimos de função objetivo obtidos computacionalmente com o uso de derivadas. Antes, toma como referência apenas os menores valores ótimos obtidos sem derivada com os *solvers* envolvidos na comparação.

Ainda assim, para que se possa confrontar essas duas abordagens mais adiante, reproduz-se na Tabela 7.2 o conjunto de valores ótimos dos problemas de P apresentados em [21]. De acordo com essa publicação, tais valores foram obtidos com o pacote **KNITRO** [56], que se utiliza de derivadas de até segunda ordem durante o processo de otimização e realiza cálculos com acurácia de 14 dígitos significativos.

Orçamento Computacional

O orçamento computacional foi definido de forma a garantir que cada método tivesse à sua disposição um número de avaliações de função equivalente ao cálculo de, pelo menos, 150 gradientes simplex por problema. Para calcular um gradiente simplex no \mathbb{R}^n são necessárias $n + 1$ avaliações de função. Portanto, como a maior dimensão presente nos problemas de \mathcal{P} é 15, μ_f foi fixado em 2400.

nome	n	f_L	nome	n	f_L
ALLINITU	4	5.74438491032034E+00	FREUROTH	10	1.01406407257452E+03
ARGLINB	10	4.63414634146338E+00	GENHUMPS	5	9.31205762089110E-33
ARGLINC	8	6.13513513513513E+00	GULF	3	5.70816776659866E-29
ARWHEAD	15	5.32907051820075E-15	HAIRY	2	2.00000000000000E+01
BARD	3	8.21487730657899E-03	HATFLDD	3	6.61511391864778E-08
BDQRTIC	10	1.82811617535935E+01	HATFLDE	3	4.43440070723924E-07
BEALE	2	1.03537993810258E-30	HELIX	3	1.81767515239766E-28
BIGGS3	3	3.49751055496115E-25	HILBERTA	10	1.51145573593758E-20
BIGGS6	6	5.49981608181981E-16	HIMMELBF	4	3.18571748791125E+02
BOX2	2	3.32822794031215E-23	HIMMELBG	2	1.17043537660229E-27
BOX3	3	1.85236429640516E-20	JENSMP	2	1.24362182355615E+02
BRKMCC	2	1.69042679196450E-01	KOWOSB	4	3.07505603849238E-04
BROWNAL	10	1.49563496755546E-16	MANCINO	10	1.24143266331958E-19
BROWNDEN	4	8.58222016263563E+04	MARATOSB	2	-1.00000006249999E+00
CHEBYQAD	8	1.75843686283896E-03	MEXHAT	2	-4.01000000000000E-02
CHROSEN	15	1.21589148855346E-19	MOREBV	10	1.85746736253704E-24
CRAGGLVY	10	1.88656589666311E+00	NASTY	2	1.53409170790554E-72
CUBE	2	5.37959996529976E-25	OSBORNEB	11	4.01377362935478E-02
DENSCHND	3	2.15818302178292E-04	PALMER1C	8	9.75979912629838E-02
DENSCHNE	3	1.29096866601748E-18	PALMER3C	8	1.95376385131058E-02
DENSCHNF	2	6.51324621983021E-22	PALMER5C	6	2.12808666605511E+00
DIXMAANC	15	1.00000000000000E+00	PALMER8C	8	1.59768063470262E-01
DIXMAANG	15	1.00000000000000E+00	POWER	10	6.03971630559837E-31
DIXMAANI	15	1.00000000000000E+00	ROSENBR	2	3.74397564313947E-21
DIXMAANK	15	1.00000000000000E+00	SINEVAL	2	7.09027697800298E-20
DIXON3DQ	10	2.95822839457879E-31	SINGULAR	4	6.66638187151797E-12
DQDRTIC	10	5.91645678915759E-29	SISSER	2	1.06051492721772E-12
ENGVAL1	2	0.00000000000000E+00	VARDIM	10	1.59507305257139E-26
ENGVAL2	3	0.00000000000000E+00	YFLTU	3	6.66972048929030E-13
EXPFIT	2	2.40510593999058E-01	ZANGWIL2	2	-1.82000000000000E+01

Tabela 7.2: Tabela dos valores ótimos dos problemas-teste de Fasano, Morales e Nocedal [21].

Análise de Resultados

A seguir, são apresentados os resultados dos experimentos computacionais descritos no Capítulo 7. O presente capítulo pode ser dividido em duas partes ou blocos principais. Na primeira, são abordadas a sensibilidade da implementação **DF0-LP** ao parâmetro ϵ_0 e algumas implicações da existência do passo de criticalidade no Algoritmo 3.1. Na segunda parte, é feita uma comparação entre o desempenho dos *solvers* de \mathcal{S} , seguindo a metodologia do Capítulo 5. Para cada uma delas, o cenário de teste foi configurado de maneira análoga e consiste, essencialmente, na aplicação de um conjunto de *solvers* na resolução de um conjunto de problemas-teste. Por questão de clareza, o conjunto de *solvers* da primeira parte foi nomeado de \mathcal{S}_{ϵ_0} , enquanto que o da segunda foi mantido como \mathcal{S} . Os conjuntos de problemas-teste coincidiram em ambas as partes e, portanto, foram tratados apenas como conjunto \mathcal{P} .

O termo **referenciação externa** foi adotado para aludir ao emprego de f_L^p (cf. Seção 5.1) como valor extraído da Tabela 7.2 originária do trabalho de Fasano *et al.* [21] com o **KNITRO**. Além desse termo, menções como “vertente de Marazzi *et al.*” [33] e suas derivações também foram utilizadas. Quando f_L^p era dado em função do desempenho dos próprios *solvers* que estavam sendo testados nos problemas selecionados, optou-se pelo termo **referenciação interna** ou, alternativamente, “vertente de Moré e Wild”.

Em todos os casos testados, a terminação das execuções dos *solvers* se baseou exclusivamente no número de avaliações de função. Em conformidade com o Capítulo 7, somente quando o orçamento computacional era esgotado se determinava o fim da execução. Em raras ocorrências, a parada foi decretada antes do uso de todas as avaliações de função disponíveis. Casos como este foram resultados, por exemplo, da anulação da norma do gradiente do modelo, o que força a parada ainda que as tolerâncias tenham sido ajustadas para zero.

Durante a realização dos testes, a codificação **SIF** do problema **HILBERTA** apresentou inconsistências numéricas nos valores retornados pelo **CUTEr**. Para contorná-las, tal função foi substituída por uma implementação própria, baseada no trabalho de Schittkowski [48] referenciado no próprio arquivo **SIF** do problema.

8.0.1 Tratamento *a posteriori* dos dados coletados

O procedimento descrito nesta seção foi aplicado aos dois blocos de testes realizados e, portanto, será relatado de maneira genérica que abranja ambas as situações.

Dentre o vasto conjunto de dados gerado durante os testes estão as estruturas **histout** apresentadas no Capítulo 7. Para cada par (p, s) um *array* **histout** foi obtido como produto da resolução do problema $p \in \mathcal{P}$ pelo *solver* $s \in \mathcal{S}^1$. A partir delas foram construídos vetores $h^{p,s} \in \mathbb{R}^{\mu_f}$ para cada par (p, s) . Da maneira como foi concebida, a k -ésima componente desse vetor pode ser escrita como [37]:

$$h_k^{p,s} = \min \{f_p(x^j) : 0 \leq j \leq k\}, \quad (8.0.1)$$

onde f_p é a função objetivo do problema p e x^j é o ponto corrente da iteração j . Dessa forma, $h_k^{p,s}$ é interpretado como o melhor valor de função obtido com o *solver* s após k avaliações de função.

Observe que, à diferença de **histout**, cujo número de linhas é o número de iterações realizadas, a quantidade de linhas de $h^{p,s}$ é igual ao orçamento μ_f (ou seja, 2400). Como algumas iterações podem consumir mais do que uma única avaliação de função, a quantidade de iterações, em geral, não corresponde ao tanto de avaliações de função gasto. Nos casos em que o algoritmo foi terminado precocemente (e, portanto, com sobra de orçamento), o último valor de função obtido foi reproduzido nas linhas restantes de $h^{p,s}$ até seu completo preenchimento.

Em seguida, todos os vetores $h^{p,s}$ foram reunidos em uma só estrutura tridimensional H , com $H \in \mathbb{R}^{\mu_f \times n_p \times n_s}$, cuja componente (i, p, s) satisfaz:

$$H_{ips} = h_i^{p,s}, \quad (8.0.2)$$

para $i = 1, \dots, \mu_f$, $p = 1, \dots, n_p$ e $s = 1, \dots, n_s$.

Níveis de acurácia

Os níveis de acurácia escolhidos para construir os perfis de desempenho e perfis informativos para cada bloco de experimentos foram $\tau = 10^{-k}$, com $k \in \{1, 3, 5, 7\}$. Com esses valores, buscava-se conseguir focar as análises no comportamento a curto-prazo dos algoritmos para com os problemas de \mathcal{P}

Moré e Wild, em suas análises, destacam que esses níveis não são tão exigentes quanto os testes de convergência baseados na norma do gradiente da função. Por exemplo, se $\tau = 10^{-5}$ e $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é uma função duplamente continuamente diferenciável em uma vizinhança de um minimizador x^* , bem escalada no sentido de que $f(x_0) = 1$ e $f(x^*) = 0$, e com Hessiana $\nabla^2 f(x^*)$ definida positiva, então, para uma norma $\|\cdot\|_*$ apropriada,

$$\|\nabla f(x)\|_* \leq 4.5 \cdot 10^{-3},$$

para qualquer x nessa vizinhança. No entanto, essa diferença é aceitável e natural, uma vez que não se podem esperar critérios que lhes sirvam, ao mesmo tempo, às ambições da otimização com e da sem derivadas.

¹Ou $s \in \mathcal{S}_{\epsilon_0}$.

8.1 Primeiro bloco de testes: o parâmetro ϵ_0

Com as três escolhas para o parâmetro ϵ_0 , tomadas com relação à norma do gradiente do modelo inicial, foram definidas as três versões do DF0-LP:

- DF0-LP1: $\epsilon_0 = 25\%$ de $\|g^0\|_2$;
- DF0-LP2: $\epsilon_0 = 50\%$ de $\|g^0\|_2$;
- DF0-LP3: $\epsilon_0 = 75\%$ de $\|g^0\|_2$.

Essas versões foram testadas como se fossem *solvers* diferentes e independentes compondo um conjunto S_{ϵ_0} de métodos. Seus perfis informativos são apresentados nas Figuras 8.1 e 8.2, nas quais a referenciação é externa e interna, respectivamente. Nos dois conjuntos de gráficos, é possível observar uma grande semelhança de desempenho entre as versões do DF0-LP. Ao contrário do que se poderia esperar, aparentemente não faz muita diferença facilitar ou dificultar a primeira troca de pontos designada exclusivamente a obter um conjunto Λ -posicionado, dentro do Passo 1 do algoritmo; não, pelo menos, para os valores de ϵ_0 e para o conjunto \mathcal{P} adotados aqui.

A única exceção, ainda que branda, é a do Gráfico 8.1d. De acordo com ela, caso o usuário disponha em seu orçamento algo em torno de 80 a 100 gradientes simplex e deseje um nível de acurácia $\tau = 10^{-7}$, conseguirá resolver alguns problemas a mais se adotar o DF0-LP1. Nesse gráfico, o cenário de desempenho se estabilizou com a resolução de aproximadamente 93% dos problemas para cada um dos *solvers* de S_{ϵ_0} , por volta do equivalente ao gasto com 160 gradientes simplex.

Na Figura 8.2d, por sua vez, a estabilidade foi alcançada com um gasto cinco vezes maior, mas com a vantagem de que, nesse cenário, 100% dos problemas foram decretados como resolvidos. Entretanto, destaca-se aqui que tal robustez advém da semelhança de desempenho das versões testadas. Estas atingem valores ótimos muito parecidos para todo o conjunto de problemas e, portanto, como a referência é interna, decreta-se que 100% dos problemas foram resolvidos sem que isso, de fato, represente um desempenho exemplar para qualquer um dos *solvers*. Portanto, numa comparação em que os *solvers* tenham desempenhos ruins e suficientemente semelhantes, a referenciação interna pode conduzir a conclusões precipitadas sobre a robustez dos métodos.

Nestes casos, o perfil informativo se presta mais a acompanhar a rapidez da evolução rumo ao valor ótimo, em termos do número de avaliações de função gasto, para o conjunto \mathcal{P} . A Figura 8.2d mostra, por exemplo, que com o orçamento equivalente ao cálculo de menos de 170 gradientes simplex, aproximadamente 98% dos problemas já haviam atingido seu melhor patamar possível dentro do número máximo de avaliações previsto e de acordo com a acurácia $\tau = 10^{-7}$.

Por sua vez, os perfis de desempenho exibidos nas Figuras 8.3 e 8.4 realçam uma pequena diferença entre os efeitos dos valores atribuídos a ϵ_0 , principalmente com o aumento do nível de acurácia exigido, isto é, com a diminuição de τ . Para $\tau = 10^{-1}$ e $\tau = 10^{-3}$, tanto nos Gráficos 8.3a e 8.3b como nos correspondentes 8.4a e 8.4b, a diferença entre eles é praticamente inexistente, uma vez que o alcance do eixo da taxa de desempenho não chega a 1.2. Isso significa que, apenas para não mais do que uma pequena porcentagem de problemas, o orçamento gasto foi maior do que o melhor resultado obtido dentre todos *solvers* de \mathcal{S} e que tal diferença não chegou aos 20% desse melhor valor. Entretanto, nos outros quadros dessas duas figuras, pode-se notar que o DF0-LP1 exibe robustez igual e eficiência sempre superior aos demais, nos casos aqui abordados.

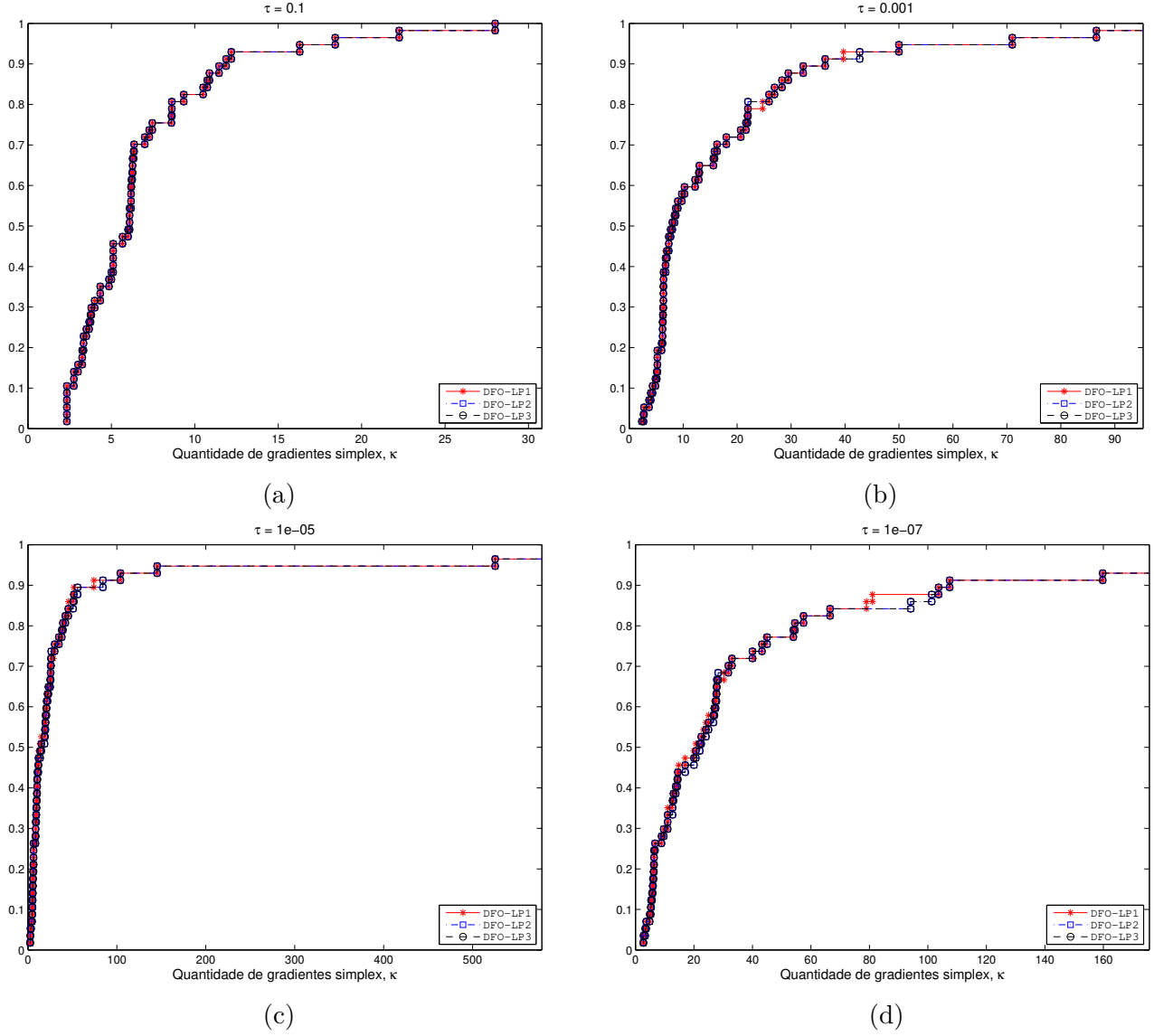


Figura 8.1: Perfis informativos das três versões de DFO-LP, diferenciadas pelo valor de ϵ_0 , para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p extraída da Tabela 7.2 de *Fasano et al.* Eixo horizontal em unidade equivalente ao número de avaliações de função necessário para o cálculo de um gradiente simplex, κ .

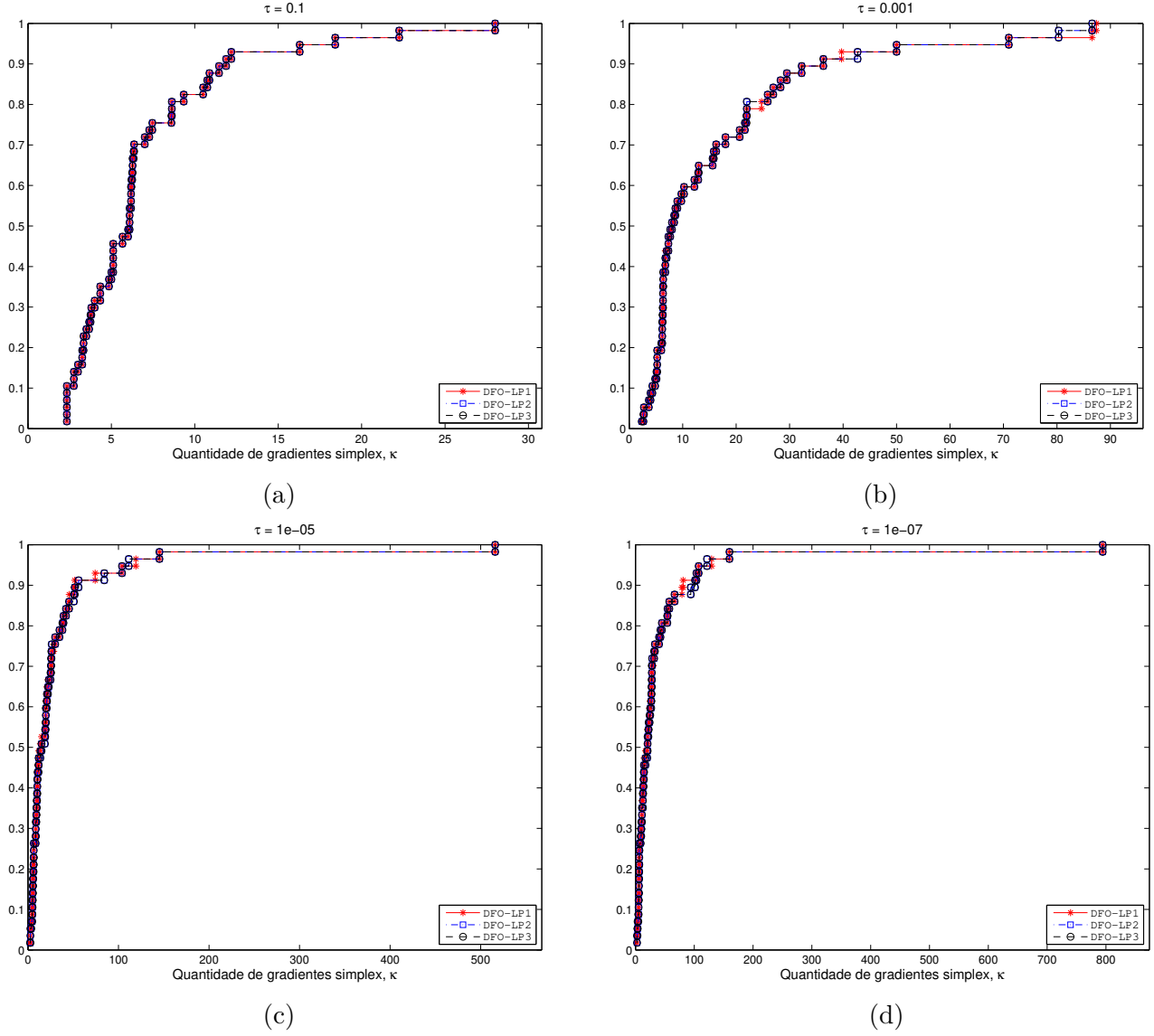


Figura 8.2: Perfis informativos das três versões de DF0-LP, diferenciadas pelo valor de ϵ_0 , para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p obtida como o menor valor de $h_{2400}^{p,s}$ (cf. (8.0.1)) dentre os *solvers* de \mathcal{S} . Eixo horizontal em unidade equivalente ao número de avaliações de função necessário para o cálculo de um gradiente simplex.

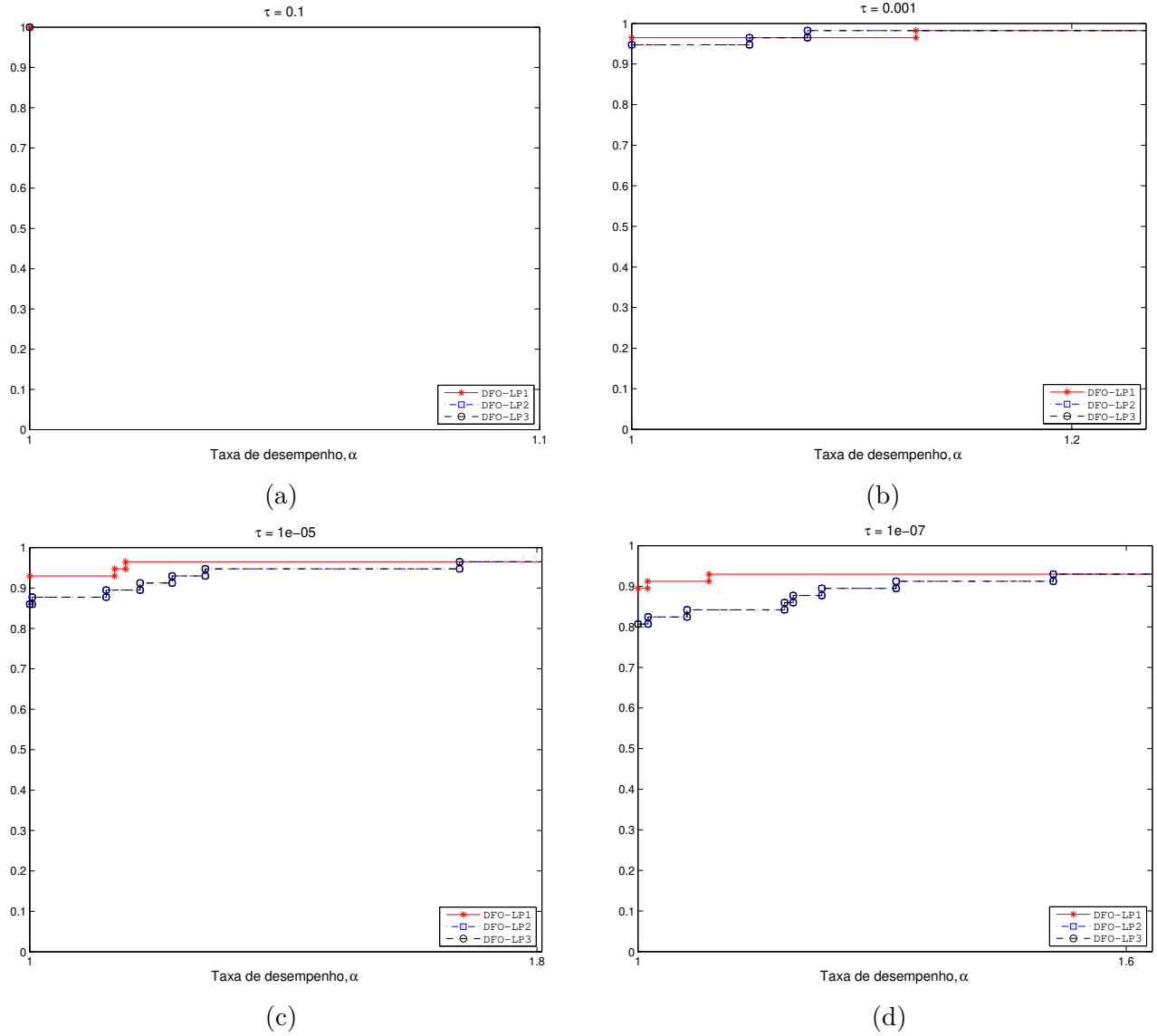


Figura 8.3: Perfis de desempenho das três versões de DFO-LP, diferenciadas pelo valor de ϵ_0 , para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p extraída da Tabela 7.2 de *Fasano et al.* O eixo horizontal representa a taxa de desempenho relacionada à medida “número de avaliações de função”.

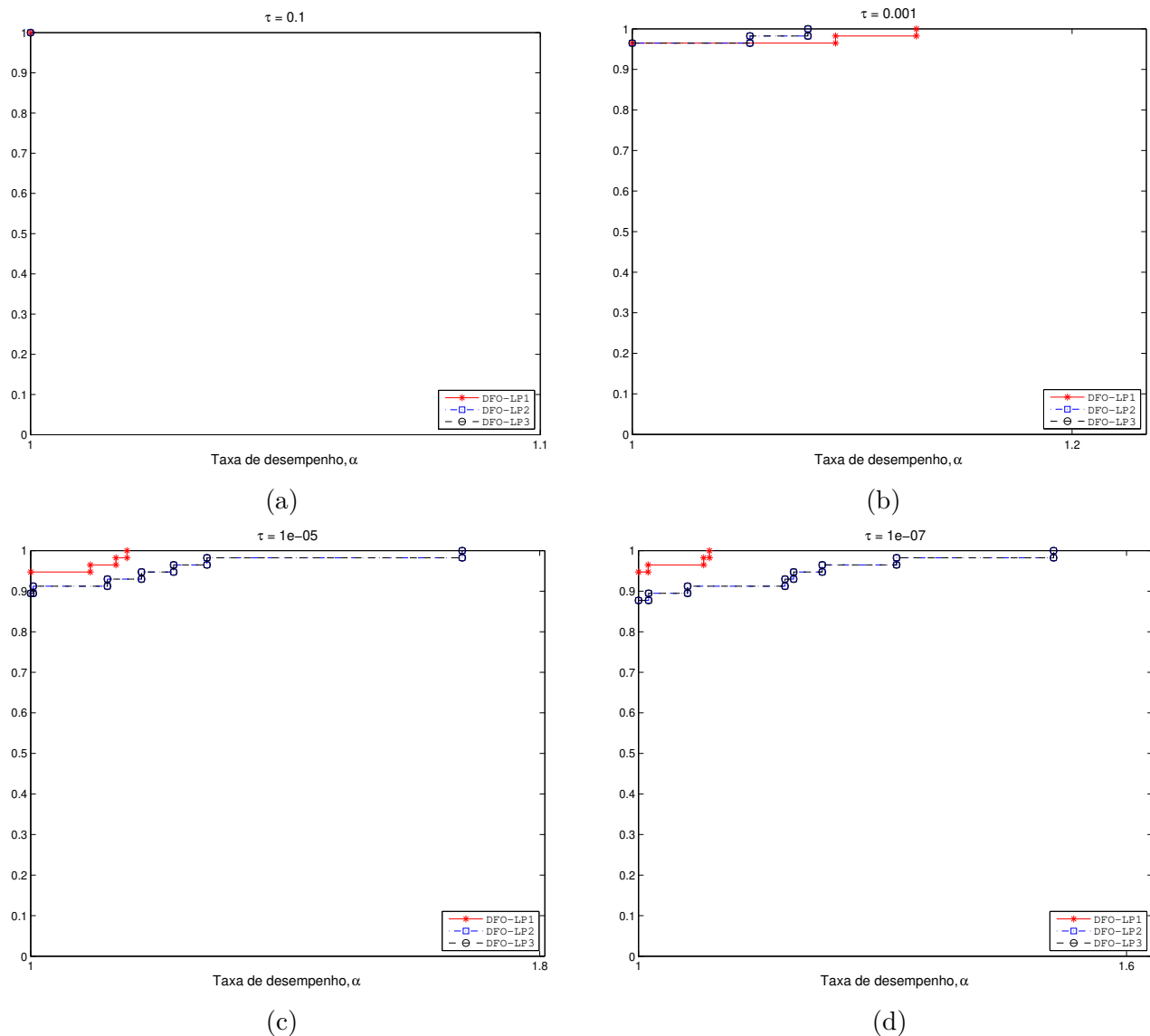


Figura 8.4: Perfis de desempenho das três versões de DFO-LP, diferenciadas pelo valor de ϵ_0 , para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p obtida como o menor valor de $h_{2400}^{p,s}$ (cf. (8.0.1)) dentre os *solvers* de \mathcal{S} . O eixo horizontal representa a taxa de desempenho relacionada à medida “número de avaliações de função”.

De um modo geral, pode-se dizer que a robustez do método DF0-LP não é consideravelmente sensível ao valor do parâmetro ϵ_0 . Ainda assim, a eficiência do DF0-LP1 se sobressai às demais, por mais que comedido, quanto maior o nível de acurácia requisitado. Dessa forma, em sendo necessário optar por um valor para o parâmetro ϵ_0 a partir dos testes apresentados aqui, a decisão mais coerente com a metodologia adotada é a escolha do DF0-LP1. Portanto, ela será usada na comparação com os demais *solvers* do conjunto \mathcal{S} durante a realização do segundo bloco de testes.

8.1.1 Os efeitos dos testes de criticalidade no orçamento

Com um papel de melhora direta na geometria dos pontos amostrais, o Passo 1 do algoritmo proposto por Scheinberg e Toint [47] é indispensável para a demonstração de convergência global do algoritmo. Seus autores advogam que, apesar desse passo não poder ser completamente eliminado, ele desempenharia um papel importante apenas num estágio mais avançado da execução do Algoritmo 3.1. De fato, a preocupação com o passo geométrico se justifica porque, essencialmente, ela envolve a troca de pontos visando a melhora do posicionamento do conjunto amostral e, portanto, pode representar um gasto inviável em avaliações de função, comprometendo o desempenho geral do método.

Visando monitorar os efeitos do teste de criticalidade, durante a execução do primeiro bloco de testes com os *solvers* de \mathcal{S}_{ϵ_0} , foram armazenadas as seguintes informações:

- quantidade de entradas no Passo 1 com construção ou certificação de modelo Λ -posicionado;
- gasto, em avaliações de função, na construção do modelo Λ -posicionado;
- número de repetições de laços *while* (reincidências no **Passo 1b**);
- gasto, em avaliações de função, por repetição do laço de *while*.

Os resultados mostram que, dos 57 problemas de \mathcal{P} , 14 jamais satisfizeram o critério $\|g^k\| \leq \epsilon_0$ de entrada no **Passo 1b** de 3.1, mesmo para a versão DF0-LP3. Esse critério, quando satisfeito, dava acesso ao processo que certificava se o conjunto de pontos corrente era Λ -posicionado ou o modificava para que passasse a sê-lo. Portanto, para aproximadamente 25% dos problemas, a existência do teste de criticalidade foi numérica e orçamentariamente indiferente nos experimentos deste trabalho. Como resultado, as sequências de pontos geradas por DF0-LP1, DF0-LP2 e DF0-LP3 foram iguais para esses problemas.

A Figura 8.5 ilustra o decréscimo percentual do valor da função objetivo com relação à quantidade de avaliações de função gasta para os 14 problemas que não acionaram o teste de criticalidade, além de fornecer os nomes e as dimensões desse subconjunto de \mathcal{P} . Note que para a grande maioria dos problemas, a queda no valor da função acontece bruscamente e já nas primeiras avaliações de função, tendo em vista as 2400 avaliações disponíveis. De fato, para 8 desses problemas, o último sucesso ocorreu antes ou durante a décima iteração, já alcançando o valor ótimo que o DF0-LP1 foi capaz de atingir para essas funções objetivo. Vale alertar que, ainda nessa figura, as curvas de **ARGLINC**, **PALMER1C**, **PALMER3C** e **PALMER8C** acabaram se sobrepondo umas às outras.

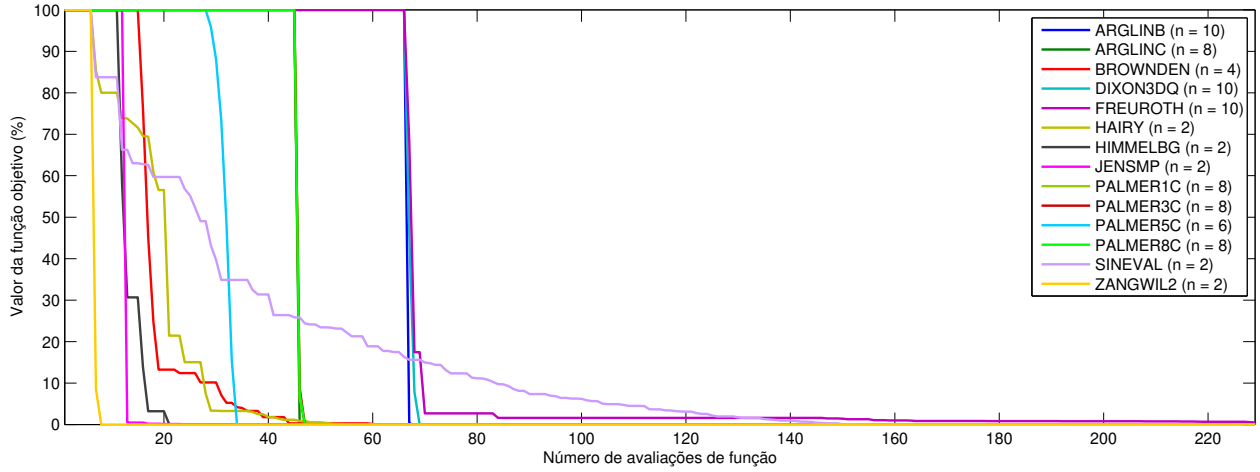
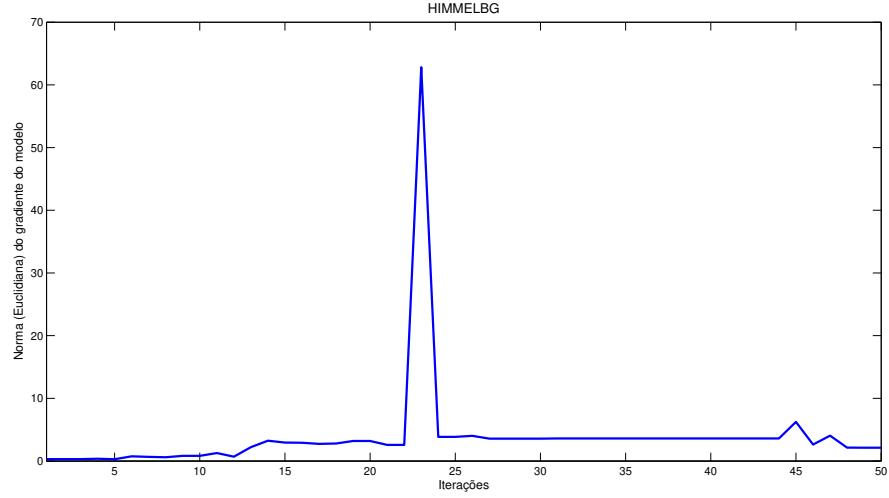


Figura 8.5: Processo evolutivo de minimização da função objetivo dos 14 problemas que jamais satisfizeram o Critério $\|g^k\| \leq \epsilon_0$.

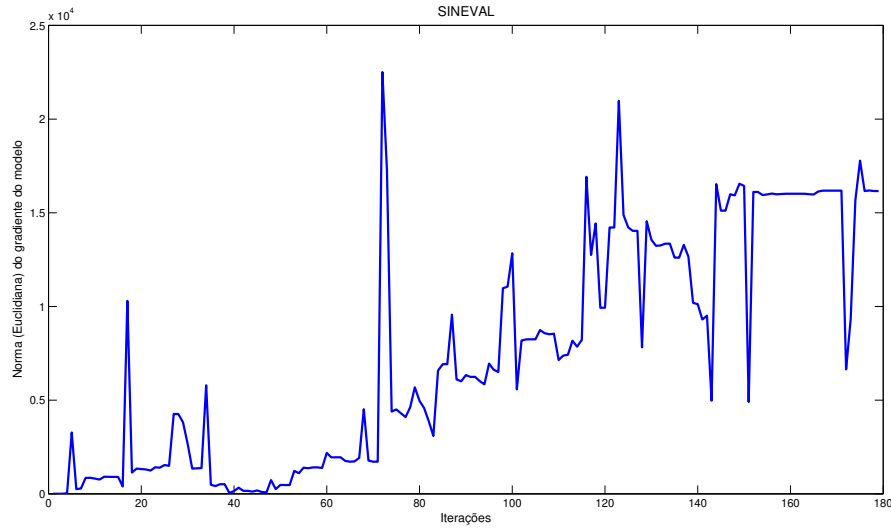
Dentre os demais 43 problemas, 9 não apresentaram nenhum gasto de avaliação de função no Passo 1 de nenhum dos *solvers* de \mathcal{S}_{ϵ_0} , o que significa que, apesar de terem satisfeito o critério de entrada no Passo 1b para pelo menos uma das três versões de DF0-LP, o conjunto de pontos já era Λ -posicionado e, portanto, foi apenas certificado como tal. Devido à maneira com que o Passo 1 foi implementado, cada vez que esse critério era satisfeito, o modelo e os polinômios de Lagrange eram reestruturados e representados em torno de um outro ponto, o corrente. Esses cálculos e os erros intrínsecos a eles causaram uma leve mudança na sequência de pontos gerada para esses 9 problemas. Ainda assim, a maior diferença (absoluta) entre os valores ótimos das três versões do DF0-LP para um mesmo problema foi da ordem de 10^{-13} .

A fim de ilustrar dois comportamentos da norma do gradiente do modelo no decorrer das iterações iniciais até a ocorrência do último sucesso, a Figura 8.6 exibe tal histórico para as funções **HIMMELBG** e **SINEVAL**. Como se pode notar nessas iterações, o valor da norma do gradiente do modelo se manteve mais alto que seu estado inicial. O último sucesso da função **SINEVAL** ocorreu num momento em que a norma do gradiente do modelo era da ordem de $1.6 \cdot 10^4$. Apesar disso, esse problema foi decretado como resolvido pelo DF0-LP1 em todas os níveis de acurácia e para os dois tipos de referenciação.

Por fim, restaram 34 problemas cuja resolução demandou avaliações de função extras em decorrência do teste de criticalidade, para um ou mais valores de ϵ_0 . Ratificando a indiferença observada nos resultados apresentados anteriormente neste capítulo, esse gasto extra foi idêntico, problema a problema, para os casos $\epsilon_0 = 0.5\|g^0\|$ e $\epsilon_0 = 0.75\|g^0\|$. A Figura 8.7 exibe um gráfico que quantifica esses gastos exclusivos do Passo 1 de maneira que leva em consideração a dimensão do problema, utilizando como unidade o equivalente ao necessário para se construir um gradiente simplex, isto é, em múltiplos de $n + 1$, onde n é a dimensão do problema associado. Pode-se observar que o maior gasto ocorre para o problema **ARWHEAD** ($n = 15$) com o uso do DF0-LP1, equivalendo a aproximadamente 16 gradientes simplex. De um modo geral, portanto, o custo dos



(a)



(b)

Figura 8.6: Norma do gradiente do modelo a cada iteração durante a resolução dos problemas **HIMMELBG** e **SINEVAL** pelo DF0-LP1 até a última iteração de sucesso.

testes de criticalidade não ultrapassou 11% do orçamento μ_f de 150 gradientes simplex. De certa forma, trata-se de um resultado positivo, visto que uma das preocupações com o algoritmo residia no custo de, potencialmente, trocar todos os pontos do conjunto Y durante o Passo 1 na busca pelo Λ -posicionamento. Entretanto, não fica evidente nenhuma relação direta entre os custos, as dimensões e os valores de ϵ_0 para os problemas afetados.

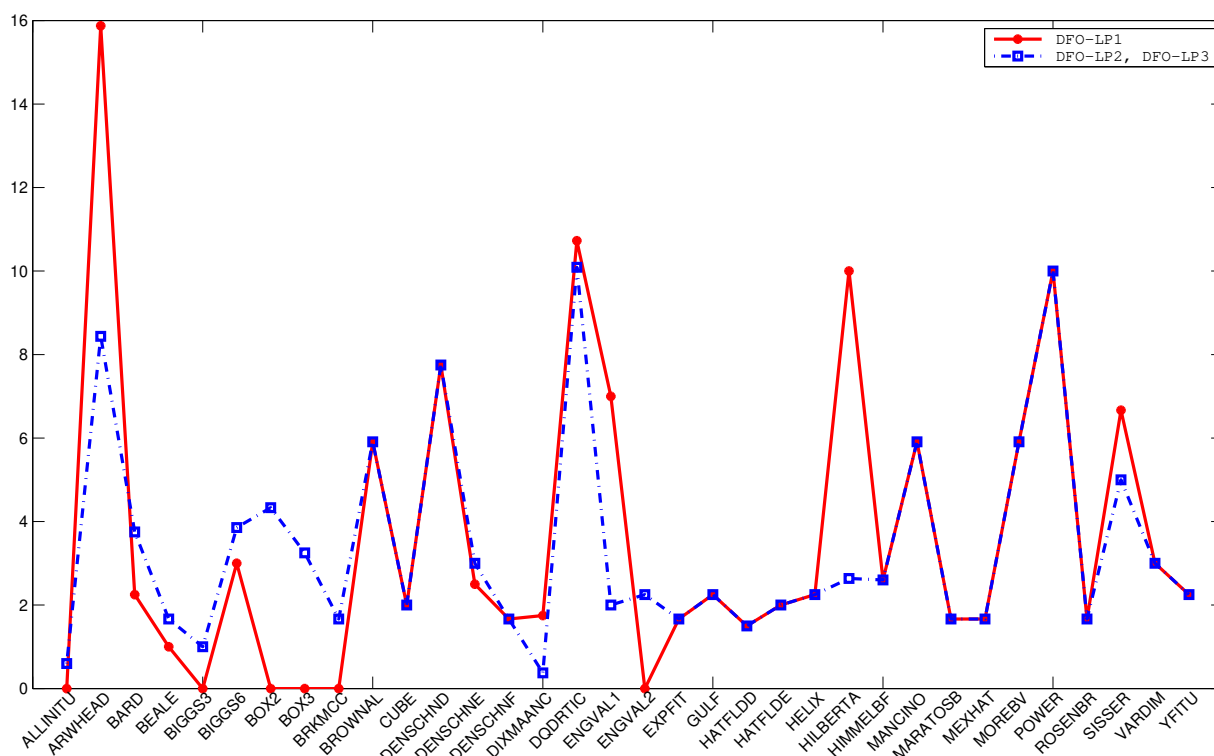


Figura 8.7: Gasto das três versões do DFO-LP exclusivamente com as repetições do Passo 1 durante a resolução dos 34 problemas para os quais em algum momento foi necessário realizar pelo menos um troca de pontos para garantir a construção de um modelo Λ -posicionado. Unidade de medida equivalente ao cálculo de um gradiente simplex para o problema associado (múltiplos de $n + 1$).

Para uma maior especificação a respeito do gasto total de cada um dos elementos de \mathcal{S}_{ϵ_0} exclusivamente com o teste de criticalidade, consulte a Tabela 8.1.

8.2 Segundo bloco de testes: a comparação dos *solvers* de \mathcal{S}

Com a definição do DFO-LP1 como código representante do algoritmo teórico originalmente proposto em [47], os experimentos de comparação com os demais *solvers* de \mathcal{S} puderam, finalmente, ser realizados. Seus resultados são apresentados a seguir, seguindo a mesma metodologia de comparação do primeiro bloco, através de perfis informativos e de desempenho.

problema	DF0-LP1	DF0-LP2	DF0-LP3	problema	DF0-LP1	DF0-LP2	DF0-LP3
ALLINITU	0.0	0.6	0.6	FREUROTH	0.0	0.0	0.0
ARGLINB	0.0	0.0	0.0	GENHUMPS	0.0	0.0	0.0
ARGLINC	0.0	0.0	0.0	GULF	2.3	2.3	2.3
ARWHEAD	15.9	8.4	8.4	HAIRY	0.0	0.0	0.0
BARD	2.3	3.8	3.8	HATFLDD	1.5	1.5	1.5
BDQRTIC	0.0	0.0	0.0	HATFLDE	2.0	2.0	2.0
BEALE	1.0	1.7	1.7	HELIX	2.3	2.3	2.3
BIGGS3	0.0	1.0	1.0	HILBERTA	10.0	2.6	2.6
BIGGS6	3.0	3.9	3.9	HIMMELBF	2.6	2.6	2.6
BOX2	0.0	4.3	4.3	HIMMELBG	0.0	0.0	0.0
BOX3	0.0	3.3	3.3	JENSMP	0.0	0.0	0.0
BRKMCC	0.0	1.7	1.7	KOWOSB	0.0	0.0	0.0
BROWNAL	5.9	5.9	5.9	MANCINO	5.9	5.9	5.9
BROWNDEN	0.0	0.0	0.0	MARATOSB	1.7	1.7	1.7
CHNROSNB	0.0	0.0	0.0	MEXHAT	1.7	1.7	1.7
CRAGGLVY	0.0	0.0	0.0	MOREBV	5.9	5.9	5.9
CUBE02	2.0	2.0	2.0	OSBORNEB	0.0	0.0	0.0
DENSCHND	7.8	7.8	7.8	PALMER1C	0.0	0.0	0.0
DENSCHNE	2.5	3.0	3.0	PALMER3C	0.0	0.0	0.0
DENSCHNF	1.7	1.7	1.7	PALMER5C	0.0	0.0	0.0
DIXMAANC	1.8	0.4	0.4	PALMER8C	0.0	0.0	0.0
DIXMAANG	0.0	0.0	0.0	POWER	10.0	10.0	10.0
DIXMAANI	0.0	0.0	0.0	ROSENBR	1.7	1.7	1.7
DIXMAANK	0.0	0.0	0.0	SINEVAL	0.0	0.0	0.0
DIXON3DQ	0.0	0.0	0.0	SISSER	6.7	5.0	5.0
DQDRTIC	10.7	10.1	10.1	VARDIM	3.0	3.0	3.0
ENGVAL1	7.0	2.0	2.0	YFITU	2.3	2.3	2.3
ENGVAL2	0.0	2.3	2.3	ZANGWIL2	0.0	0.0	0.0
EXPFIT	1.7	1.7	1.7				

Tabela 8.1: Gastos totais devidos ao Passo 1 do DF0-LP em unidade correspondente ao cálculo um gradiente simplex de um conjunto de $n + 1$ pontos.

Ainda com o interesse canalizado no comportamento dos *solvers* a curto prazo, foi mantida a proposta inicial de disponibilizar para cada par $(p, s) \in \mathcal{P} \times \mathcal{S}$ um orçamento de avaliações de função que garantisse pelo menos o equivalente à construção de 150 gradientes simplex. Como o conjunto \mathcal{P} foi mantido inalterado, a maior dimensão presente permaneceu sendo $n = 15$ e, portanto, μ_f foi conservado com o valor de 2400.

A obtenção e o tratamento dos dados se deram da maneira relatada na Seção 8.0.1, com a construção da estrutura H a partir dos históricos de saída de cada um dos elementos de \mathcal{S} . Novamente, ambas as vertentes de referência para o valor de f_L^p foram consideradas na construção dos gráficos.

As Figuras 8.9 e 8.8 contêm os perfis informativos dos quatro *solvers*, de acordo com a referência externa e interna, respectivamente. Através dessas figuras, é possível observar que, de um modo geral, o DF0-LP1 é capaz de resolver mais problemas de \mathcal{P} do que seus rivais, principalmente com o aumento da exigência na acurácia, isto é, com a diminuição do valor de τ . Ainda, fixado τ , observa-se que quanto menor o número de gradientes simplex disponível, mais equiparáveis são os desempenhos dos *solvers*, ainda que nitidamente o **fminsearch** não tenha superado nenhum dos demais *solvers* nesses casos. Naturalmente, quanto menor a exigência na acurácia do valor ótimo da função, mais problemas são resolvidos por um mesmo *solver*.

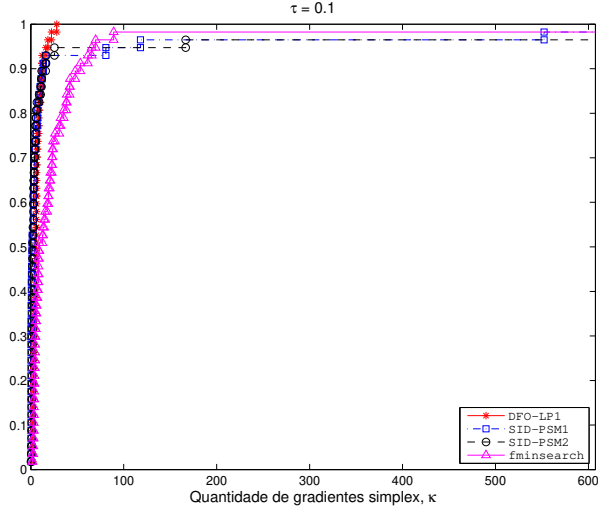
Apesar do fraco desempenho geral do **fminsearch**, as Figuras 8.8a e 8.9a mostram que para $\tau = 0.1$ e com o equivalente a mais de 100 gradientes simplex à disposição, o usuário conseguiria resolver mais problemas com o **fminsearch** do que com qualquer uma das duas implementações do **SID-PSM**.

Ainda que a semelhança entre as Figuras 8.9 e 8.8 seja considerável, deve-se notar algumas diferenças sutis entre elas, que se intensificam a partir de seus itens (c). As curvas do Gráfico 8.9c estão levemente deslocadas para baixo com relação ao Gráfico 8.8c; todos os *solvers* exibem mais robustez no segundo do que primeiro. A dificuldade em atingir os valores de **KNITRO** fica mais evidente no item (d) dessas figuras. A primeira diferença se nota na amplitude do eixo horizontal: enquanto o cenário em 8.9d se estabiliza antes dos 350 gradientes simplex, em 8.8d isso só ocorre por volta dos 800 gradientes simplex. Além disso, todos os *solvers* conseguiram resolver mais problemas em 8.8d do que em 8.9d. Por exemplo, em ambos os gráficos, o **SID-PSM1** alcança a estabilidade com o equivalente a 150 gradientes simplex. Entretanto, não consegue reproduzir no segundo os 85% de resolução dos problemas atingido no primeiro.

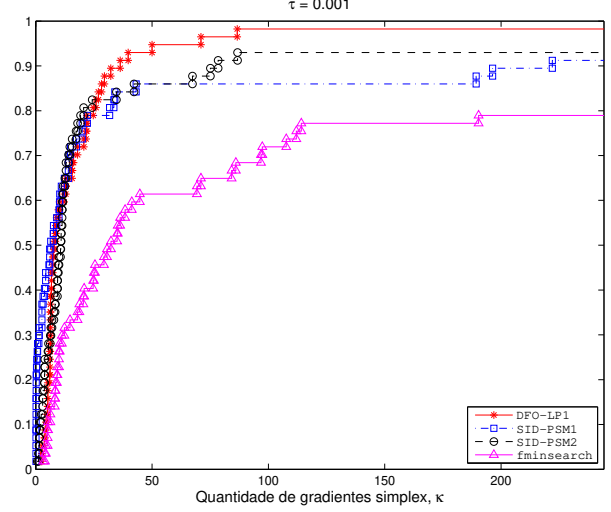
Os perfis de desempenho para os *solvers* de \mathcal{S} são apresentados nas Figuras 8.10 e 8.11 em escala semilogarítmica. Novamente, confirmando as constatações dos perfis informativos, em todos os níveis de acurácia e para ambas as referências de f_L^p , a rotina DF0-LP1 foi mais robusta do que as demais. Ainda, para os níveis de precisão mais altos ($\tau = 10^{-5}$ e $\tau = 10^{-7}$) e também para os dois tipos de referência, o DF0-LP1 foi mais eficiente que todos os seus concorrentes. Considerando o cenário aparentemente mais rigoroso com os *solvers*, isto é, o correspondente ao Gráfico 8.11d, os números sobre o desempenho do DF0-LP1 são notáveis:

- ele resolveu com menos avaliações de função que os demais *solvers* uma fatia de aproximadamente 55% dos problemas de \mathcal{P} em um nível $\tau = 10^{-7}$ de acurácia, para valores de referência f_L^p vindos da aplicação de um *software* com uso de derivadas, respeitado o limite $\mu_f = 2400$ no número máximo de avaliações de função;

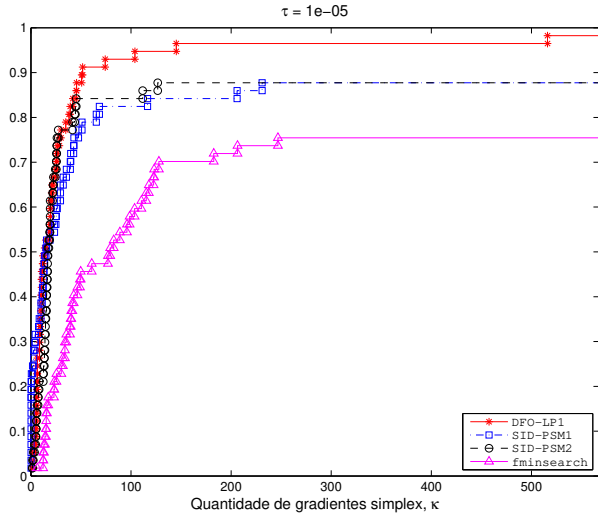
- dentro do mesmo orçamento μ_f , foi capaz de resolver aproximadamente 93% dos problemas de \mathcal{P} , o que representa quase 10% a mais que o segundo *solver* mais robusto deste cenário.



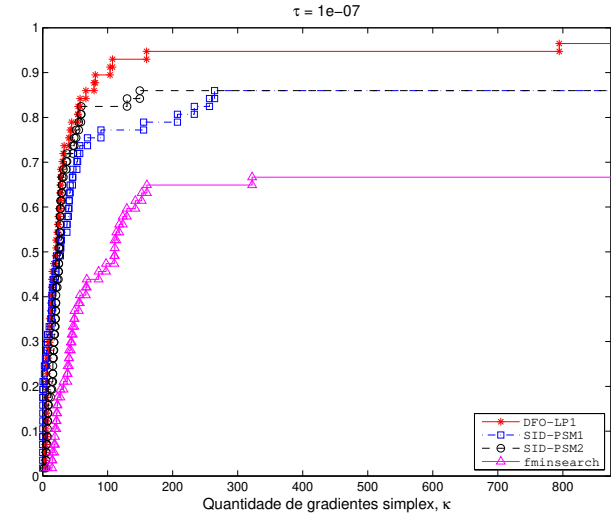
(a)



(b)



(c)



(d)

Figura 8.8: Perfis informativos dos *solvers* de \mathcal{S} para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p obtida como o menor valor de $h_{2400}^{p,s}$ (cf. (8.0.1)) dentre os *solvers* de \mathcal{S} . Eixo horizontal em unidade equivalente ao número de avaliações de função necessário para o cálculo de um gradiente simplex.

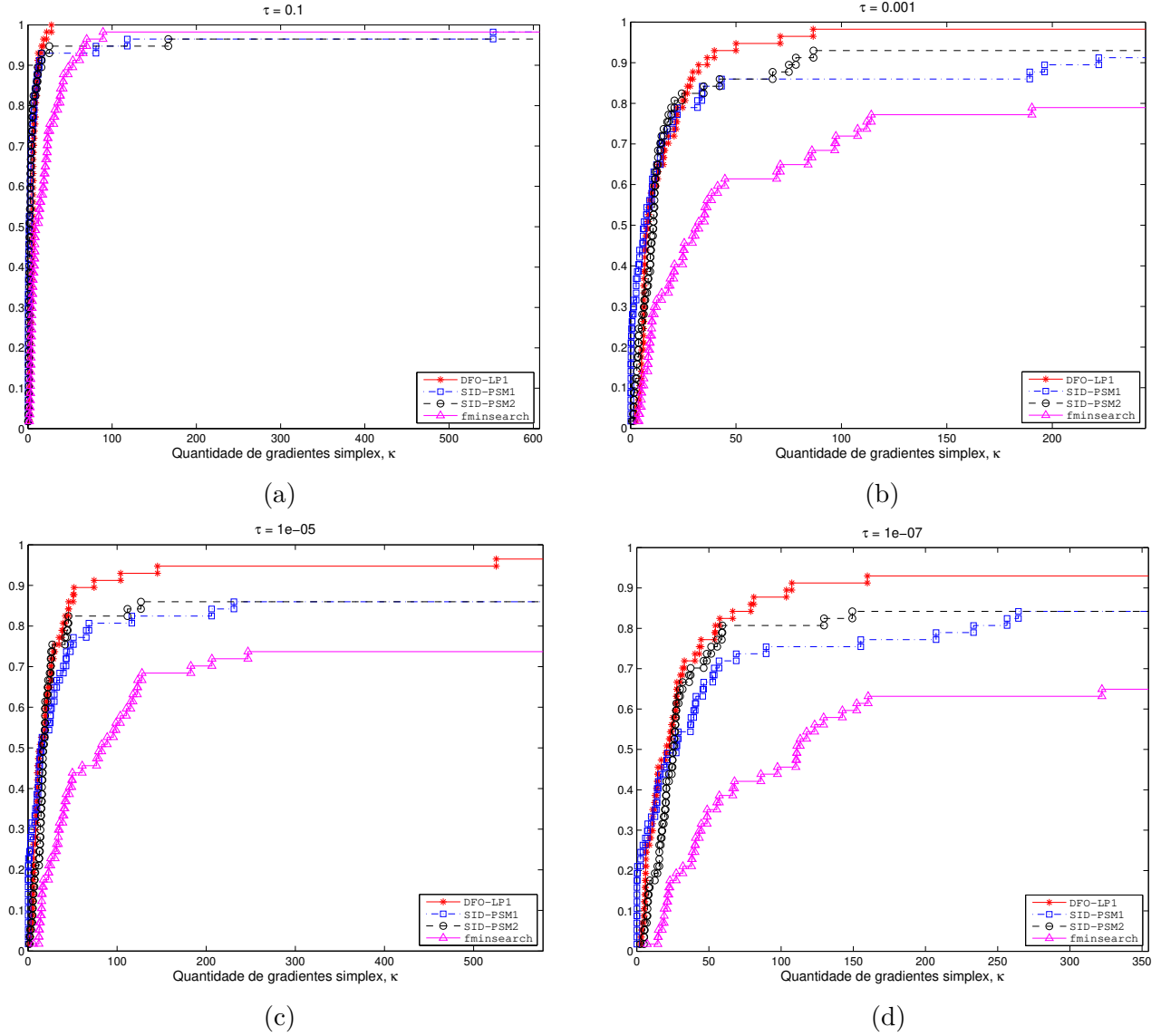
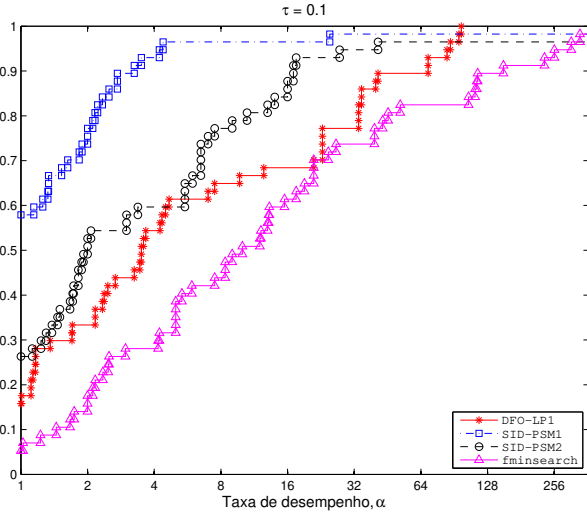


Figura 8.9: Perfis informativos dos *solvers* de \mathcal{S} para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p extraída da Tabela 7.2 de *Fasano et al.* Eixo horizontal em unidade equivalente ao número de avaliações de função necessário para o cálculo de um gradiente simplex.

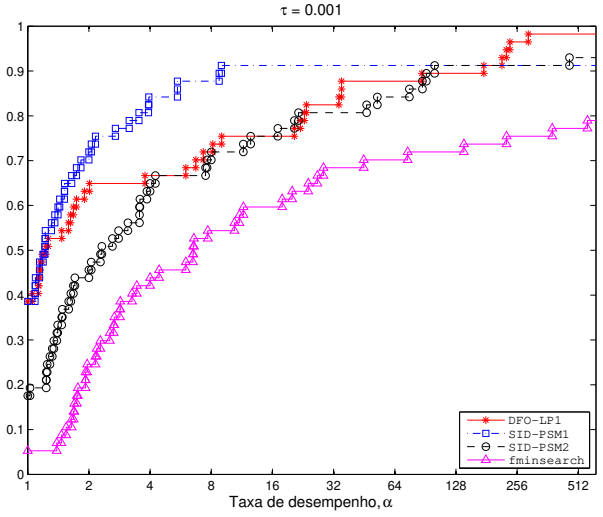
Diversas outras informações podem ser extraídas dos perfis de desempenho. Por exemplo, no Gráfico 8.10b, o DFO-LP1 atinge o valor aproximado de 73% quando a taxa de desempenho gira em torno de 8, enquanto que para esta mesma taxa, o SID-PSM1 alcança um patamar próximo aos 87%. Das definições de taxa e perfil de desempenho é possível concluir que o SID-PSM1 resolveu em torno de 14% a mais de problemas de \mathcal{P} consumindo menos do que 8 vezes o gasto mínimo possível para esses problemas com os *solvers* de \mathcal{S} , se comparado ao DFO-LP1. Curiosamente, quando $\tau = 0.1$, o SID-PSM1 resolveu quase 60% dos problemas gastando tanto ou menos do que

o restante das rotinas de \mathcal{S} (Gráficos 8.10a e 8.11a).

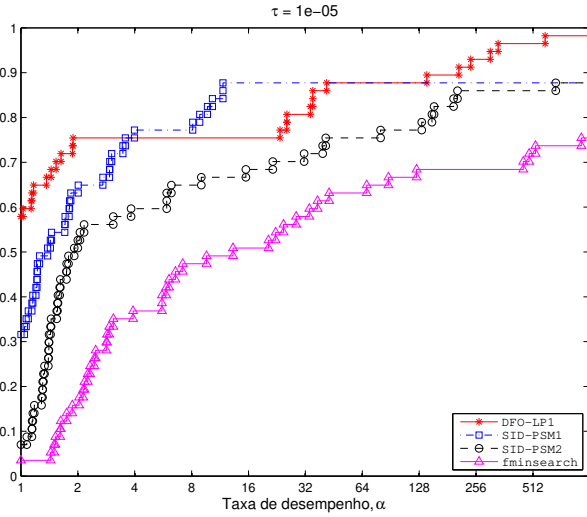
Por fim, vale também destacar que ao contrário do que se esperava baseado nos resultados de Cervelin [8], o SID-PSM1 obteve desempenho quase sempre igual ou superior ao de seu irmão SID-PSM2, tanto em eficiência como em robustez. As raras exceções são as dos Gráficos 8.10b e 8.11b, nos quais o SID-PSM2 conseguiu resolver ligeiramente mais problemas que o SID-PSM1, numa ultrapassagem de último instante por volta da taxa de desempenho $\alpha = 512$.



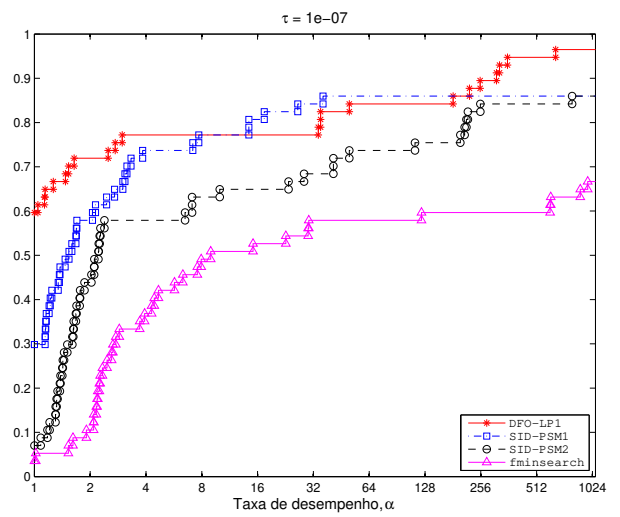
(a)



(b)

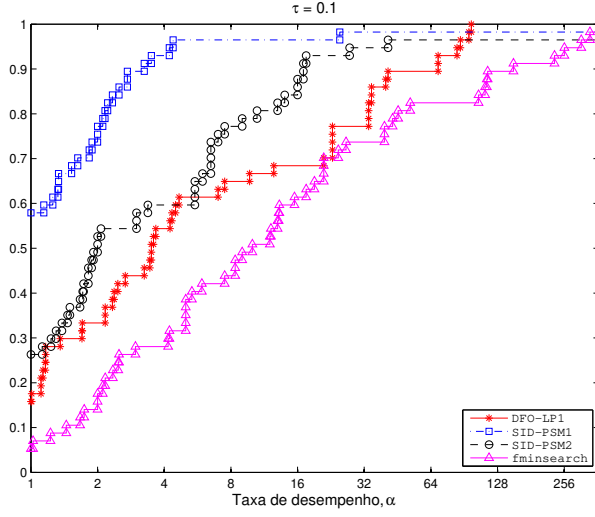


(c)

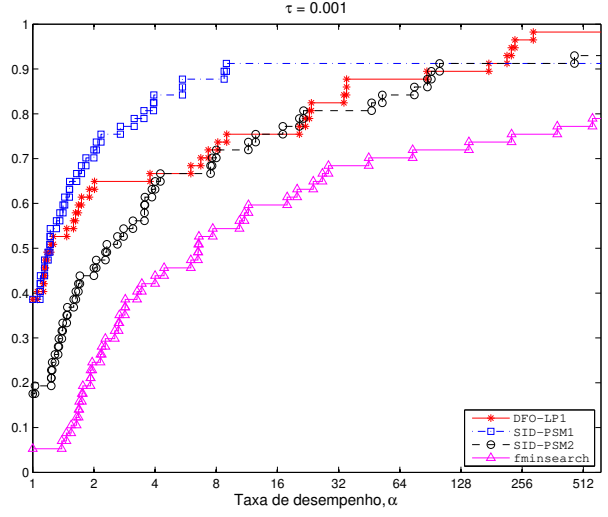


(d)

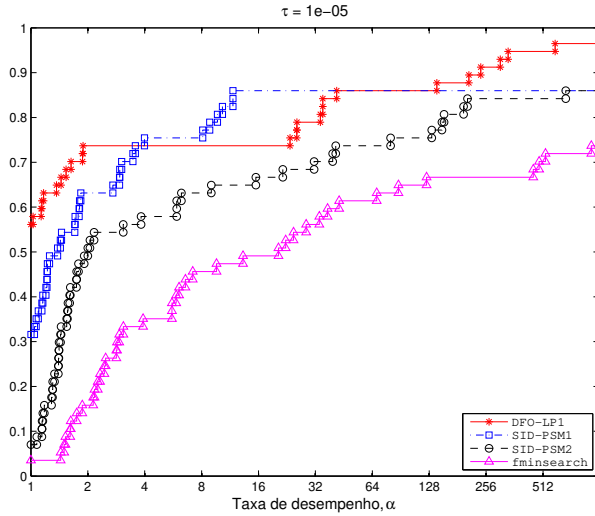
Figura 8.10: Perfis de desempenho dos *solvers* de \mathcal{S} para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p obtida como o menor valor de $h_{2400}^{p,s}$ (cf. (8.0.1)) dentre os solvers de \mathcal{S} . O eixo horizontal representa a taxa de desempenho relacionada à medida “número de avaliações de função”.



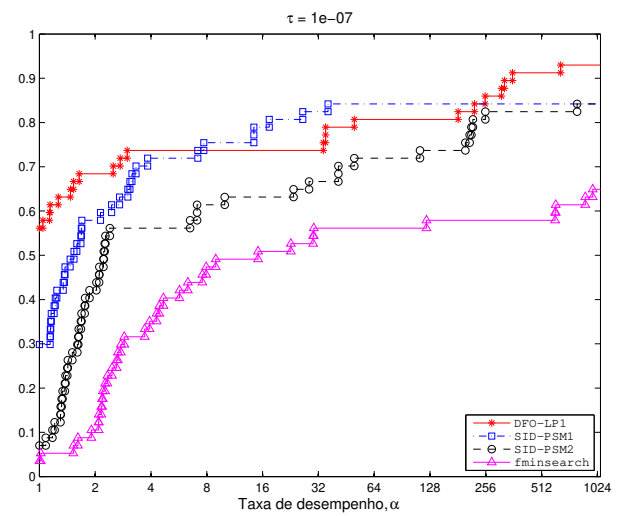
(a)



(b)



(c)



(d)

Figura 8.11: Perfis de desempenho dos *solvers* de \mathcal{S} para quatro níveis de acurácia τ . Teste de convergência com base na referência f_L^p extraída da Tabela 7.2 de *Fasano et al.* O eixo horizontal representa a taxa de desempenho relacionada à medida “número de avaliações de função”.

8.2.1 Problemas não resolvidos pelo DF0-LP1

A metodologia de comparação de desempenho de *solvers* baseada nos perfis informativos e de desempenho tem um caráter quantitativo e, até certo ponto, panorâmico, no sentido de que atribui uma mesma importância a problemas com características peculiares. Esse caráter é plenamente justificado quando se leva em consideração que um tratamento individual para os problemas de \mathcal{P} tornaria o processo longo e de conclusão árdua. Junte-se a isso o fato de que as comparações são feitas com base no valor da função objetivo alcançado com a evolução das iterações e, sem informações adicionais sobre os problemas, não se prestam de garantia para a qualidade das soluções obtidas. Assim sendo, para uma maior compreensão dos resultados deste trabalho, alguns problemas serão analisados com um pouco mais de profundidade nesta seção.

De acordo com o Gráfico 8.11d, o DF0-LP1 conseguiu resolver aproximadamente 97% do problemas de \mathcal{P} com acurácia $\tau = 10^{-7}$, sendo o valor f_L^p obtido da Tabela 7.2. Com o armazenamento dos históricos das resoluções, foi possível rastrear os problemas não aprovados no teste de convergência e aos quais, portanto, se atribuiu o *status* de não resolvidos para esse *solver*. Foram eles: **CHNROSNB** ($n = 15$), **JENSMP** ($n = 3$), **KOWOSB** ($n = 4$) e **MARATOSB** ($n = 2$).

A Tabela 8.2 exibe os valores ótimos para esses quatro problemas. Com exceção dos dados do KNITRO, cujos resultados são os relatados por Fasano *et al.* [21], os demais valores se referem aos experimentos deste trabalho com os *solvers* de \mathcal{S} .

	CHNROSNB	JENSMP
KNITRO	1.215891488553000e-19	1.243621823556150e+02
DF0-LP1	3.942731470417066	1.243630934205725e+02
SID-PSM1	0	1.243621823556148e+02
SID-PSM2	1.569455709223100e-04	1.243621823556148e+02
fminsearch	33.220014452645955	1.243621823556148e+02
	KOWOSB	MARATOSB
KNITRO	3.075056038492380e-04	-1.000000062499990
DF0-LP1	3.078009542140385e-04	-0.968361511108442
SID-PSM1	3.078009467333194e-04	-0.144113268797668
SID-PSM2	3.078009467333190e-04	0.962212863475847
fminsearch	3.078009467333199e-04	-0.031146149500186

Tabela 8.2: Valores ótimos obtidos com o KNITRO e com os *solvers* de \mathcal{S} para os quatro problemas selecionados.

Em se analisando direta e isoladamente os resultados finais dessa tabela, observa-se que para **JENSMP** e **KOWOSB**, o valor alcançado pelo DF0-LP1 é o que menos se aproxima do referencial (KNITRO). Ainda assim, há uma certa concordância entre os *solvers* de \mathcal{S} no que se refere a dígitos significativos e ordem de grandeza dos valores ótimos obtidos.

No entanto, para **CHNROSNB** e **MARATOSB**, há uma discordância entre os valores de

cada *solver* apresentados na tabela. Se, por um lado, em **CHNROSNB** o DF0-LP1 exibe um dos valores ótimos mais *distantes* do esperado, por outro, em **MARATOSB**, ele é o que mais se aproxima do referencial seguido. Vale frisar, também, que o SID-PSM1 conseguiu obter o resultado ótimo teórico previsto para o problema **CHNROSNB**.

Apenas para ilustrar as sutilezas que diferenciam uma análise por valor de função de uma análise por solução encontrada, considere esses mesmos dois problemas para os quais houve um maior contraste nos resultados, dentre os quatro representantes filtrados. A solução teórica do problema **CHNROSNB** é o vetor do \mathbb{R}^{15} com entradas unitárias, precisamente a solução obtida pelo SID-PSM1. O DF0-LP1, no entanto, obteve a seguinte solução para esse problema:

$$x^* = \begin{pmatrix} 0.999974749013595 \\ 0.999982921534821 \\ 0.999969946075182 \\ 0.999998256079741 \\ 0.999986523392786 \\ 0.999896350620229 \\ 0.999955415136419 \\ 0.999835185597430 \\ 0.999401340596688 \\ 0.998853304673640 \\ 0.998540571427863 \\ 0.997688818902005 \\ 0.993721177743978 \\ 0.986574015039094 \\ -0.970873946692002 \end{pmatrix},$$

cujo erro relativo na norma Euclidiana é de aproximadamente 51%, atribuído em grande parte à última componente da solução numérica. Contudo, para o problema bidimensional de **MARATOSNB**, cuja solução teórica prevista na literatura [27, Capítulo 5] é o vetor $(-1 \ 0)^T$, a solução obtida pelo DF0-LP1 foi a seguinte:

$$x^* = \begin{pmatrix} -0.968492664154838 \\ -0.249065075201878 \end{pmatrix},$$

significando um erro relativo de 25% na solução numérica, também na norma Euclidiana.

Como, na prática, o principal interesse do usuário de *softwares* sem derivada está no decréscimo no valor de função, olhar apenas para os valores ótimos, sem considerar os valores iniciais, pode resultar em conclusões menos realísticas. Ademais, uma comparação com a solução “verdadeira” (desconhecida) quase sempre é impossível de ser feita. De fato, o teste de convergência 5.1.1 reproduz esse interesse em situações práticas reais.

Na Tabela 8.3 são apresentados os decréscimos resultantes da aplicação dos *solvers* de \mathcal{S} e o decréscimo do KNITRO, este último ainda baseado nos valores publicados por Fasano *et al.* Com ela, é possível observar melhor por que esses quatro problemas foram decretados como não resolvidos por DF0-LP1 para uma acurácia $\tau = 10^{-7}$. De acordo com (5.1.1), para esse valor de τ , o DF0-LP1

precisaria apresentar um decréscimo de pelo menos $(1 - 10^{-7}) = 99.99999\%$ do obtido pelo KNITRO, para cada problema com o *status* de “resolvido”.

	CHNROSNB	JENSMP
KNITRO	2.121280000000000e+03	4.046943979604877e+03
DF0-LP1	2.117337268529583e+03	4.046943068539920e+03
SID-PSM1	2.121280000000000e+03	4.046943979604877e+03
SID-PSM2	2.121279843054429e+03	4.046943979604877e+03
fminsearch	2.088059985547354e+03	4.046943979604877e+03

	KOWOSB	MARATOSB
KNITRO	0.005006109754343	4.840210000006259e+04
DF0-LP1	0.005005814403978	4.840206836151119e+04
SID-PSM1	0.005005814411459	4.840124411326888e+04
SID-PSM2	0.005005814411459	4.840013778713661e+04
fminsearch	0.005005814411459	4.840113114614959e+04

Tabela 8.3: Decréscimo $[f(x^0) - f(x^*)]$ para cada um dos quatro problemas selecionados, onde $f(x^*)$ é o valor ótimo obtido pelos *solvers* de \mathcal{S} , exceto no caso do KNITRO, para o qual $f(x^*)$ é o valor ótimo correspondente retirado da Tabela 7.2.

Finalmente, para que sirva de fomento à discussão sobre qual deve ser a origem do valor de referência f_L^p utilizado, destaca-se que nos experimentos deste trabalho nem sempre o valor ótimo do KNITRO extraído da Tabela 7.2 foi menor do que os obtidos pelos *solvers* de \mathcal{S} . O próprio DF0-LP1 obteve valores ótimos numericamente menores que os do KNITRO em 18 de 57 problemas P , sendo que em 4 destes ele foi o único de \mathcal{S} a superar o oponente de otimização com derivada. Isso significa que aproximadamente 32% dos valores ótimos alcançados pelo DF0-LP1 foram numericamente melhores que os de KNITRO relatados por Fasano *et. al.* Em outras palavras, a referência interna de f_L^p foi mais exigente do que a externa para quase um terço dos problemas, nos experimentos aqui descritos. Ou seja, para quase 70% dos problemas, foi mais fácil declará-los como resolvidos quando a referência era interna do que quando esta era externa, o que justifica a maior robustez geral apresentada pelos *solvers* de \mathcal{S} nas Figuras 8.8 a 8.10.

Para finalizar, são mencionados na Tabela 8.4 os problemas não resolvidos para cada um dos *solvers* e acurácias considerados no segundo bloco de experimentos deste trabalho. Por concisão, cada problema foi citado apenas na primeira vez em que se constatou a sua não resolução por determinado *solver*. Entretanto, está claro que se um *solver* não conseguiu resolver determinado problema em certo nível τ de acurácia, ele também não conseguirá resolvê-lo para níveis de acurácia ainda mais exigentes que esse.

$\tau = 10^{-1}$	
DF0-LP1	—
SID-PSM1	ENGVAL2.
SID-PSM2	BIGGS6; GULF.
fminsearch	FREUROTH.
$\tau = 10^{-3}$	
DF0-LP1	CHNROSNB.
SID-PSM1	GULF; HIMMELBF; OSBORNEB; YFITU.
SID-PSM2	SINEVAL; YFITU
fminsearch	BOX2; BOX3; BROWNAL; CHNROSNB; CRAGGLVY; DIXMAANC; DIXMAANG; DIXMAANI; DIXMAANK; MANCINO.
$\tau = 10^{-5}$	
DF0-LP1	KOWOSB; MARATOSB;
SID-PSM1	FREUROTH; HIMMELBF; KOWOSB; MARATOSB.
SID-PSM2	FREUROTH; HIMMELBF; KOWOSB; MARATOSB.
fminsearch	KOWOSB; MARATOSB; OSBORNEB.
$\tau = 10^{-7}$	
DF0-LP1	MARATOSB.
SID-PSM1	PALMER8C
SID-PSM2	PALMER8C
fminsearch	ARWHEAD; PALMER1C; PALME3C; PALMER8C; VARDIM.

Tabela 8.4: Problemas decretados como não resolvidos de acordo com a referência externa. Para evitar repetições, apenas a primeira ocorrência para cada *solver* está indicada. Portanto, fixado o *solver*, um problema não resolvido em certo nível τ_1 permanece com esse mesmo *status* nos níveis $\tau < \tau_1$ subsequentes.

Considerações finais e perspectivas futuras

Neste trabalho, foi apresentada uma implementação de um método de região de confiança direcionado a problemas não lineares de otimização irrestrita sem derivadas e que se favorece do mecanismo de autocorreção da geometria dos pontos amostrais tratado em [47], onde também é demonstrada sua convergência global. Com a utilização de modelos de interpolação quadráticos completos, os experimentos numéricos relatados aqui sugerem tanto a eficiência como a robustez do método DF0-LP1, dentro da representatividade dos conjuntos \mathcal{S} e \mathcal{P} . Tal conclusão é ainda mais evidente se constatada a necessidade de níveis de acurácia mais altos, como o de $\tau = 10^{-7}$.

Ainda, enfatizou-se uma metodologia relativamente nova de comparação de *solvers*, com o uso dos perfis informativos [37], mais focada na aplicação prática dos métodos e no processo de decisão a que um usuário de otimização sem derivadas tipicamente é submetido. Quanto à escolha do valor ótimo de referência adotado no teste de convergência (5.1.1), observou-se que a proposta da chamada referenciação interna pressupõe que cada problema pode ser razoavelmente bem resolvido por algum dos *solvers* do conjunto \mathcal{S} ; caso contrário, as conclusões sobre eficiência e robustez podem ser ilusórias ou extremamente limitadas. Por outro lado, sob um aspecto mais conceitual, não se pode esperar que o desempenho de *solvers* com e sem derivada sejam sempre equiparáveis, o que advoga a favor desse tipo de referenciação.

Através das análises de entrada no teste de criticalidade e do monitoramento dos gradientes dos modelos, foi possível perceber que, mesmo quando o método está próximo à solução ótima conhecida, a norma do gradiente do modelo não é, em geral, uma boa medida de otimalidade para o problema. Tais conclusões persistiram mesmo quando a primeira entrada no teste de criticalidade foi facilitada ou dificultada através da variação dos valores de ϵ_0 . Os resultados indicam também que, ao contrário do que se poderia esperar, o custo em avaliações de função atribuído exclusivamente aos testes de criticalidade não foi relativamente significativo, se comparado à potencial troca de todos os pontos do conjunto \mathcal{Y} durante a busca pelo modelo Λ -posicionado no Passo 1.

Por fim, o estudo apresentado aqui, assim como o algoritmo teórico proposto por Scheinberg e Toint [47], limitou a construção dos modelos aproximadores à situação em que a cardinalidade do conjunto \mathcal{Y} é exatamente igual à dimensão do espaço polinomial considerado. Um possível prolongamento deste trabalho está na flexibilização da cardinalidade de \mathcal{Y} , o que envolveria modelos de aproximação subdeterminados ou até mesmo modelos de regressão não linear. A extensão

do conceito de polinômios de Lagrange em alguns desses casos é imediata e, assim, o estudo de uma possível geometria autocorretiva nessas situações parece bastante plausível. É de nosso interesse particular o caso que envolve o uso de modelos de norma- ℓ_1 mínima, os quais obtiveram resultados surpreendentes em um trabalho recente de Bandeira *et. al.* [5]. Uma pergunta que surge naturalmente e que, até onde é de nosso conhecimento, não foi respondida, está relacionada com a possível definição de polinômios de Lagrange de norma- ℓ_1 mínima, análoga à dos já bem definidos polinômios de Lagrange de norma de Frobenius mínima: estaria a geometria autocorretiva presente também nesses casos? A análise sobre as limitações da extensão de tais conceitos pode servir, portanto, de objeto de estudos futuros.

Referências Bibliográficas

- [1] *Introducing MEX-files*. http://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html. Último acesso em 28/02/2014.
- [2] *Mathworks, accelerating the pace of engineering and science*. <http://www.mathworks.com/>. Último acesso em 28/02/2014.
- [3] M. A. ABRAMSON, *Pattern search algorithms for mixed variable general constrained optimization problems*, PhD thesis, École Polytechnique de Montréal, 2002.
- [4] C. AUDET, V. BÉCHARD, AND J. CHAOUKI, *Spent potliner treatment process optimization using a MADS algorithm*, Optimization and Engineering, 9 (2008), pp. 143–160.
- [5] A. S. BANDEIRA, K. SCHEINBERG, AND L. N. VICENTE, *Computation of sparse low degree interpolating polynomials and their application to derivative-free optimization*, Mathematical programming, 134 (2012), pp. 223–257.
- [6] M. S. BAZARAA, H. D. SHERALI, AND C. M. SHETTY, *Nonlinear programming: theory and algorithms*, John Wiley & Sons, third ed., 2006.
- [7] D. BORTZ AND C. KELLEY, *The simplex gradient and noisy optimization problems*, in Computational Methods for Optimal Design and Control, Springer, 1998, pp. 77–90.
- [8] B. H. CERVELIN, *Sobre um método de minimização irrestrita baseado em derivadas simplex*. Universidade Estadual de Campinas, 2013. Dissertação de Mestrado.
- [9] P. G. CIARLET AND P. RAVIART, *General Lagrange and Hermite interpolation in \mathbb{R}^n with applications to finite element methods*, Archive for Rational Mechanics and Analysis, 46 (1972), pp. 177–199.
- [10] A. R. CONN, N. I. M. GOULD, D. ORBAN, AND PH. L. TOINT, *The SIF Reference Report (revised version)*, Relatório online, 5 (2003).
- [11] A. R. CONN, N. I. M. GOULD, AND PH. L. TOINT, *Trust region methods*, no. 1, SIAM, Philadelphia, 2000.

-
- [12] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Geometry of interpolation sets in derivative free optimization*, Mathematical Programming, 111 (2008), pp. 141–172.
- [13] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Global convergence of general derivative-free trust-region algorithms to first-and second-order critical points*, SIAM Journal on Optimization, 20 (2009), pp. 387–415.
- [14] —, *Introduction to derivative-free optimization*, vol. 8, SIAM, Philadelphia, 2009.
- [15] A. L. CUSTÓDIO, H. ROCHA, AND L. N. VICENTE, *Incorporating minimum Frobenius norm models in direct search*, Computational Optimization and Applications, 46 (2010), pp. 265–278.
- [16] A. L. CUSTÓDIO AND L. N. VICENTE, *SID-PSM: A pattern search method guided by simplex derivatives for use in derivative-free optimization*. <http://www.mat.uc.pt/sid-psm/>. Último acesso em 28/02/2014.
- [17] —, *Using sampling and simplex derivatives in pattern search methods*, SIAM Journal on Optimization, 18 (2007), pp. 537–555.
- [18] S. N. DEMING, L. R. PARKER JR, AND M. BONNER DENTON, *A review of simplex optimization in analytical chemistry*, Critical Reviews in Analytical Chemistry, 7 (1978), pp. 187–202.
- [19] M. A. DINIZ-EHRHARDT, V. L. R. LOPES, AND L. G. PEDROSO, *Métodos sem derivadas para minimização irrestrita*, Notas em Matemática Aplicada, 49 (2010), pp. 1608–1694.
- [20] E. D. DOLAN AND J. J. MORÉ, *Benchmarking optimization software with performance profiles*, Mathematical programming, 91 (2002), pp. 201–213.
- [21] G. FASANO, J. L. MORALES, AND J. NOCEDAL, *On the geometry phase in model-based algorithms for derivative-free optimization*, Optimization Methods & Software, 24 (2009), pp. 145–154.
- [22] N. I. M. GOULD, D. ORBAN, AND PH. L. TOINT, *CUTEr: a Constrained and Unconstrained Testing Environment, revisited*. <http://www.cuter.rl.ac.uk/>. Último acesso em 28/02/2014.
- [23] —, *CUTEr Development Website*. <http://tracsvn.mathappl.polymtl.ca/trac/cuter/>. Último acesso em 28/02/2014.
- [24] —, *CUTEr: New MATLAB interface*. <http://tracsvn.mathappl.polymtl.ca/trac/cuter/wiki/NewMatlabInterface>. Último acesso em 28/02/2014.
- [25] —, *CUTEst: a Constrained and Unconstrained Testing Environment on steroids*. <http://ccpforge.cse.rl.ac.uk/gf/project/cutest/>. Último acesso em 28/02/2014.

- [26] S. GRATTON, PH. L. TOINT, AND A. TRÖLTZSCH, *An active-set trust-region method for derivative-free nonlinear bound-constrained optimization*, Optimization Methods and Software, 26 (2011), pp. 873–894.
- [27] L. GRIPPO, F. LAMPARIELLO, AND S. LUCIDI, *A class of nonmonotone stabilization methods in unconstrained optimization*, Numerische Mathematik, 59 (1991), pp. 779–805.
- [28] H. L. GUIDORIZZI, *Um curso de cálculo*, vol. 1, Editora LTC, Rio de Janeiro, 2001.
- [29] R. HOOKE AND T. A. JEEVES, “*Direct search*” *solution of numerical and statistical problems*, Journal of the ACM (JACM), 8 (1961), pp. 212–229.
- [30] T. G. KOLDA, R. M. LEWIS, AND V. TORCZON, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM review, 45 (2003), pp. 385–482.
- [31] J. C. LAGARIAS, J. A. REEDS, M. H. WRIGHT, AND P. E. WRIGHT, *Convergence properties of the Nelder–Mead simplex method in low dimensions*, SIAM Journal on Optimization, 9 (1998), pp. 112–147.
- [32] H. LE THI, A. VAZ, AND L. VICENTE, *Optimizing radial basis functions by DC. programming and its use in direct search for global derivative-free optimization*, TOP, 20 (2012), pp. 190–214.
- [33] M. MARAZZI AND J. NOCEDAL, *Wedge trust region methods for derivative free optimization*, Mathematical programming, 91 (2002), pp. 289–305.
- [34] A. L. MARSDEN, J. A. FEINSTEIN, AND C. A. TAYLOR, *A computational framework for derivative-free optimization of cardiovascular geometries*, Computer methods in applied mechanics and engineering, 197 (2008), pp. 1890–1905.
- [35] K. I. M. MCKINNON, *Convergence of the Nelder–Mead Simplex method to a nonstationary point*, SIAM Journal on Optimization, 9 (1998), pp. 148–158.
- [36] J. J. MORÉ AND D. C. SORESENSEN, *Computing a trust region step*, SIAM Journal on Scientific and Statistical Computing, 4 (1983), pp. 553–572.
- [37] J. J. MORÉ AND S. M. WILD, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), pp. 172–191.
- [38] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, The Computer Journal, 7 (1965), pp. 308–313.
- [39] R. OEUVRAY, *Trust-region methods based on radial basis functions with application to bio-medical imaging*, PhD thesis, École Polytechnique Fédérale de Lausanne, 2005.
- [40] M. J. D. POWELL, *On the Lagrange functions of quadratic models that are defined by interpolation*, Optimization Methods and Software, 16 (2001), pp. 289–309.

-
- [41] —, *UOBYQA: unconstrained optimization by quadratic approximation*, Mathematical Programming, 92 (2002), pp. 555–582.
- [42] —, *The NEWUOA software for unconstrained optimization without derivatives*, in Large-Scale Nonlinear Optimization, G. Pillo and M. Roma, eds., vol. 83 of Nonconvex Optimization and Its Applications, Springer US, 2006, pp. 255–297.
- [43] —, *Developments of NEWUOA for minimization without derivatives*, IMA Journal of Numerical Analysis, 28 (2008), pp. 649–664.
- [44] C. QUINTERO, C. RAMIREZ, R. SANCHEZ, ET AL., *CUTEr Installation Manual for LINUX platform under Ubuntu 9.04*. <http://www.math.utep.edu/Student/rsanchez/files/CUTErManual.pdf>. Último acesso em 28/02/2014.
- [45] L. M. RIOS AND N. V. SAHINIDIS, *Derivative-free optimization: A review of algorithms and comparison of software implementations*, Journal of Global Optimization, (2012), pp. 1–47.
- [46] T. J. RIVLIN, *The Lebesgue constants for polynomial interpolation*, in Functional Analysis and its Applications, H. Garnir, K. Unni, and J. Williamson, eds., vol. 399 of Lecture Notes in Mathematics, Springer Berlin Heidelberg, 1974, pp. 422–437.
- [47] K. SCHEINBERG AND PH. L. TOINT, *Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization*, SIAM Journal on Optimization, 20 (2010), pp. 3512–3532.
- [48] K. SCHITTKOWSKI, *More test examples for nonlinear programming codes*, vol. 282 of Lecture Notes in Economics and Mathematical Systems, Springer, New York, 1987.
- [49] R. G. C. SILVA, T. MACEDO, J. SOARES, ET AL., *Modelo de dissertação/tese do Instituto de Matemática, Estatística e Computação Científica (IMECC) da Universidade Estadual de Campinas (UNICAMP)*. https://github.com/r-gaia-cs/modelo_tese_imecc, 2013. Último acesso em 28/02/2014.
- [50] A. S. SIQUEIRA, D. GONÇALVES, L. F. PRUDENTE, AND R. G. C. SILVA, *Apostila de instalação do CUTEr*. https://github.com/abelsiqueira/tutoriais/blob/master/CUTEr/cuter_pt_br.pdf, 3 2013. Último acesso em 28/02/2014.
- [51] S. J. SMITH, *Lebesgue constants in polynomial interpolation*, Annales Mathematicae et Informaticae, 33 (2006), pp. 1787–5021.
- [52] W. G. R. F. R. SPENDLEY, G. R. HEXT, AND F. R. HIMSWORTH, *Sequential application of simplex designs in optimisation and evolutionary operation*, Technometrics, 4 (1962), pp. 441–461.
- [53] P. STEIN, *A note on the volume of a simplex*, American Mathematical Monthly, 73 (1966), pp. 299–301.

- [54] P. TSENG, *Fortified-descent simplicial search method: A general approach*, SIAM Journal on Optimization, 10 (1999), pp. 269–288.
- [55] A. I. F. VAZ AND L. N. VICENTE, *Pswarm: a hybrid solver for linearly constrained global derivative-free optimization*, Optimization Methods and Software, 24 (2009), pp. 669–685.
- [56] R. A. WALTZ AND J. NOCEDAL, *KNITRO user's manual*, Tech. Report OTC-2003/5, Northwestern University, Evanston, Illinois, 2003.
- [57] D. WINFIELD, *Function minimization by interpolation in a data table*, IMA Journal of Applied Mathematics, 12 (1973), pp. 339–347.
- [58] Z. ZHAO, J. C. MEZA, AND M. VAN HOVE, *Using pattern search methods for surface structure determination of nanomaterials*, Journal of Physics: Condensed Matter, 18 (2006), pp. 8693–8706.