ALGORITMO DE PONTO INTERIOR PARA PROGRAMAÇÃO LINEAR BASEADO NO FDIPA

Angélica Miluzca Victorio Celis

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Nelson Maculan Filho
José Herskovits Norman

Rio de Janeiro
Março de 2018

ALGORITMO DE PONTO INTERIOR PARA PROGRAMAÇÃO LINEAR BASEADO NO FDIPA

Angélica Miluzca Victorio Celis

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. José Herskovits Norman, D.Ing.

Prof. Susana Scheimberg de Makler, D.Sc.

Prof. Anatoli Leontiev, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Celis, Angélica Miluzca Victorio

Algoritmo de Ponto Interior para Programação Linear Baseado no FDIPA/Angélica Miluzca Victorio Celis. – Rio de Janeiro: UFRJ/COPPE, 2018.

X, 39 p. 29, 7cm.

Orientadores: Nelson Maculan Filho

José Herskovits Norman

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 37 – 39.

1. Otimização Linear. 2. Algoritmo de Direções Viável. 3. Algoritmo de Ponto Interior. I. , Nelson Maculan Filho *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicado a meus pais Aurelia e Juan.

Agradecimentos

Quero muito destacar e agradecer:

Ao Professor Dr. Nelson Maculan Filho por acreditar em mim e me aceitar como sua orientada, por suas recomendações, direções e sua paciência, que me guiaram numa selva do conhecimento acadêmico. Ao professor D.Ing. José Herskovits Norman, que muitas vezes perdeu o sono tentando entender algum conceito, teorema ou artigo, suas direções me levaram a caminhos mais tranquilos. Aos demais professores do PESC, em especial às professoras Márcia Fampa e Susana Scheimberg pelas aulas que contribuíram para a minha pesquisa.

Aos meus pais, Aurelia e Juan, por todo o cuidado que tiveram comigo, todo esforço e abnegação a que se propuseram para me conceder a oportunidade de estudar, pela motivação e pelos conselhos, que me ajudaram a enfrentar as angústias e os desafios. Incluo meu irmão Juan, meu sobrinho Dayiro e meu tio Vicente por todo o carinho que, mesmo à distância, me ajudou a prosseguir.

Ao Dr. Edward Quijada Orellana por todo o carinho, por seu companheirismo e por compartilhar seu tempo, tanto na pesquisa, quanto na vida pessoal, a você agradeço com muito carinho.

A tantos outros que fizeram parte de minha vida acadêmica e social, bem como a todos meus amigos do PPGI, PESC e da Vila Residencial, com os quais dividi almoços, ideias, angústias, prazos, disciplinas, listas de exercícios e tantas outras coisas que tornaram a vida acadêmica possível, todos eles contribuíram para tornar a vivência no Brasil uma experiência boa.

Ao PESC/COPPE pelos profissionais que contribuíram em todos os processos administrativos, os quais mesmo em situações de desconforto por tantas outras demandas, jamais deixavam de oferecer o melhor aos alunos e seus pesquisadores.

À CAPES pelo apoio financeiro e ao povo do Brasil por sua hospitalidade.

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

ALGORITMO DE PONTO INTERIOR PARA PROGRAMAÇÃO LINEAR BASEADO NO FDIPA

Angélica Miluzca Victorio Celis

Março/2018

Orientadores: Nelson Maculan Filho
José Herskovits Norman

Programa: Engenharia de Sistemas e Computação

Neste trabalho usamos um algoritmo de pontos interiores e direções viáveis denominado FDIPA, "Feasible Direction Interior Point Algorithm", para resolução de problemas de otimização linear. Em cada iteração o FDIPA calcula uma direção de descida viável do problema mediante uma iteração tipo Newton para resolver as condições de Karush-Kuhn-Tucker (KKT), gerando dois sistemas lineares de equações. São propostas técnicas numéricas para resolver os mesmos de forma eficiente, em particular, mediante um método de gradiente conjugado precondicionado, no qual conseguimos um critério para poder truncá-lo. Finalmente, vários problemas testes serão resolvidos e comparados com resultados da literatura.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

INTERIOR POINT ALGORITHM FOR LINEAR PROGRAMMING BASED ON FDIPA

Angélica Miluzca Victorio Celis

March/2018

Advisors: Nelson Maculan Filho
José Herskovits Norman

Department: Systems Engineering and Computer Science

In this work we use an algorithm of interior points and feasible directions called FDIPA, "Feasible Direction Interior Point Algorithm", for solving linear optimization problems. In each iteration FDIPA calculates a feasible descent direction of the problem by a Newton-type iteration to solve the Karush-Kuhn-Tucker (KKT) conditions; generating two linear systems of equations. Numerical techniques are proposed to solve them efficiently, in particular by means of a preconditioned conjugate gradient method, in which we obtain a criterion to truncate it. Finally, several test problems will be solved and compared with results from the literature.

Sumário

Lista de Tabelas

Capítulo 1

Introdução

Neste trabalho, estamos interessados em estudar o problema de Programação Linear (PL)

$$\begin{aligned} &\text{minimize} && c^t x \\ &\text{sujeito a} && \\ & && A_1 x = b_1, \\ & && A_2 x \leq b_2, \\ & && x \geq 0; \end{aligned} \tag{1.1}$$

onde $A_1 \in R^{p \times n}$, $b_1 \in R^p$, $A_2 \in R^{m \times n}$, $b_2 \in R^m$, $c \in R^n$, $x \in R^n$.

Além disso, $A1$ tem posto completo de linhas e as restrições de não negatividade de x são chamadas de desigualdades triviais.

Dado um problema de programação linear (1.1), podemos acrescentar variáveis de folga às desigualdades não triviais, passando dessa maneira a trabalhar com restrições de igualdade e desigualdade triviais. Ou seja, obteríamos um problema de programação linear da forma padrão (Primal)

$$\begin{aligned} &\text{minimize} && C^t x' \\ &\text{sujeito a} && \\ & && Ax' = b, \\ & && x' \geq 0; \end{aligned}$$

onde $A \in R^{(p+m) \times (n+m)}$, $b \in R^{p+m}$, $C \in R^{n+m}$, $x' \in R^{n+m}$.

A programação linear é um dos grandes sucessos da otimização. Devido à sua vasta aplicabilidade, tem havido muito interesse em encontrar algoritmos eficientes que acham as melhores soluções. O primeiro algoritmo utilizado para

Programação Linear foi o algoritmo simplex. Nos anos 1940, Dantzig desenvolveu o algoritmo simplex para problemas de programação linear na forma padrão. Este método que segue um caminho ao longo das bordas do conjunto de soluções viáveis. É sabido há muito tempo que o número de iterações necessárias pelo método simplex pode crescer exponencialmente na dimensão do problema (ver Klee V. [1], Dantzig G. [2], [3], Maculan & Fampa [4]).

No final dos anos 1990, estimulado pelo trabalho de Karmarkar [5], uma variedade de métodos de ponto interior foram desenvolvidos para Programação linear. Entre eles, temos o método projetivo, que foi estudado por Anstreicher [6, 7], Lustig [8] e Gonzaga [9]; o método de afim-escala, originalmente proposto por Dikin [10] e estudado por Barnes [11], Cavalier & Soyster [12], entre outros; e o método de caminho central, estudado por Gonzaga [13], [14] e Monteiro & Adler [15, 16].

Estes métodos de pontos interiores seguem um caminho através do interior do conjunto de soluções viáveis. Eles são baseados em teorias muito elegantes e convergem para um ótimo em tempo polinomial. Na prática, o número de iterações para problemas de grande porte é muito menor que o do método simplex, porém, cada um desses passos é muito mais difícil e complexo, e precisa da solução de sistemas lineares.

Os métodos de pontos interiores do tipo primal-dual aplicam o método de Newton às igualdades das condições de otimalidade [17], gerando um sistema linear e obtendo um sistema maior do que se trabalhássemos só com o problema original. Nossa proposta é trabalhar apenas com o problema original (1.1), sem acrescentar variáveis usando o algoritmo FDIPA (A Feasible Direction Interior Point Algorithm for Nonlinear Programming) proposto por Herskovits [18].

O FDIPA também resolve as condições de otimalidade mediante uma iteração tipo Newton, gerando dois sistemas lineares, transformando-os em sistemas duais lineares equivalentes, cuja matriz é simétrica definida positiva. Uma vez obtidos, esses sistemas duais podem ser resolvidos através de métodos diretos ou iterativos, como por exemplo, o método do gradiente conjugado [19].

Neste trabalho, para resolver os sistemas internos do FDIPA, usamos o método direto, se o problema de programação linear tiver restrições de igualdade e desigualdade (1.1), e usamos o método iterativo se o problema de programação linear só tiver restrições de desigualdade.

O algoritmo é implementado no MATLAB, com o objetivo de realizar testes numéricos de grande porte encontrados na literatura, como por exemplo, na biblioteca NETLIB [20].

No capítulo 2, definimos os conceitos principais da Programação Não-Linear e Linear necessários para a formulação do problema em estudo, em particular, as condições de otimalidade. Alem disso, fazemos uma breve descrição do método de pontos interiores.

No capítulo 3, descrevemos o Algoritmo de Pontos Interiores e Direções Viáveis (FDIPA), este é um algoritmo que resolve o problema geral de programação não-linear com restrições e que tem sido aplicado em problemas de otimização, em diversas áreas da engenharia.

No capítulo 4, apresentamos uma versão do método FDIPA dentro do marco da programação linear e seu algoritmo correspondente.

No capítulo 5, propomos um critério de parada para o método do gradiente precondicionado pelo Cholesky, usado na resolução dos sistemas lineares do FDIPA para problemas de programação linear com restrições de desigualdade.

No capítulo 6, expomos os resultados de alguns problemas teste da biblioteca NETLIB, usando o algoritmo FDIPA adaptado para programação linear, mostrando o desempenho dele.

Finalmente, no capítulo 7, apresentamos algumas conclusões decorrentes deste trabalho e sugestões para futuros trabalhos.

Capítulo 2

Preliminares

Neste capítulo abordamos conceitos e resultados essenciais para o desenvolvimento dos capítulos subsequentes deste trabalho. As demonstrações dos resultados citados neste capítulo encontram-se nas referências indicadas.

2.1 Problema de Minimização

O problema de otimização pode ser visto como um problema de decisão que envolve encontrar o melhor vetor $x \in R^n$ das variáveis de decisão sobre todos os vetores possíveis em uma região (Ω do R^n). Entenda-se por melhor vetor aquele onde o menor valor de função (f) ocorre.

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{sujeito a} \\ &\qquad\qquad x \in \Omega \subseteq R^n, \end{aligned}$$

onde $f : R^n \to R$ é chamada de função objetivo e Ω é frequentemente definido por igualdades e/ou desigualdades.

Seja o problema de otimização não linear dado por

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{sujeito a} \\ &\qquad\qquad g(x) \leq 0; \\ &\qquad\qquad h(x) = 0; \end{aligned} \tag{2.1}$$

onde $x \in R^n$ é o vetor das variáveis do problema, $f : R^n \to R$ é a função objetivo, $g : R^n \to R^m$ é a função que define as restrições de desigualdade, e $h : R^n \to R^p$ é a função que define as restrições de igualdade. As funções f, g e h são contínuas diferenciáveis.

Definição 2.1 *O conjunto Ω será chamado de conjunto viável do problema* 2.1

$$\Omega = \{x \in R^n \mid g(x) \leq 0,\ h(x) = 0\},$$

os pontos de Ω serão chamados de pontos viáveis.

Definição 2.2 *Dizemos que um ponto $\bar{x} \in \Omega$ é*

1. *minimizador global de* 2.1*, se*

$$f(\bar{x}) \leq f(x) \quad \text{para todo } x \in \Omega;$$

2. *minimizador local de* 2.1*, se existe uma vizinhança U de $\bar{x}$ tal que*

$$f(\bar{x}) \leq f(x) \quad \text{para todo } x \in \Omega \cap U.$$

Definição 2.3 *Dizemos que $d \in R^n$ é uma direção viável em relação ao conjunto Ω no ponto $\bar{x} \in \Omega$, quando existe $\tau > 0$ tal que*

$$\bar{x} + td \in \Omega \quad \text{para todo } t \in [0, \tau].$$

Definição 2.4 *Dizemos que $d \in R^n$ é uma direção de descida de $f : R^n \to R$ no ponto $\bar{x} \in R^n$, se existe $\tau > 0$ tal que*

$$f(\bar{x} + td) < f(\bar{x}) \quad \text{para todo } t \in (0, \tau].$$

Sendo $f(x)$ diferenciável em x e $d^t \nabla f(x) < 0$, então d é uma direção de descida de $f(x)$.

Definição 2.5 *O conjunto dos índices das restrições ativas no ponto $\bar{x} \in \Omega$ será denotado por*

$$I(\bar{x}) = \{i = 1,\ \ldots, m \mid g_i(\bar{x}) = 0\}.$$

Não incluímos nesta definição as restrições de igualdade, porque todas elas sempre são ativas em todo ponto viável.

Definição 2.6 *Dado um ponto $\bar{x} \in \Omega$, a condição de regularidade de independência linear das restrições é satisfeita em $\bar{x}$, se os gradientes das restrições de igualdade e de desigualdade ativas são linearmente independentes, ou seja, o conjunto $\{\nabla h_i(\bar{x}),\ i = 1,\ \ldots p\} \cup \{\nabla g_i(\bar{x}),\ i \in I(\bar{x})\}$ é linearmente independente.*

Os pontos que satisfazem a condição de regularidade de independência linear das restrições são chamados pontos regulares.

O espaço tangente em $\bar{x}$ é

$$T = \{y \ / \ \nabla g_i^t(\bar{x})y < 0 \text{ para } i \in I(\bar{x}) \ \text{ e } \ \nabla h_i^t(\bar{x})y = 0 \ \text{ para } i = 1, ..., p\}$$

Teorema 2.1 *(Condições Necessárias de Primeira Ordem Karush-Kuhn-Tucker) Seja $\bar{x}$ um ponto regular; temos que este ponto é um mínimo local do problema (2.1). Então existe um vetor $\bar{\lambda} \in R^m$ e um vetor $\bar{\mu} \in R^p$ tal que:*

$$\begin{aligned}
\nabla f(\bar{x}) + \nabla g(\bar{x})\bar{\lambda} + \nabla h(\bar{x})\bar{\mu} &= 0 && (2.2)\\
g_i(\bar{x})\bar{\lambda}_i &= 0 \quad i=1,...,m && (2.3)\\
h(\bar{x}) &= 0 && (2.4)\\
g(\bar{x}) &\leq 0 && (2.5)\\
\bar{\lambda} &\geq 0. && (2.6)
\end{aligned}$$

Os vetores $\bar{\lambda}$ e $\bar{\mu}$ são conhecidos como os multiplicadores de Lagrange.

Teorema 2.2 *(Condição de Segunda Ordem) Seja $\bar{x}$ um ponto regular; temos que este ponto é um mínimo local do problema (2.1). Então existe um vetor $\bar{\lambda} \in R^m$ e um vetor $\bar{\mu} \in R^p$ tal que o resultado do Teorema 2.1 é verdadeiro e a matriz*

$$H(\bar{x}, \bar{\lambda}, \bar{\mu}) = \nabla^2 f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla^2 g_i(\bar{x}) + \sum_{i=1}^{p} \bar{\mu}_i \nabla^2 h_i(\bar{x})$$

é semidefinida positiva no espaço tangente, isto é, $y^t H(\bar{x}, \bar{\lambda}, \bar{\mu})y \geq 0$ para todo $y \in T$.

Teorema 2.3 *(Condição de Suficiência de Segunda Ordem) Seja $\bar{x}$ tal que $g(\bar{x}) \leq 0$ e $h(\bar{x}) = 0$. Sejam os vetores $\bar{\lambda} \in R^m$ (com $\bar{\lambda} \geq 0$) e $\bar{\mu} \in R^p$ tal que*

$$\nabla f(\bar{x}) + \nabla g(\bar{x})\bar{\lambda} + \nabla h(\bar{x})\bar{\mu} = 0$$

e $H(\bar{x}, \bar{\lambda}, \bar{\mu})$ é definida positiva no espaço tangente. Então, $\bar{x}$ é um mínimo local estrito.

2.2 Programação Linear

Um problema de programação linear pode ser definido da seguinte maneira:

$$\begin{aligned} &\text{minimize} && c^t x && (2.7)\\ &\text{sujeito a} && && \\ & && A_1 x = b_1 && (2.8)\\ & && A_2 x \leq b_2, && (2.9)\\ & && x \geq 0 && (2.10) \end{aligned}$$

onde $c \in R^n$, $A_1 \in R^{p \times n}$, $A_2 \in R^{q \times n}$, $b_1 \in R^p$ e $b_2 \in R^q$ são dados; e os $x \in R^n$ representam as variáveis de decisão da função linear a ser minimizada em (2.7). As restrições de não negatividade (2.10) são conhecidas como triviais.

Cada restrição $i = 1, 2, ..., q$ de (2.9) pode ser substituída com o acréscimo de uma variável $s_i \geq 0$ (denominada variável de folga), por uma restrição de igualdade e uma restrição trivial:

$$\begin{aligned} &\text{minimize} && c^t x \\ &\text{sujeito a} && \\ & && A_1 x = b_1 \\ & && A_2 x + s = b_2 \\ & && x \geq 0, \\ & && s \geq 0, \end{aligned} \qquad (2.11)$$

Logo, é sempre possível expressarmos um problema de programação linear da seguinte forma:

$$\begin{aligned} &\text{minimize} && c^t x \\ &\text{sujeito a} && \\ & && Ax = b \\ & && x \geq 0, \end{aligned} \qquad (2.12)$$

Esta formulação é denominada problema primal e está associada a um problema dual:

$$\begin{aligned} &\text{maximize} && b^t y \\ &\text{sujeito a} && \\ & && A^t y + z = c \\ & && z \geq 0, \end{aligned} \qquad (2.13)$$

Proposição 2.1 *Seja $A \in R^{m \times n}$ uma matriz de posto igual a m, $m < n$. Então, a*

matriz AA^t é não singular.

Demonstração. Ver em Gonzaga [13].

A teoria da dualidade explica a relação entre os dois problemas (2.12) e (2.13). Um dos resultados é a adaptação das condições de KKT para os dois problemas Primal e Dual.

Teorema 2.4 *(Condições de KKT para o Primal e Dual) (x, y, z) são soluções ótimas para o par primal-dual se, e somente se, as seguintes condições são satisfeitas:*

$$\begin{aligned} Ax &= b, && (2.14)\\ A^t y + z &= c, && (2.15)\\ x_i z_i &= 0 \quad i = 1, ..., n, && (2.16)\\ (x, z) &\geq 0. && (2.17) \end{aligned}$$

Demonstração. Ver em Wright [17].

O teorema acima e o método de Newton que apresentamos no capítulo seguinte são frequentemente usados na descrição e análise dos algoritmos utilizados para programação linear.

2.3 Método de Pontos Interiores

Estes métodos de pontos interiores seguem um caminho através do interior do conjunto de soluções viáveis. Todos os do tipo primal-dual são baseados na resolução do sistema trabalhando simultaneamente, na forma primal e dual, ou seja, ao encontrar uma solução para o dual, encontra-se para o primal.

Se existe a solução do primal e dual do problema de programação linear, então satisfaz as condições de otimalidade dadas pelo teorema 2.4

$$F(x, y, z) = \begin{bmatrix} Ax - b \\ A^t y + z - c \\ XZe \end{bmatrix} = 0, \quad (x, z) \geq 0 \qquad (2.18)$$

onde $X = diag(x)$, $Z = diag(z)$ e $e = (1, 1, ..., 1)^t \in R^n$.

Este método é um processo iterativo, no qual precisamos de uma direção e um tamanho do passo. Para o cálculo da direção $(dx, dy, dz)^t$, aplicamos o método de Newton ao sistema (2.18) e resolvemos o sistema linear seguinte:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - z - A^t y \\ -XZe \end{bmatrix} \tag{2.19}$$

A direção calculada pelo sistema (2.19) é denominada direção afim-escala (método de pontos interiores dual-primal afim-escala).

Uma desvantagem do método afim-escala é que ele mantém as componentes de (x, z) movendo-se muito próximas da fronteira $(x, z) = 0$. Direções de busca calculadas muito próximas desta fronteira tendem a ser distorcidas devido a erros de arredondamentos. Para evitar essa aproximação, se realiza uma perturbação (μ) nas condições de complementariedade (2.16).

Então, do sistema (2.18) substituímos $XZe = 0$ pela equação parametrizada $XZe = \psi e$, ψ é um parâmetro positivo. Então, aplicando o método de Newton ao sistema perturbado, obtemos

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - z - A^t y \\ \psi e - XZe \end{bmatrix} \tag{2.20}$$

Esta alteração nas condições de otimalidade gera outro método chamado método de pontos interiores primal-dual seguidor de caminho.

Ele mantém seus iterados próximos a um caminho central C, parametrizado pelo escalar $\psi > 0$. Tomar passos próximos a C faz com que a direção de busca aponte para o interior da região determinada por $(x, z) > 0$. Isso porque o parâmetro ψ pode ser visto como o parâmetro originado do problema de barreira logarítmica definido a partir de (2.12). Para mais detalhes ver [17].

Capítulo 3

(FDIPA) Algoritmo de Pontos Interiores e Direções Viáveis

Neste capítulo faremos uma abordagem sobre o algoritmo de pontos interiores e direções viáveis FDIPA, apresentado por Herkovits em [18], o qual foi desenvolvido, buscando solucionar o problema geral de programação não-linear dado pela formulação (2.1). Primeiramente, apresentaremos os métodos de tipo Newton e Quase-Newton, necessários para o desenvolvimento do FDIPA (Feasible Direction interior Point Algorithm).

3.1 Algoritmo de Newton

Devido á sua velocidade de convergência e eficiência, o método de Newton tem sido bastante utilizado para o propósito de obter soluções de equações não-lineares. Consiste em substituir a função não-linear por sua aproximação linear. Mais precisamente, queremos resolver o sistema de equações

$$F(y) = 0, \tag{3.1}$$

onde $F : R^n \to R^n$ é uma função contínua com derivadas contínuas.

Dado um ponto inicial y^0 do domínio de F, tal que $\nabla F(y^0)$ seja não nula; construímos a aproximação linear de F, isto é,

$$F(y) \approx F(y^0) + \nabla^t F(y^0)(y - y^0).$$

O ponto y que anula a aproximação linear da função F é o ponto que verifica:

$$F(y^0) + \nabla^t F(y^0)(y - y^0) = 0.$$

Definimos $d = y - y^0$ como uma direção que nos aproxima à solução do problema (3.1), ou seja,

$$y = y^0 + d, \tag{3.2}$$

onde o vetor d pode ser obtido resolvendo o seguinte sistema linear:

$$\nabla^t F(y^0)d = -F(y^0).$$

Como foi utilizada uma aproximação linear da função F, o ponto y definido pela equação (3.2) não é, em geral, solução da equação (3.1). Neste caso, deve ser feita uma nova aproximação linear, agora no ponto y, e continuar com o mesmo procedimento, até satisfazer algum critério de convergência.

Algoritmo 3.1 *(Método de Newton)*

Dados: $y \in R^n$ tal que $F(y) \approx 0$.

Passo 1: Teste de convergência.

Passo 2: Determinação da direção de Newton.
Resolver o sistema linear em d:

$$\nabla^t F(y)d = -F(y)$$

Passo 3: Atualização.
Defina

$$y := y + d.$$

Vá para o passo 1.

Neste algoritmo, é necessário calcular o Jacobiano $\nabla F(y)$ e resolver um sistema linear a cada iteração, o que poderia ser muito custoso em termos de esforço computacional. Não obstante, a convergência global não é garantida.

O Jacobiano do sistema, em termos de otimização, equivale ao Hessiano das funções envolvidas, e seu cálculo analítico, dependendo do problema, pode não ser possível.

Com o objetivo de reduzir o esforço computacional, o método Quase-Newton

define uma aproximação do Jacobiano ou de sua inversa, usando informações colhidas ao longo de sucessivas iterações.

3.2 Algoritmo Quase-Newton

A idéia básica do método Quase-Newton é a seguinte: Seja B^k a aproximação atual de $\nabla F(y^k)$, então a nova aproximação B^{k+1} é obtida de

$$B^{k+1} = B^k + \Delta B^k. \tag{3.3}$$

Sendo que

$$F(y^{k+1}) - F(y^k) \approx \nabla^t F(y^k)(y^{k+1} - y^k),$$

ΔB^k é definida de forma que

$$F(y^{k+1}) - F(y^k) = [B^{k+1}]^t(y^{k+1} - y^k). \tag{3.4}$$

Substituições da equação (3.3) em (3.4) definem n condições a serem satisfeitas por ΔB^k. Sendo que ΔB^k tem n^2 elementos, estas condições não são suficientes para determiná-lo. Várias regras de atualização para B^{k+1} foram propostas (ver Luenberger [21]), sendo a regra de Broyden a mais bem sucedida,

$$B^{k+1} = B^k + (\gamma - B^k\delta)\delta^t/\delta^t\delta, \tag{3.5}$$

onde $\delta = y^{k+1} - y^k$ e $\gamma = F(y^{k+1}) - F(y^k)$.

Algoritmo 3.2 *Método Quase-Newton*

Dados: $y \in R^n$ e $B \in R^{n\times n}$ iniciais.

Passo 1: Cálculo da direção d.

Resolver o sistema linear em d,

$$B^t d = -F(y).$$

Passo 2: Atualizar.

$$y := y + d, \text{ e}$$

$$B := B + \Delta B.$$

Vá para o passo 1.

No Passo 2, B pode ser atualizado mediante (3.5) ou outras regras de atualização a serem definidas.

O método Quase-Newton tem a vantagem de evitar o cálculo de $\nabla F(y)$; para conseguirmos convergência local, a matriz B deverá ser uma boa aproximação de $\nabla F(y)$.

3.3 Descrição do Algoritmo FDIPA

Este algoritmo é um método iterativo, dado pela regra $x^{k+1} = x^k + t^k d^k$, onde x^k é o ponto da iteração atual, t^k é o passo e d^k a direção de busca na iteração k. Ele gera uma sequência de pontos interiores $\{x^n\}_{n \in N}$ totalmente contida ao interior da região viável Ω, a partir de um dado ponto inicial x_0, e que converge globalmente com ordem superlinear para um ponto de KKT do problema (2.1), após um número infinito de iterações, reduzindo a função objetivo $f(x)$ em cada iteração.

Primeiramente, desxcreveremos o algoritmo FDIPA para problemas de programação não linear com restrições de desigualdade. Seja o problema

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{sujeito a} \\ &\qquad\qquad g(x) \leq 0; \end{aligned} \tag{3.6}$$

onde $f : R^n \to R$ e $g : R^n \to R^m$ são funções continuamente diferenciáveis e o conjunto viável é $\Delta = \{x \in R^n \mid g(x) \leq 0\}$.

As condições de primeira orden de KKT correspondentes ao problema (3.6) são:

$$\nabla f(x) + \nabla g(x)\lambda = 0 \tag{3.7}$$

$$g_i(x)\lambda_i = 0 \quad \text{i=1,...,m} \tag{3.8}$$

$$g(x) \leq 0 \tag{3.9}$$

$$\lambda \geq 0. \tag{3.10}$$

3.3.1 Direção de Descida

A definição da direção de busca do algoritmo FDIPA é inspirada na aplicação do método de Newton (descrito na seção 3.1) no sistema não linear das equações (3.7) e (3.8).

Com efeito, tomando x_0 e λ_0 como as novas estimativas de x e λ respectiva-

mente, obtemos:

$$\begin{bmatrix} B & \nabla g(x) \\ \Lambda \nabla g(x)^t & G(x) \end{bmatrix} \begin{bmatrix} x_0 - x \\ \lambda_0 - \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f(x) + \nabla g(x)\lambda \\ G(x)\lambda \end{bmatrix},$$

onde Λ^k é uma matriz diagonal $m \times m$, com $[\Lambda^k]_{ii} = \lambda_i^k$; G^k é uma matriz diagonal $m \times m$, com $[G^k]_{ii} = g_i^k$; B^k é uma matriz simétrica definida positiva de ordem $n \times n$.

Pode ser a matriz identidade, a própria hessiana $H(x, \lambda, \mu)$ ou uma aproximação dela obtida por alguma técnica de quase-Newton. No entanto, B deve ser sempre definida positiva para que se garanta a convergência global do problema.

Se definimos $d_0 = x_0 - x$, então

$$Bd_0 + \nabla g(x)\lambda_0 = -\nabla f(x), \tag{3.11}$$

$$\Lambda \nabla g^t(x) d_0 + G(x)\lambda_0 = 0 \tag{3.12}$$

Herskovits provou que d_0 é uma direção de descida para f. No entanto, d_0 pode não ser uma direção viável pois, desenvolvendo a equação (3.12), temos:

$$\lambda_i \nabla g_i^t(x) d_0 + g_i(x)\lambda_{0i} = 0; \quad \text{i=1,2, ..., m.}$$

Então, se $g_i(x) \to 0$,

$$\nabla g_i^t(x) d_0 \to 0,$$

ou seja, quando alguma restrição se aproxima de zero, d_0 tende a uma direção tangente ao conjunto viável.

3.3.2 Direção de Descida Viável

Para evitar este efeito, adiciona-se um vetor negativo no lado direito da equação (3.12), produzindo o efeito de deflexão de d_0 para o interior do conjunto viável. Define-se um novo sistema em $(d, \bar{\lambda})$:

$$Bd + \nabla g(x)\bar{\lambda} = -\nabla f(x), \tag{3.13}$$

$$\Lambda \nabla g^t(x) d + G(x)\bar{\lambda} = -\rho\lambda \tag{3.14}$$

onde $\rho > 0$, d é a nova direção e, $\bar{\lambda}$ é a nova estimativa de λ. Neste caso, (3.14) é equivalente a:

$$\lambda_i \nabla g_i^t(x) d + g_i(x)\bar{\lambda_i} = -\rho\lambda; \quad \text{i=1,2, ..., m.}$$

e, consequentemente,

$$\nabla g_i^t(x)d = -\rho\lambda < 0$$

para as restrições ativas. Logo, d é uma direção viável.

A deflexão de d_0 é proporcional ao ρ; é possível garantir que d também seja uma direção de descida, se ρ for escolhido convenientemente (ver Herskovits [18]).

Definindo

$$d = d_0 + \rho d_1,$$

$$\bar{\lambda} = \lambda_0 + \rho\lambda_1,$$

e substituindo no sistema de equações formado por (3.13) e (3.14), se d_0 e d_1 são L.I., encontramos

$$\begin{bmatrix} B & \nabla g(x) \\ \Lambda\nabla g^t(x) & G(x) \end{bmatrix} \begin{bmatrix} d_0 & d_1 \\ \lambda_0 & \lambda_1 \end{bmatrix} = - \begin{bmatrix} \nabla f(x) & 0 \\ 0 & \lambda \end{bmatrix}.$$

Resolvendo o sistema, obtemos a direção d e $\bar{\lambda}$. Desta forma, é facil estabelecer limites para ρ.

Como $d_0^t\nabla f(x) < 0$, impomos:

$$d^t\nabla f(x) \leq \alpha d_0^t\nabla f(x), \quad \alpha \in (0,1), \tag{3.15}$$

substituindo d por $d_0 + \rho d_1$ na equação acima obtemos

$$\rho d_1^t\nabla f(x) \leq (\alpha - 1)d_0^t\nabla f(x). \tag{3.16}$$

Se $d_1^t\nabla f(x) < 0$, a equação (3.16) se verifica para todo $\rho > 0$, em caso contrário,

$$\rho \leq (\alpha - 1)\frac{d_0^t\nabla f(x)}{d_1^t\nabla f(x)}$$

garante (3.15); portanto, podemos afirmar que d é uma direção viável e de descida.

Finalmente, para encontrar um novo ponto primal viável e uma diminuição satisfatória da função objetivo, procede-se à busca linear inexata, como Armijo ou Wolfe, descritos em Luenberger.

Algoritmo 3.3 *Algoritmo FDIPA para o problema (3.6)*

Dados: $x \in \Delta^0$, $\lambda \in R^m_{++}$, $B \in R^{n\times n}$ simétrica positiva definida, $\alpha \in (0,1)$, $v \in (0,1)$, $\eta \in (0,1)$.

Passo 1: Determinação da direção de descida.

(i) Resolver o sistema linear em $(d_0,\ \lambda_0)$:

$$\begin{bmatrix} B & \nabla g(x) \\ \Lambda\nabla g^t(x) & G(x) \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ 0 \end{bmatrix},$$

Se $d_0 = 0$ parar.

(ii) Resolver o sistema linear em $(d_1,\ \lambda_1)$:

$$\begin{bmatrix} B & \nabla g(x) \\ \Lambda\nabla g^t(x) & G(x) \end{bmatrix} \begin{bmatrix} d_1 \\ \lambda_1 \end{bmatrix} = - \begin{bmatrix} 0 \\ \lambda \end{bmatrix},$$

onde $d_1 \in R^n$, $\lambda_1 \in R^m$.

(iii) Se $d_1^t \nabla f(x) > 0$, então se define:

$$\rho = inf \left[\|d_0\|^2; \frac{(\alpha-1)d_0^t \nabla f(x)}{d_1^t \nabla f(x)} \right]$$

caso contrário, se define:

$$\rho = \|d_0\|^2$$

(iv) Calcular d

$$d = d_0 + \rho d_1, \text{ e}$$
$$\bar{\lambda} = \lambda_0 + \rho\lambda_1$$

Passo 2: Busca Linear

Calcula t, o primeiro número da sequência $\{1, v, v^2, v^3, ...\}$ satisfaz

$$\begin{aligned} &f(x+td) \le \phi_c(x) + t\eta d^t \nabla f(x), \\ &g_i(x+td) < 0 \quad \text{se } \bar{\lambda}_i \ge 0, \text{ ou} \\ &g_i(x+td) \le g_i(x), \quad \text{caso contrário.} \end{aligned}$$

Passo 3: Atualização.

(i) Assumir

$$x := x + td$$

e definir um novo valor para: $\lambda > 0$ e B simétrica positiva definida.

(ii) Vá para o passo 1.

Agora apresentaremos o Algoritmo FDIPA para o problema geral de programação não linear:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{sujeito a} & \\ & g(x) \leq 0; \\ & h(x) = 0; \end{array} \tag{3.17}$$

onde $f : R^n \to R$, $g : R^n \to R^m$ e $h : R^n \to R^p$ são funções continuamente diferenciáveis.

As restrições de igualdade devem ser necessariamente ativas na solução. Para tal, utiliza-se uma técnica que inclui estas restrições. Então, para que elas sejam satisfeitas, pode ser necessário um acréscimo da função objetivo, mediante a seguinte função potencial:

$$\phi(x, c) = f(x) + \sum_{i=1}^{p} c_i |h_i(x)|,$$

onde c_i são escalares positivos. Zangwill [22] e Mayne e Polack [23] utilizaram esta função potencial e demonstraram que existe um c finito, tal que o mínimo de $\phi(x, c)$, sujeito unicamente às restrições de desigualdade, ocorre na solução do problema (3.17). Desta maneira, o uso de $\phi(x, c)$ como função potencial tem a vantagem de não precisar de parâmetros de penalidade crescendo indefinidamente.

No entanto, $\phi(x, c)$ não é continuamente diferenciável, uma vez que não tem derivada única nos pontos onde as restrições de igualdade são ativas ($h_i(x) = 0$). Na busca linear, resulta necessário usar técnicas que evitem estes pontos de não diferenciabilidade.

Com este objetivo, no início do proceso de otimização, as restrições de igualdade são redefinidas de maneira que todas elas tenham o mesmo sinal. A busca linear utiliza a função potencial $\phi(x, c)$, e exige que os pontos gerados fiquem sempre do mesmo lado das restrições ativas, inclusive as igualdades, evitando assim, pontos onde a função potencial não é diferenciável.

Desta maneiria, o conjunto viável e seu interior são neste caso:

$$\Omega = \{x \in R^n | \, g_i(x) \leq 0, \; h_j(x) \leq 0\}, \;\; i = 1, 2, ..., m : \;\; j = 1, 2, ..., p.$$

$$\Omega^0 = \{x \in R^n | \, g_i(x) < 0, \; h_j(x) < 0\}, \;\; i = 1, 2, ..., m : \;\; j = 1, 1, ..., p.$$

A descrição mais detalhada deste algoritmo pode ser encontrada em [18].

Algoritmo 3.4 *FDIPA*

Dados: $x \in \Omega^0$, $\lambda \in R^m$, $\lambda_i > 0$, $\mu_i > 0$, $B \in R^{n \times n}$ simétrica positiva definida, $\alpha \in (0,1)$, $c \in R^p$, $c_i = 0$, $v \in (0,1)$, $\eta \in (0,1)$.

Passo 1: Determinação da direção de descida.

(i) Resolver o sistema linear em $(d_0,\ \lambda_0,\ \mu_0)$:

$$\begin{bmatrix} B & \nabla g(x) & \nabla h(x) \\ \Lambda \nabla g^t(x) & G(x) & 0 \\ \nabla h^t(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \\ \mu_0 \end{bmatrix} = - \begin{bmatrix} \nabla f(x) \\ 0 \\ h(x) \end{bmatrix},$$

Se $d_0 = 0$ parar.

(ii) Resolver o sistema linear em $(d_1,\ \lambda_1,\ \mu_1)$:

$$\begin{bmatrix} B & \nabla g(x) & \nabla h(x) \\ \Lambda \nabla g^t(x) & G(x) & 0 \\ \nabla h^t(x) & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ \lambda_1 \\ \mu_1 \end{bmatrix} = - \begin{bmatrix} 0 \\ \lambda \\ 0 \end{bmatrix},$$

onde $d_1 \in R^n$, $\lambda_1 \in R^m$, $\mu \in R^p$.

(iii) Se $c_i \leq \|\mu_0\|$, faca $c_i = 1.2\|\mu_0\|$, para $i = 1, \ldots, p$.

(iv) Seja:

$$\phi(x,c) = f(x) + c^t \|h(x)\|_2$$

se $d_1^t \nabla \phi(x,c) > 0$, então se define:

$$\rho = inf \left[\|d_0\|^2; \frac{(\alpha - 1) d_0^t \nabla \phi(x,c)}{d_1^t \nabla \phi(x,c)} \right]$$

em caso contrário, se define:

$$\rho = \|d_0\|^2$$

(v) Calcular d

$$d = d_0 + \rho d_1, \text{ e}$$
$$\bar{\lambda} = \lambda_0 + \rho \lambda_1$$

Passo 2: Busca Linear

Calcula t, o primeiro número da sequência $\{1, v, v^2, v^3, ...\}$ satisfaz

$$\begin{aligned}
&\phi(x + td) \leq \phi_c(x) + t\eta d_0^t \nabla\phi_c(x), \\
&h(x + td) \leq 0 \quad \text{e} \\
&g_i(x + td) < 0 \quad \text{se } \bar{\lambda} \geq 0, \text{ ou} \\
&g_i(x + td) \leq g_i(x), \quad \text{caso contrário.}
\end{aligned}$$

Passo 3: Atualização.

(i) Assumir

$$x := x + td$$

e definir um novo valor para: $\lambda > 0$ e B simétrica positiva definida.

(ii) Vá para o passo 1.

Capítulo 4

Implementação de Programação Linear com FDIPA

Neste capítulo adaptaremos o algoritmo FDIPA para os problemas de programação linear.

4.1 Problema de Programação Linear com Restrições de Desigualdades

Nesta seção, usaremos o algoritmo FDIPA para resolver o problema de programação linear com restrições de desigualdade, representado da seguinte maneira:

$$\begin{aligned} &\text{minimize} \quad c^t x \\ &\text{sujeito a} \\ &\qquad\qquad Ax \leq b; \end{aligned} \tag{4.1}$$

onde $A \in R^{m \times n}$, $b \in R^m$, $c \in R^n$ e $x \in R^n$.

Além disso, o conjunto viável é

$$\Omega = \{x \in R^n / Ax - b \leq 0\}$$

e

$$\Omega^0 = \{x \in R^n / Ax - b < 0\}$$

é o interior;

$$L(x, \lambda) = c^t x + \lambda^t (Ax - b)$$

é o Lagrangiano; e a Hessiana $H(x, \lambda) = 0$, já que, as funções objetivo e as restrições são lineares. Ademais, $G(x)$ é uma matriz diagonal tal que:

$$G_{ii}(x) = (Ax - b)_i.$$

Também usaremos a definição $\Lambda = \text{diag}(\lambda_1, \lambda_2 \ldots \lambda_m)$.

As condições de otimalidade de KKT aplicadas ao problema (4.1) para encontrar a direção viável são:

$$\begin{aligned} c + A^t\lambda &= 0 && (4.2)\\ G(x)\lambda &= 0 && (4.3)\\ A_i x - b_i &\leq 0 \qquad \text{i=1, ... ,m.} && (4.4)\\ \lambda &\geq 0, && (4.5) \end{aligned}$$

O algoritmo FDIPA, baseado no conceito de multiplicadores de Lagrange, constrói uma sequência de pontos $\{x^k\}$, começando com um ponto estritamente viável e convergindo a um ponto ótimo.

4.1.1 Direção viável

Para achar uma direção viável resolvemos as equações (4.2) e (4.3) nas condições de KKT. Utilizando o método de Newton em (x, λ), obtemos o seguinte sistema:

$$\begin{bmatrix} H(x^k, \lambda_0^k) & A^t \\ \Lambda A & G(x^k) \end{bmatrix} \begin{bmatrix} x^{k+1} - x^k \\ \lambda^{k+1} - \lambda^k \end{bmatrix} = - \begin{bmatrix} c + A^t\lambda^k \\ G(x^k)\lambda^k \end{bmatrix},$$

onde consideramos $d_0 = x^{k+1} - x^k$ e $\lambda_0 = \lambda^{k+1}$.

A Hessiana do sistema de equações anterior é zero. Porém, se considerarmos isso, teremos um mau condicionamiento no sistema de equações. Portanto, consideraremos, para propósitos de iteração, que a matriz $H(x^k, \lambda_0^k) = B$; em nosso caso, para a implementação, usamos $B = \beta I_n$, onde I_n é a identidade em n dimensões e $\beta \in R$.

Escrevemos o sistema de equações da seguinte forma:

$$\begin{bmatrix} B & A^t \\ \Lambda A & G(x) \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \end{bmatrix} = - \begin{bmatrix} c \\ 0 \end{bmatrix} \qquad (4.6)$$

A partir de (4.6), eliminando d_0, chegamos a um sistema com matriz definida positiva que será resolvido por gradiente conjugado precondicionado com Cholesky truncado (que será estudado no capítulo seguinte), o qual deve proporcionar uma direção viável de descida (d_0).

Lema 4.1 *O vector d_0, definido no sistema (4.6), é uma direção viável de descida.*

Prova Ver Herskovits [18].

4.1.2 Tamanho de passo

Outro passo importante para resolver nosso problema de programação linear (4.1) com o FDIPA é encontrar o tamanho de passo (t) para cada iteração. Para tal, calcularemos o tamanho de passo em cada iteração de forma numérica fazendo com que as restrições do problema (4.1) sejan satisfeitas.

A sequência de pontos que gera o algoritmo FDIPA fica no interior do conjunto viável Ω; portanto, se $x^k \in \Omega^0$, então o seguinte ponto $x^{k+1} = x^k + td_0$ também deve estar em Ω^0, ou seja,

$$Ax^{k+1} < b,$$

substituindo x^{k+1} na equação acima obtemos,

$$Ax^k + tAd_0 < b,$$

para cada desigualdade $i = 1, 2, ..., m$ temos que

$$t_i(Ad_0)_i < (b - Ax^k)_i \tag{4.7}$$

logo, se $(Ad_0)_i \leq 0$, então t_i é ilimitado.
Caso contrário, da equação (4.7) obtemos que t_i é limitado

$$t_i < \frac{(b - Ax^k)_i}{(Ad_0)_i}.$$

Assim, para garantir que x^{k+1} fique no interior da região viável, o valor de t é dado desta forma:

$$t = \gamma \min_{1 \leq i \leq m} \left\{ \frac{(b - Ax^k)_i}{(Ad_0)_i} \right\},$$

onde $\gamma \in (0, 1)$. Em nossa implementação, consideramos $\gamma = 0.9995$.

Algoritmo 4.1 *FDIPA para resolver o problema (4.1)*

Dados: $x \in \Omega^0$, $\lambda \in R^m_{++}$, $\gamma \in (0,1)$

Passo 1: Determinação da direção de descida.

Resolver o sistema linear em (d_0, λ_0):

$$\begin{bmatrix} \beta I & A^t \\ \Lambda A & G(x) \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \end{bmatrix} = - \begin{bmatrix} c \\ 0 \end{bmatrix}$$

Se $d_0 = 0$ parar.

Passo 2: Busca linear

Calcule $t > 0$, se $Ax - b < 0$

$$t = \gamma \min_{1 \leq i \leq m} \left\{ \frac{(b - Ax)_i}{(Ad_0)_i} \right\}$$

Passo 3: Atualização.

(i) Assumir:

$$x = x + td$$

(ii) Vá para o passo 1.

4.2 Problema de Programação Linear com Restrições de Igualdade e Desigualdades

Nesta seção, aplicaremos o algoritmo FDIPA para resolver o problema geral de programação linear considerado. Este problema é representado da seguinte maneira:

$$\begin{array}{ll} \text{minimize} & c^t x \\ \text{sujeito a} & \\ & A_1 x = b_1; \\ & A_2 x \leq b_2; \end{array} \tag{4.8}$$

onde $A_1 \in R^{p \times n}$, $b_1 \in R^p$, $A_2 \in R^{m \times n}$, $b_2 \in R^m$ e $c \in R^n$. A matriz A_1 tem posto completo.

Também usaremos as definições onde Λ e $G(x)$ são matrizes diagonais, tais que:

$$G_{ii}(x) = (A_2 x - b_2)_i$$

$$\Lambda_{ii} = \lambda_i$$

Usaremos o método de FDIPA para achar a solução do problema (4.8). As condições de otimalidade de KKT para encontrar a direção viável são:

$$\begin{aligned} c + A_2^t\lambda + A_1^t\mu &= 0 && (4.9)\\ G(x)\lambda &= 0 && (4.10)\\ A_1x - b_1 &= 0 && (4.11)\\ A_2x - b_2 &\leq 0 && (4.12)\\ \lambda &\geq 0. && (4.13) \end{aligned}$$

Aplicamos o método de Newton ao sistema composto pelas equações (4.9), (4.10) e (4.11), gerando um novo sistema de equações

$$\begin{bmatrix} B & A_2^t & A_1^t \\ \Lambda A_2^t & G(x) & 0 \\ A_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \\ \mu_0 \end{bmatrix} = - \begin{bmatrix} c \\ 0 \\ A_1x - b_1 \end{bmatrix},$$

onde B, no princípio, é a matriz identidade $(n \times n)$ e depois, em cada iteração, ela faz uma atualização (em nosso algoritmo, usamos $B^{k+1} := 0.00001B^k$).

Para o problema (4.8), usaremos a deflexão de d_0, definindo um novo sistema em (d_1, λ_1, μ_1):

$$\begin{bmatrix} B & A_2^t & A_1^t \\ \Lambda A_2 & G(x) & 0 \\ A_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ \lambda_1 \\ \mu_1 \end{bmatrix} = - \begin{bmatrix} 0 \\ \lambda \\ 0 \end{bmatrix}.$$

Resolvendo o sistema, obtemos a direção $d = d_0 + \rho d_1$, onde ρ é

$$\rho \leq (\alpha - 1)\frac{d_0^t \nabla f(x)}{d_1^t \nabla f(x)},$$

se $d_1^t \nabla f(x) < 0$. Em nossa implementação, consideramos $\alpha = 0.85$.

Para iniciar o algoritmo, o ponto inicial deve satisfazer as restrições de igualdade e desigualdade. Então as iterações seguintes andarão pelas restrições ativas e não precisaremos da função potencial.

A busca linear é a mesma que usamos para resolver o problema de Programação Linear com restrições de desigualdade.

Os dois sistemas de equações são resolvidos usando o método direto.

Algoritmo 4.2 *FDIPA para resolver o problema (4.8)*

Dados: $x \in \Omega$, $\lambda \in R^m$, $\lambda_i > 0$, $\mu_i > 0$, $B = I_n$, $\alpha \in (0,1)$, $\varsigma_i = 0$.

Passo 1: Determinação da direção de descida.

(i) Resolver o sistema linear em $(d_0,\ \lambda_0,\ \mu_0)$:

$$\begin{bmatrix} B & A_2^t & A_1^t \\ \Lambda A_2^t & G(x) & 0 \\ A_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \\ \mu_0 \end{bmatrix} = - \begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix},$$

Se $d_0 = 0$ parar.

(ii) Resolver o sistema linear em $(d_1,\ \lambda_1,\ \mu_1)$:

$$\begin{bmatrix} B & A_2^t & A_1^t \\ \Lambda A_2 & G(x) & 0 \\ A_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ \lambda_1 \\ \mu_1 \end{bmatrix} = - \begin{bmatrix} 0 \\ \lambda \\ 0 \end{bmatrix},$$

onde $d_1 \in R^n$, $\lambda_1 \in R^m$, $\mu \in R^p$.

(iii) se $d_1^t c > 0$, então se define:

$$\rho = inf \left[\|d_0\|^2; \frac{(\alpha - 1) d_0^t c}{d_1^t c} \right]$$

em caso contrário, se define:

$$\rho = \|d_0\|^2$$

(iv) Calcular d

$$d = d_0 + \rho d_1, \text{ e}$$
$$\bar{\lambda} = \lambda_0 + \rho \lambda_1$$

Passo 2: Busca Linear

Calcule $t > 0$, se $Ax - b < 0$

$$t = \gamma \min_{1 \le i \le m} \left\{ \frac{(b - Ax)_i}{(Ad)_i} \right\}$$

Passo 3: Atualização.

(i) Assumir

$$x := x + td$$

$$B := 0.00001B$$

e definir um novo valor para: $\lambda > 0$.

(ii) Vá para o passo 1.

Capítulo 5

Resolução dos Sistemas Lineares Internos do FDIPA

Neste capítulo, os sistemas internos do FDIPA para programação linear são apresentados de forma detalhada.

Os métodos iterativos para resolver sistemas de equações lineares partem de uma estimativa inicial da solução e geram uma sequência de pontos que convergem para a solução do problema. Na prática, o processo é finalizado quando o ponto atual é suficientemente próximo da solução do problema. Em muitos casos, é possível obter uma boa aproximação da solução, com um custo computacional muito menor do que o necessário para obter a solução exata por meio de um método direto.

Primeiramente, apresentaremos o método de gradiente conjugado e o método de gradiente conjugado precondicionado, necessários para a resolução dos sistemas lineares.

5.1 Método do Gradiente Conjugado

O método do gradiente conjugado, introduzido por Hestenes-Stiefel(1952), é um dos processos iterativos mais populares para solução de sistemas de equações do tipo:

$$Ax = b, \tag{5.1}$$

onde a matriz de coeficientes do sistema $A \in R^{n\times n}$ é simétrica positiva definida, $x \in R^n$ é a incógnita do problema e $b \in R^n$ é o termo independente. Este método explora o fato de que a solução do sistema dado pela equação (5.1) equivale à

minimização da função quadrática:

$$f(x) = \frac{1}{2}x^t Ax - b^t x$$

Este método possui as melhores propriedades de convergência e requer o armazenamento somente de alguns poucos vetores, além dos dados originais do problema.

Algoritmo 5.1 *Algoritmo do Método do Gradiente Conjugado*

Dados: $A \in R^{n\times n}$ simétrica positiva definida, $b \in R^n$ e $x_0 \in R^n$.

Passo 1: Calcular

$$r_0 = b - Ax_0,$$

$$p_0 = r_0$$

Passo 2: Determinação

Para $i = 0, 1, ...,$ até convergir, fazer

$$\begin{aligned}
\alpha_i &= \frac{r_i^t r_i}{(Ap_i)^t p_i} \\
x_{i+1} &= x_i + \alpha_i p_i \\
r_{i+1} &= r_i - \alpha_i A p_i \\
\beta_i &= \frac{r_{i+1}^t r_{i+1}}{r_i^t r_i} \\
p_{i+1} &= r_{i+1} - \beta_i p_i
\end{aligned}$$

fim.

5.1.1 Método do Gradiente Conjugado Precondicionado

Na solução de sistemas com matrizes simétricas e definidas positivas, é utilizado o método do gradiente conjugado. Quando trabalhamos com matrizes mal condicionadas, é necessário o uso de precondicionadores para garantir a convergência do método (Saad [25]).

Algoritmo 5.2 *Algoritmo do Método do Gradiente Conjugado Precondicionado*

Dados: $A \in R^{n\times n}$ simétrica positiva definida, $b \in R^n$ e $x_0 \in R^n$.

Passo 1: Calcular

$$r_0 = b - Ax_0,$$

$$z_0 = N^{-1} r_0$$

$$p_0 = z_0$$

Passo 2: Determinação

Para $i = 0, 1, ...,$ até convergir, fazer

$$\begin{aligned}
\alpha_i &= \frac{r_i^t z_i}{(Ap_i)^t p_i} \\
x_{i+1} &= x_i + \alpha_i p_i \\
r_{i+1} &= r_i - \alpha_i A p_i \\
\beta_i &= \frac{r_{i+1}^t z_{i+1}}{r_i^t z_i} \\
p_{i+1} &= z_{i+1} - \beta_i p_i
\end{aligned}$$

fim.

A escolha de um bom precondicionador é muito importante. Golub [19] e Greenbaum [26] consideram o precondicionador do tipo decomposição incompleta de Cholesky um eficiente pré-condicionador para sistemas com matriz de coeficientes esparsa.

5.2 Sistema Linear Interno do FDIPA para Problemas Lineares com Desigualdades

Seja o sistema linear do problema FDIPA mostrado na equação (4.6), dado por:

$$\begin{bmatrix} B & A^t \\ \Lambda A & G(x) \end{bmatrix} \begin{bmatrix} d_0 \\ \lambda_0 \end{bmatrix} = - \begin{bmatrix} c \\ 0 \end{bmatrix}.$$

Este sistema é denominado primal-dual para achar d_0 e λ_0:

$$Bd_0 \quad + A^t\lambda_0 \quad = -c \tag{5.2}$$

$$\Lambda A d_0 \quad + G(x)\lambda_0 \quad = 0. \tag{5.3}$$

Da equação (5.2) tem-se:

$$d_0 = -B^{-1}(c + A^t\lambda_0). \tag{5.4}$$

Substituindo a equação (5.4) na equação (5.3), temos o sistema dual:

$$(-\Lambda A B^{-1} A^t + G(x))\lambda_0 = \Lambda A B^{-1} c. \tag{5.5}$$

Ao multiplicar a equação (5.5) pela matriz diagonal Λ^{-1}, obtemos:

$$(-AB^{-1}A^t + \Lambda^{-1}G(x))\lambda_0 = AB^{-1}c. \tag{5.6}$$

Garante-se que a matriz $(-AB^{-1}A^t + \Lambda^{-1}G(x))$ seja simétrica e definida positiva, permitindo usar o método do gradiente conjugado. O sistema original pode ser

um sistema bem condicionado; contudo, quando se faz a multiplicação pela matriz diagonal Λ^{-1}, este sistema, em geral, passa a ser mal condicionado, dado que os multiplicadores de Lagrange das restrições inativas tendem para zero. Portanto, é necessário o uso de precondicionadores.

Neste trabalho, utilizamos o precondicionador Cholesky incompleto, proposto em [27], para garantir a convergência do método.

5.3 Resolução do Sistema Dual

O algoritmo mostrado na seção 5.1.1 (algoritmo do método do gradiente conjugado precondicionado) é aplicado no sistema dual (5.6). Em cada iteração, este método gera uma direção de descida viável; portanto, não precisamos terminar as iterações até chegar à solução do sistema. Utilizaremos, ainda um critério para truncar as iterações, a fim de ter um melhor desenvolvimento do método e reduzir o custo computacional.

Este critério está baseado na comparação entre os passos (t) calculados na equação (5.3), na forma exata, e no sistema dual (5.6), em cada iteração do método do gradiente conjugado pré-condicionado pelo Cholesky incompleto.

Propomos o critério de parada para truncar o método, da seguinte forma:

Seja d_0^r e λ_0^r os valores achados na iteração r do gradiente conjugado pré-condicionado aplicado no sistema dual (5.6).

Supomos que é solução da equação (5.3)

$$\Lambda A d_0^r \ + \ G(x)\lambda_0^r \ = \ 0.$$

Para cada $i = 1, 2, ..., m$, temos que

$$\lambda_i a_i d_0^r \ + \ (Ax - b)_i \lambda_{0i}^r,$$

onde a_i é a linha i da matriz A, logo

$$a_i d_0^r = -(Ax - b)_i \frac{\lambda_{0i}^r}{\lambda_i}. \tag{5.7}$$

Agora, considerando as restrições de desigualdade, mediante uma aproximação destas restrições, na direção d_0^r, calculamos um passo máximo admissível $t1$:

$$(A(x + t1d_0^r) - b)_i \leq (Ax - b)_i \ + \ t1_i a_i d_0^r \leq 0$$

$$t1_i a_i d_0^r \leq -(Ax - b)_i. \tag{5.8}$$

Substituindo (5.7) em (5.8), obtemos:

$$t1_i \leq \frac{\lambda_i}{\lambda_{0i}^r}.$$

Então,

$$t1 = min(\frac{\lambda_i}{\lambda_{0i}^r}).$$

Por outro lado, calculamos o passo máximo admissível $t2$, que bate as restrições (5.8). Temos

$$(A(x + t2d_0^r) - b)_i \leq (Ax - b)_i \ + \ t2_i a_i d_0^r \leq 0.$$

Se $a_i d_0^r > 0$

$$t2_i \leq -\frac{(Ax - b)_i}{a_i d_0^r}.$$

Então,

$$t2 = min(-\frac{(Ax - b)_i}{a_i d_0^r}).$$

Por último, fazemos uma comparação dos passos. Se

$$t2 > \tau t1 \qquad \tau \in (0, 1],$$

o algoritmo pára; senão, faz-se uma nova iteração, até satisfazer o critério acima.

Capítulo 6

Resultados Numéricos

Neste cápitulo, apresentaremos os resultados da adaptação do algortimo FDIPA para programação linear implementado em MATLAB. Os experimentos computacionais foram realizados com problemas da biblioteca NETLIB [20].

Primeiro, apresentamos os resultados numéricos obtidos para problemas de programação linear com restrições de desigualdade dos seguintes problemas:

Problema 1.

$$
\begin{array}{lrrrrl}
min & -2x_1 & -x_2 & & & \\
s.a. & & & & & \\
 & -10x_1 & -3x_2 & +40 & \leq 0 & \\
 & x_1 & -2x_2 & & \leq 0 & \\
 & 8x_1 & +3x_2 & -70 & \leq 0 & \\
 & -x_1 & +10x_2 & -50 & \leq 0 & \\
 & & & x_i & \geq 0 & \text{i=1,2.}
\end{array}
$$

Problema 2.

$$
\begin{array}{lrrrrrl}
min & -20x_1 & -10x_2 & -x_3 & & & \\
s.a. & & & & & & \\
 & 3x_1 & -3x_2 & +5x_3 & -50 & \leq 0 & \\
 & x_1 & & +x_3 & -10 & \leq 0 & \\
 & x_1 & -1x_2 & +4x_3 & -20 & \leq 0 & \\
 & 2x_1 & +x_2 & +3x_3 & -30 & \leq 0 & \\
 & 2x_1 & +2x_2 & +x_3 & -66 & \leq 0 & \\
 & x_1 & +4x_2 & +3x_3 & -75 & \leq 0 & \\
 & 2x_1 & +2x_2 & +3x_3 & -80 & \leq 0 & \\
 & & & & x_i & \geq 0 & \text{i=1,2,3.}
\end{array}
$$

Consideremos os problemas Afiro e Blend da biblioteca NETLIB na forma padrão. Testamos uma versão modificada, Afiro* e Blend*, mudando o sinal das restrições de igualdade para desigualdade, gerando assim um novo problema.

Nas tabelas, mostramos o seguinte:

- $c^t x^k$: Valor da função objetivo.
- Iter FDIPA: Total de iterações do FDIPA.
- Iter GC: Total de iterações do método de gradiente conjugados precondicionado, dada pela soma do número de iterações realizadas para resolver o sistema linear que calcula a direção do FDIPA.
- Iter GC-T: Total de iterações do método de gradiente conjugados precondicionado truncado, dada pela soma do número de iterações realizadas para resolver o sistema linear que calcula a direção do FDIPA.
- Iter d_0: Total de iterações do FDIPA sem fazer a deflexão.
- Iter d: Total de iterações do FDIPA com a deflexão.

Tabela 6.1: Testes numéricos do FDIPA com gradiente conjugado precondicionado

	Linha	Coluna	$c^t x^k$	Iter FDIPA	Iter GC
Problema 1	4	2	-18.91528671	2	4
Problema 2	7	3	-299.99900155	4	8
Afiro*	27	51	-464.75202340	15	31
Blend*	74	114	-29.80016498	500	1245

Tabela 6.2: Testes numéricos do FDIPA com gradiente conjugado precondicionado truncado

	$c^t x^k$	Iter FDIPA	Iter GC-T
Problema 1	-18.91528670	2	2
Problema 2	-299.99837432	5	7
Afiro*	-464.75198398	15	15
Blend*	-30.57732381	25	286

Para comparar os resultados, usamos a rotina LINPROG do MATLAB, que foi desenvolvida baseando-se na referência Dantzig [3].

Tabela 6.3: Testes numéricos em MATLAB usando LINPROG

	$c^t x^k$	Restrições violadas
Problema 1	-18.91566265	0
Problema 2	-300	1
Afiro*	-464.75314263	0
Blend*	-30.82213945	1

Por último, apresentamos os resultados do algoritmo FDIPA para problemas de programação linear com restrições de igualdade e desigualdade.

Tabela 6.4: Descrição dos problemas originais

	Linha	Coluna	*no-zeros*
Afiro	27	32	88
Adlittle	56	97	465
Blend	74	83	521
Sc105	104	103	281
Sc50a	49	48	131
Share2b	96	79	730

Tabela 6.5: Testes numéricos do CPLEX e FDIPA para os problemas originais

	$c^t x^k$ CPLEX	Iter CPLEX	$c^t x^k$ FDIPA	Iter d_0	$c^t x^k$ FDIPA	Iter d
Afiro	-464.75314286	7	-464.75314279	14	-464.75313905	11
Blend	-30.81214984	49	-30.81214919	19	-30.81214964	19
Sc105	-52.20206114	32	-52.20206121	61	-52.20206121	23
Sc50a	-64.57507698	21	-64.57507706	12	-64.57507706	12
Share2b	-415.73224074	84	-415.73224074	39	-415.73223788	22

Tabela 6.6: Testes numéricos do FDIPA para os problemas da forma padrão

	Linha	Coluna	$c^t x^k$ d_0	Iter d_0	$c^t x^k$ d	Iter d
Afiro	27	32	-464.75314280	10	-464.75313963	7
Adittle	56	97	225494.9629	32	225488.1141	26
Blend	74	83	-30.80900624	24	-30.80900508	17

Também fizemos o teste dos problemas de NETLIB na forma primal, ou seja, acrescentando as variáveis de folga. Atualmente, os trabalhos resolvem os problemas na forma primal.

Capítulo 7

Conclusões

O FDIPA para programação linear, em comparação com os método de pontos interiores primal-dual, não precisa de uma função barreira que tem que minimizar: ele não tem parâmetros. Todas as variáveis usadas no algoritmo FDIPA e o tamanho de passo são calculados de forma numérica, usando as condições de otimalidade.

A grande vantagem do enfoque proposto é que não são necessárias variáveis de folga nem do problema dual. Trabalhamos com a formulação original do problema de Programação Linear, obtendo bons resultados.

Nos problemas de programação linear com restrições de desigualdade resolvidos com o FDIPA, propusemos um critério de truncamento nas iterações, no método de gradiente conjugado precondicionado.

Nos próximos trabalhos, podemos procurar um critério para truncar o método de gradiente para os problemas de programação linear com restrições de igualdade e desigualdade.

Referências Bibliográficas

[1] KLEE, V. "How good is the simplex method", *Inequalities*, pp. 159–175, 1972.

[2] DANTZIG, G. B. "Maximization of a linear function of variables subject to linear inequalities", *New York*, 1951.

[3] DANTZIG, G. *Linear programming and extensions*. Princeton university press, 2016.

[4] MACULAN, N., FAMPA, M. H. C. *Otimização Linear*. Brasília: EdUnB, 2006.

[5] KARMARKAR, N. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311. ACM, 1984.

[6] ANSTREICHER, K. M. "A combined phase I-phase II projective algorithm for linear programming", *Mathematical Programming*, v. 43, n. 1-3, pp. 209–223, 1989.

[7] ANSTREICHER, K. M. "A monotonic projective algorithm for fractional linear programming", *Algorithmica*, v. 1, n. 1-4, pp. 483–498, 1986.

[8] LUSTIG, I. J. *A Practical Approach to Karmarkar's Algorithm.* Relatório técnico, STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB, 1985.

[9] GONZAGA, C. C. "Conical projection algorithms for linear programming", *Mathematical Programming*, v. 43, n. 1-3, pp. 151–173, 1989.

[10] DIKIN, I. "Iterative solution of problems of linear and quadratic programming". In: *Soviet Mathematics Doklady*, v. 8, pp. 674–675, 1967.

[11] BARNES, E. R. "A variation on Karmarkar's algorithm for solving linear programming problems", *Mathematical programming*, v. 36, n. 2, pp. 174–182, 1986.

[12] CAVALIER, T., SOYSTER, A. *Some computational experience and a modification of the Karmarkar algorithm.* Pennsylvania State Univ., College of

Engineering, Department of Industrial and Management Systems Engineering, 1985.

[13] GONZAGA, C. C. "An algorithm for solving linear programming problems in O (n 3 L) operations". In: *Progress in mathematical programming*, Springer, pp. 1–28, 1989.

[14] GONZAGA, C. C. "Path-following methods for linear programming", *SIAM review*, v. 34, n. 2, pp. 167–224, 1992.

[15] MONTEIRO, R. D., ADLER, I. "An extension of Karmarkar type algorithm to a class of convex separable programming problems with global linear rate of convergence", *Mathematics of Operations Research*, v. 15, n. 3, pp. 408–422, 1990.

[16] MONTEIRO, R. D., ADLER, I. "Interior path following primal-dual algorithms. Part II: Convex quadratic programming", *Mathematical Programming*, v. 44, n. 1-3, pp. 43–66, 1989.

[17] WRIGHT, S. J. *Primal-Dual Interior-Point Methods*, v. 54. SIAM, 1997.

[18] HERSKOVITS, J. "Feasible direction interior-point technique for nonlinear optimization", *Journal of optimization theory and applications*, v. 99, n. 1, pp. 121–146, 1998.

[19] GOLUB, G. H., VAN LOAN, C. F. *Matrix computations*, v. 3. JHU Press, 2012.

[20] GAY, D. M. "Electronic mail distribution of linear programming test problems", *Mathematical Programming Society COAL Newsletter*, v. 13, pp. 10–12, 1985.

[21] LUENBERGER, D. G., YE, Y., OTHERS. *Linear and nonlinear programming*, v. 2. Springer, 1984.

[22] ZANGWILL, W. I. "Non-linear programming via penalty functions", *Management science*, v. 13, n. 5, pp. 344–358, 1967.

[23] MAYNE, D. Q., POLAK, E. "Feasible directions algorithms for optimization problems with equality and inequality constraints", *Mathematical Programming*, v. 11, n. 1, pp. 67–80, 1976.

[24] LUENBERGER, D., YE, Y. *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science. Springer US, 2008.

[25] SAAD, Y. *Iterative methods for sparse linear systems*, v. 82. siam, 2003.

[26] GREENBAUM, A. *Iterative methods for solving linear systems*, v. 17. Siam, 1997.

[27] LIN, C.-J., MORÉ, J. J. "Incomplete Cholesky factorizations with limited memory", *SIAM Journal on Scientific Computing*, v. 21, n. 1, pp. 24–45, 1999.