ERRATA nos 2 exemplares da dissertação do aluno Alessandro Fonseca Esteves Coelho
Em folha ii:

Onde se lê:	
Área de concentração: Pesquisa Operacional	Desconsidere
Programa de Pós-Graduação: **Mestrado em Matemática Aplicada**	Leia-se: Programa de Pós-Graduação – **Matemática Aplicada**

Prof. Dr. Luiz Koodi Hotta
Coordenador CPG/IMECC
Matric. 042471 - UNICAMP

Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
Departamento de Matemática Aplicada

Solução de Sistemas Lineares de Grande Porte Usando Variantes do Método dos Gradientes Conjugados

Alessandro Fonseca Esteves Coelho*
Mestrado em Matemática Aplicada - Campinas - SP

Orientador: Prof. Dr. Aurelio Ribeiro Leite de Oliveira
Coorientadora: Profa. Dra. Marta Ines Velazco Fontova

∗ Este trabalho teve apoio financeiro da CAPES.

Solução de Sistemas Lineares de Grande Porte Usando Variantes do Método dos Gradientes Conjugados

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por **Alessandro Fonseca Esteves Coelho** e aprovada pela comissão julgadora.

Campinas, 31 de maio de 2011.

Prof. Dr. Aurelio Ribeiro Leite de Oliveira
Orientador

Prof^{a}. Dr^{a}. Marta Ines Velazco Fontova
Coorientadora

Banca Examinadora:
Prof. Dr. Aurelio Ribeiro Leite de Oliveira (IMECC - UNICAMP)
Prof^{a}. Dr^{a}. Márcia Aparecida Gomes Ruggiero (IMECC - UNICAMP)
Prof^{a}. Dr^{a}. Rubia Mara de Oliveira Santos (UFMS)

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do título de MESTRE em Matemática Aplicada.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Coelho, Alessandro Fonseca Esteves
C65s Solução de sistemas lineares de grande porte usando variantes do método dos gradientes conjugados/Alessandro Fonseca Esteves Coelho-- Campinas, [S.P. : s.n.], 2011.

Orientador : Aurelio Ribeiro Leite de Oliveira; Marta Ines Velazco Fontova.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1.Programão linear. 2.Métodos de pontos interiores. 3.Métodos do gradiente conjugado. I. Oliveira, Aurelio Ribeiro Leite de, 1962-. II.Velazco Fontova, Marta Ines. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Título em inglês: Large scale linear systems solution using variants of the conjugate gradient method

Palavras-chave em inglês (Keywords): 1.Linear programming. 2.Interior point methods. 3.Conjugate gradient methods.

Área de concentração: Pesquisa operacional

Titulação: Mestre em Matemática Aplicada

Banca examinadora: Prof. Dr. Aurelio Ribeiro Leite de Oliveira (IMECC – UNICAMP)
Profa. Dra. Márcia Aparecida Gomes Ruggiero (IMECC – UNICAMP)
Profa. Dra. Rubia Mara de Oliveira Santos (UFMS)

Data da defesa: 31/05/2011

Programa de Pós-Graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 31 de maio de 2011 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof.(a). Dr(a). AURELIO RIBEIRO LEITE DE OLIVEIRA

Prof.(a). Dr(a). MÁRCIA APARECIDA GOMES RUGGIERO

Prof.(a). Dr(a). RUBIA MARA DE OLIVEIRA SANTOS

Aos meus pais, Vanda Fonseca Coelho e Yvan Esteves Coelho.

Agradecimentos

Primeiramente, gostaria de agradecer a Deus por todas as oportunidades e aos meus pais que sempre me apoiaram.

Ao Prof. Aurelio pela sua orientação e disponibilidade em todos os momentos.

À Prof Marta pelas importantes contribuições na coorientação e pela ajuda com o código PCx.

À Prof Cristiane pelas horas de estudo no início do mestrado .

Aos colegas de pós-graduação: Luciana , Salvador , Rafael, Rodrigo, Jair, Cecília, Leandro, Douglas, Thadeu e tantos outros que, em algum momento, apoiaram e ajudaram.

Aos verdadeiros amigos, próximos ou distantes, novos ou antigos.

A todos os professores presentes em minha formação.

Aos funcionários da secretaria de pós-graduação.

À CAPES, pelo suporte financeiro.

Resumo

Um método frequentemente utilizado para a solução de problemas de programação linear é o método de pontos interiores. Nestes métodos precisamos resolver sistemas lineares para calcular a direção de Newton a cada iteração. A solução desses sistemas consiste no passo de maior esforço computacional nos métodos de pontos interiores. A fatoração de Cholesky é a opção mais utilizada para resolver estes sistemas. Contudo, quando trabalhamos com problemas de grande porte, esta fatoração pode ser densa e torna-se inviável trabalhar com esses métodos. Nestes casos, uma boa opção consiste no uso de métodos iterativos precondicionados. Estudos anteriores utilizam o método dos gradientes conjugados precondicionado para obter uma solução destes sistemas. Particularmente, os sistemas originados dos métodos de pontos interiores, são, naturalmente, sistemas de equações normais. Porém, a versão padrão do método dos gradientes conjugados, não considera a estrutura de equações normais do sistema. Neste trabalho propomos a utilização de duas versões do método de gradientes conjugados precondicionado que consideram a estrutura de equações normais destes sistemas. Estas versões serão comparadas com a versão de gradientes conjugados precondicionada que não considera a estrutura de equações normais do sistema. Resultados numéricos com problemas de grande porte mostram que uma dessas versões é competitiva em relação à versão padrão.

Abstract

An often used method for solving linear programming problems is the interior point method. In these methods we need to solve linear systems to compute the Newton search direction at each iteration. The solution of these systems is the procedure of most computational effort in interior point methods. The Cholesky factorization is the most often used method to solve these systems. However, when dealing with large scale problems, this factorization can be dense and it become impossible to apply such methods. In such cases, a good option is the use of preconditioned iterative methods. Previous studies have used the preconditioned conjugate gradient method to find the solution of these systems. Particularly, the systems arising from interior point methods are, naturally, systems of normal equations type. Nevertheless, the standard version of the conjugate gradient method, does not take into account the normal equations system structure. This study proposes the use of two versions of preconditioned conjugate gradient method considering the normal equations structure of these systems. These versions are compared with the preconditioned conjugate gradient version that does not consider that structure. Numerical results with large scale problems show that one of these versions is competitive with the standard one.

Sumário

1 Introdução **1**

2 Métodos de Pontos Interiores **3**
2.1 Problema de Programação Linear 3
2.2 Método Primal-Dual Afim Escala 6
2.2.1 Método de Newton 6
2.2.2 Método Primal-Dual Afim Escala 8
2.3 Método Primal-Dual Seguidor de Caminhos 10
2.4 Método Preditor-Corretor 11
2.5 O Sistema de Equações Normais 14
2.6 Considerações Finais 15

3 Métodos Iterativos para Solução de Sistemas Lineares **16**
3.1 Métodos de Projeção 17
3.2 Método dos Gradientes Conjugados 18
3.3 Método dos Gradientes Conjugados e Equações Normais 20
3.3.1 Versão Normal Residual -NR 21
3.3.2 Versão Normal Error - NE 22
3.4 Precondicionadores 23
3.5 Método dos Gradientes Conjugados Precondicionado 24
3.5.1 Gradientes Conjugados Precondicionado para Equações Normais . . . 25
3.6 Considerações Finais 27

4 Precondicionadores para o Sistema de Equações Normais **28**
4.1 Precondicionador Híbrido 29
4.1.1 Fatoração Controlada de Cholesky 29
4.1.2 Precondicionador Separador 30
4.1.3 Mudança de Fase 30
4.2 Métodos de Pontos Interiores e Métodos Iterativos 31
4.3 Considerações Finais 33

5 Experimentos Numéricos **34**
5.1 Modificações no PCx 34
5.2 Problemas Teste 35
5.3 Resultados Computacionais 35
5.3.1 Perfil de Desempenho 44
5.4 Considerações Finais 47

6 Conclusões e Trabalhos Futuros **49**

Lista de Tabelas

5.1 Dados dos Problemas Teste - Parte 1 36
5.2 Dados dos Problemas Teste Parte - 2 37
5.3 Parâmetro η para versão PGCNR 38
5.4 Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NR (PGCNR) - Tempo 40
5.5 Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NE (PGCNE) - Tempo 42
5.6 Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NE x NR - Tempo 43

Lista de Figuras

5.1 Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Iterações de pontos interiores 46
5.2 Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Iterações de gradientes conjugados 46
5.3 Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Tempo de resolução 47

Notações Utilizadas

Letras maiúsculas, $A, B, \ldots$, representam matrizes reais. O elemento da matriz A, na linha i e coluna j, será a_{ij}.

Letras minúsculas, $a, b, \ldots, x, y, z$, representam vetores coluna, isto é, $a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$. O elemento do vetor a, na linha k, será a_k. O vetor linha será a transposição do vetor coluna, ou seja, $a^t = \begin{pmatrix} a_1 & a_2 & \ldots & a_n \end{pmatrix}$. Indicaremos a operação de transposição de vetores e matrizes por $(\cdot)^t$.

Letras gregas, $\alpha, \beta, \ldots$, representam escalares.

Uma função de uma variável será denotada por $F(x)$ e a função de várias variáveis por $F(x, y, z)$, por exemplo.

$\Re$: representa o conjunto dos números reais.

e: representa um vetor com todas as componentes iguais a 1.

Quando uma matriz A for quadrada não-singular, sua inversa será denotada por A^{-1}.

$\nabla F(x)$: gradiente da função F em x.

$\langle x, y \rangle = x^t y = \sum_{i=1}^{n} x_i y_i$: Produto escalar entre os vetores x e y.

$\|x\| = \|x\|_2 = \langle x, x \rangle^{\frac{1}{2}}$: Norma Euclidiana do vetor x.

Capítulo 1

Introdução

Desde o trabalho de Karmarkar [19], os métodos de pontos interiores tem atraído grande interesse de diversos pesquisadores. Estes métodos, quando cuidadosamente implementados, tem se mostrado eficientes. Isto ocorre, principalmente, quando aplicados a problemas de grande porte e, neste caso, os métodos de pontos interiores podem obter uma solução mais rápido que o método simplex [23].

No método de pontos interiores mais utilizado, o preditor-corretor, resolvemos dois sistemas de equações lineares a cada iteração. Realizando operações algébricas nestes sistemas obtemos duas formulações distintas. A primeira, chamada de sistema aumentado, transforma a matriz dos coeficientes do sistema em uma matriz simétrica, indefinida e geralmente esparsa. A segunda formulação obtém um sistema de equações normais que é simétrico e definido positivo.

A solução destes dois sistemas é o passo mais caro e que demanda maior tempo e esforço computacional nos métodos de pontos interiores. Portanto, é extremamente importante uma boa escolha do método de resolução que será utilizado nessa etapa. Tipicamente, métodos diretos são utilizados para obter a solução destes sistemas, por exemplo, a fatoração de Cholesky das matrizes simétricas definidas positivas [6, 17, 22]. Porém limitações de tempo e memória podem impedir o seu uso. Assim, abordagens iterativas são mais eficientes quando bons precondicionadores são utilizados pois a matriz dos coeficientes se torna muito mal-condicionada nas iterações finais.

Uma abordagem híbrida para a solução destes sistemas foi proposta por Bocanegra, Campos e Oliveira [9]. Neste abordagem, dois precondicionadores são utilizados. O primeiro, nas iterações iniciais, é um tipo de fatoração incompleta, denominada Fatoração Controlada de Cholesky, proposta por Campos [11]. O segundo, denominado precondicionador separador, foi proposto por Oliveira [27]. A troca de precondicionadores acontece quando a Fatoração Controlada de Cholesky perde a eficiência e torna a convergência do método iterativo mais lenta.

No trabalho [9] o método iterativo escolhido foi o método de gradientes conjugados precondicionado. Neste método, resolvemos sistemas lineares do tipo $Ax = b$ necessitando apenas que a matriz dos coeficientes seja simétrica e definida positiva, caso dos sistemas ori-

ginados do método de pontos interiores, onde é possível obter a matriz nestas condições. Os sistemas originados do método de pontos interiores são sistemas de equações normais. Essa característica não é explorada quando o método de gradientes conjugados precondicionado é utilizado. Entretanto, existem versões deste mesmo método que consideram essa estrutura em sua formulação [32].

O objetivo deste trabalho é avaliar a eficiência de duas versões de gradientes conjugados precondicionado. Estas versões consideram a estrutura de equações normais dos sistemas simétricos e definidos positivos que surgem do método de pontos interiores. Assim, trabalharemos com o método preditor-corretor utilizando diferentes versões de gradientes conjugados, precondicinadas pela abordagem híbrida, para obter as soluções dos sistemas lineares. A versão clássica do método dos gradientes conjugados precondicionada já foi testada, em trabalhos anteriores, com apenas um precondicionador [28] substituindo a fatoração de Cholesky para a resolução dos sistemas lineares. O desempenho desta mesma versão de gradientes conjugados tambem já foi avaliado com a abordagem híbrida [9], isto é, utilizando dois precondicionadores. Nesta dissertação, propomos os testes com as versões de gradientes conjugados precondicionado que considerem a estrutura de equações normais dos sistemas envolvidos. Assim, poderemos avaliar se estas versões contribuirão com maior ou menor eficiência na resolução de problemas de grande porte.

Este trabalho está dividido em seis capítulos. No Capítulo 2 apresentaremos o problema de programação linear e uma descrição dos métodos de pontos interiores. O método preditor-corretor será apresentado ao final deste capítulo e utilizaremos este método de pontos interiores ao longo deste trabalho. No Capítulo 3, apresentaremos os métodos iterativos estudados no trabalho. Assim, diferentes versões do método de gradientes conjugados serão apresentadas: a versão clássica, precondicionada, para equações normais e, finalmente, as versões para equações normais precondicionadas. Os precondicionadores adotados serão descritos no Capítulo 4 e, também, a heurística desenvolvida para a troca de precondicionadores na abordagem híbrida. Ao final deste capítulo detalhes de implementação das versões de gradientes conjugados serão apresentados. No Capítulo 5, descrevemos as modificações realizadas no código PCx [12] e apresentaremos os experimentos númericos realizados. As conclusões e os trabalhos que poderão ser futuramente realizados encontram-se no Capítulo 6. Finalizando o trabalho, as Referências bibliográficas.

Capítulo 2

Métodos de Pontos Interiores

No ano de 1984, Karmarkar [19] propôs o primeiro método de pontos interiores com complexidade polinomial no pior caso. Isto provocou uma grande revolução na programação linear, pois Karmarkar afirmava que o algoritmo proposto tinha eficiência, na prática, muito melhor que o método utilizado naquela época para problemas de grande porte, o método simplex desenvolvido por Dantzig [13] em 1948.

Apesar de Karmarkar não ter apresentado resultados computacionais, os bons resultados obtidos em [6] comprovaram e impulsionaram as pesquisas sobre o método. A partir de então começaram a surgir diversas variantes destes métodos. Em 1989 surgiu um método primal-dual baseado no método de Karmarkar [20]. No ano seguinte, Mehrotra [25] apresentou o método de pontos interiores preditor-corretor e este passou a ser base para diversas implementações desses métodos.

A maioria dos métodos de pontos interiores do tipo primal-dual, encontrados na literatura, pode ser vista como variações do método de Newton aplicado às condições de otimalidade.

2.1 Problema de Programação Linear

Um problema de programação linear é definido como a otimização de uma função linear, chamada função objetivo, sujeita a um conjunto de restrições lineares de igualdade e/ou desigualdade [21]. Este conjunto de restrições determina uma região, denominada região factível. A forma padrão de representar um problema de programação linear é:

$$\begin{array}{ll} \text{minimizar/maximizar} & c^t x \\ \text{sujeito a} & Ax = b,\ x \geq 0, \end{array}$$

onde A é uma matriz $m \times n$ de posto completo e c, b e x são vetores coluna de dimensões apropriadas. O vetor x corresponde às variáveis do problema que otimizam a função objetivo. As equações do sistema $Ax = b$ podem ser multiplicadas por -1 para que $b_i \geq 0, \forall i = 1, \ldots, m$.

Observe que um problema de programação linear (PPL), escrito na forma padrão, não apresenta desigualdades no conjunto de restrições $Ax = b$. Todo PPL pode ser escrito sob esta forma. Geralmente não encontramos os problemas de programação linear na forma padrão. Eles podem ocorrer com variáveis não-negativas, livres ou canalizadas e, ainda, as restrições podem ser na forma de equações ou inequações.

Assim, quando um PPL apresenta uma ou mais equações com desigualdades, estas desigualdades podem ser eliminadas com a inclusão de variávies de folga ou de excesso. Suponha que tenhamos uma desigualdade na i-ésima equação do conjunto de restrições, isto é:

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i.$$

Podemos acrescentar uma variável de folga ao problema original com o objetivo de transformar a desigualdade apresentada em uma igualdade, ou seja:

$$a_{i1}x_1 + \ldots + a_{in}x_n + x_{n+1} = b_i,$$

onde $x_{n+1} \geq 0$. Suponha agora que tenhamos outra desigualdade, no conjunto de restrições, na k-ésima linha, da seguinte forma:

$$a_{k1}x_1 + \ldots + a_{kn}x_n \geq b_k.$$

De forma semelhante ao que fizemos anteriormente, podemos reescrever esta última equação:

$$a_{k1}x_1 + \ldots + a_{kn}x_n - x_{n+2} = b_k,$$

onde $x_{n+2} \geq 0$.

A inclusão de variáveis, para a eliminação das desigualdades, não altera o valor ótimo da função objetivo pois os coeficientes destas variávies na função objetivo são nulos, isto é, $c_k = 0$ para cada variável de folga ou excesso criada. Com estas modificações qualquer PPL, com desigualdades no conjunto de restrições, pode ser escrito na forma padrão.

Existem ainda PPL onde o vetor x é um vetor que contém variáveis livres, isto é, algumas componentes deste vetor podem assumir qualquer valor real. Neste caso, cada uma dessas componentes x_i pode ser escrita como a diferença de dois valores positivos, isto é, $x_i = u_i - v_i$, onde $u_i \geq 0$ e $v_i \geq 0$. Note que quando escrevemos $x_i = u_i - v_i$ o número de variávies do problema aumenta, ou seja, para cada x_k escrito como $u_k - v_k$ teremos o acréscimo de uma variável. E assim o problema com variáveis livres pode ser escrito na forma padrão.

Outro tipo de restrições muito comum nos PPL é a presença de variávies canalizadas, vejamos:

$$\begin{array}{ll} \text{minimizar} & c^t x \\ \text{sujeito a} & Ax = b,\ l \leq x \leq u, \end{array}$$

onde l é o vetor cujas componentes l_i são os limites inferiores para cada variável x_i e u é o vetor cujas componentes u_i são os limites superiores para cada variável x_i.

Neste caso, desejamos que cada valor x_i do vetor x que otimiza a função objetivo esteja compreendido entre u_i e l_i. Se desejamos escrever o problema com variáveis canalizadas na forma padrão devemos fazer uma substituição de variáveis e em seguida eliminar as desigualdades no conjunto de restrições. Vejamos:

$$l \leq x \leq u \Leftrightarrow 0 \leq x - l \leq u - l,$$

chamando $\tilde{x} = x - l$ e $\tilde{u} = u - l$ temos:

$$Ax = A(\tilde{x} + l) = A\tilde{x} + Al = b \Leftrightarrow A\tilde{x} = b - Al = \tilde{b} \text{ e}$$

$$c^t x = c^t(\tilde{x} + l) = c^t\tilde{x} + c^t l$$

onde $c^t l$ é uma constante e não interfere na otimização desta função, ou seja, pode ser desconsiderada. Assim:

$$\begin{array}{ll} \text{minimizar} & c^t\tilde{x} \\ \text{sujeito a} & A\tilde{x} = \tilde{b},\ 0 \leq \tilde{x} \leq \tilde{u}. \end{array}$$

Finalmente, incluimos as variáveis de folga:

$$\begin{array}{ll} \text{minimizar} & c^t\tilde{x} \\ \text{sujeito a} & A\tilde{x} = \tilde{b},\ \tilde{x} + s = \tilde{u},\ s \geq 0,\ \tilde{x} \geq 0. \end{array}$$

Com estas modificações escrevemos o problema canalizado na forma padrão.

Portanto, dado um problema de programação linear sempre podemos colocá-lo na forma padrão:

$$\begin{array}{ll} \text{minimizar} & c^t x \\ \text{sujeito a} & Ax = b,\ x \geq 0. \end{array} \tag{2.1}$$

Esta é a forma primal de um PPL, podemos associar, ao problema primal, o problema dual:

$$\begin{array}{ll} \text{maximizar} & b^t y \\ \text{sujeito a} & A^t y + z = c,\ z \geq 0, \end{array} \tag{2.2}$$

onde y é um vetor de dimensão $m \times 1$ de variáveis livres e z é um vetor $n \times 1$ de variáveis de folga duais.

Consideraremos, a partir daqui, que um PPL está na forma padrão. Assim, quando buscamos encontrar a solução de um PPL, desejamos encontrar os vetores x, y e z que atendam as condições dos problemas primal (2.1) e dual (2.2). Se os vetores x, y e z são a solução do PPL, teremos $Ax - b = 0$, $A^t y + z - c = 0$ e $x_i z_i = 0$, $\forall i = 1, \ldots, n$. Neste caso, dizemos que as condições de otimalidade foram atendidas. Portanto, é necessário que estas condições sejam atendidas para encontrarmos a solução do PPL. Com isso, as condições de otimalidade para os problemas primal e dual podem ser escritas em um sistema não-linear de equações com variáveis não-negativas:

$$F(x,y,z) = \begin{pmatrix} Ax - b \\ A^t y + z - c \\ XZe \end{pmatrix} = 0, (x,z) \geq 0, \tag{2.3}$$

onde X representa a matriz diagonal com as entradas do vetor x, Z representa a matriz diagonal com as entradas do vetor z e e é o vetor com componentes iguais a 1 de dimensão apropriada.

Para obtermos a solução do problema primal-dual basta resolvermos o sistema de equações não-lineares (2.3). Se (x, y, z) for uma solução do problema determinado pelas condições de otimalidade, então x é solução ótima do problema primal (2.1) e (y, z) é solução ótima do problema dual (2.2). Dizemos que um ponto (x, y, z) é factível se ele satisfizer o conjunto de restrições do problema primal e dual. O ponto é dito interior se $(x, z) > 0$, ou seja, quando todas as variáveis se encontram estritamente dentro dos seus limites.

2.2 Método Primal-Dual Afim Escala

Na Seção 2.1 apresentamos diferentes formas que podemos encontrar um problema de programação linear e como transformá-los para a forma padrão. A partir desta seção apresentaremos alguns métodos para obter a solução de um PPL, no caso, métodos de pontos interiores. Os métodos de pontos interiores geram uma sequência de pontos interiores que cada vez mais se aproximam de uma solução ótima até obter a convergência desejada. Para construir o método primal-dual afim escala [33] calculamos direções de busca aplicando o método de Newton nas condições de otimalidade (2.3).

2.2.1 Método de Newton

Como trabalharemos com métodos de pontos interiores que utilizam o método de Newton para a sua construção, apresentaremos aqui, brevemente, o método de Newton [33, 14].

Inicialmente, apresentaremos o método de Newton para funções de uma variável. O objetivo do método é estimar os zeros de uma função diferenciável $F(x)$. Geometricamente, dado uma abscissa inicial x_0, traçamos a tangente do gráfico de $F(x)$ para o ponto inicial $(x_0, F(x_0))$. Esta reta tangente interceptará o eixo das abscissas em x_1. Assim, traçamos novamente a reta tangente ao gráfico de $F(x)$, utilizando agora o ponto $(x_1, F(x_1))$. Encontraremos então, a nova estimativa para o zero da função, x_2. Repetiremos esse processo até encontrar o zero da função. Portanto temos um processo iterativo para obter o zero de uma função $F(x)$. Este processo pode ser descrito da seguinte forma:

$$x_{j+1} = x_j - \frac{F(x_j)}{F'(x_j)}, \tag{2.4}$$

onde j representa a j-ésima iteração do método e $F'(x_j)$ a derivada da função $F(x)$ no ponto x_j.

O método descrito por (2.4) pode ser deduzido através da expansão da função F em série de Taylor em torno do ponto x_0, isto é:

$$F(x) = F(x_0) + (x - x_0)F^{(1)}(x_0) + \frac{(x - x_0)^2}{2!}F^{(2)}(x_0) + \ldots + \frac{(x - x_0)^n}{n!}F^{(n)}(x_0) + \ldots \ .$$

Utilizando apenas os dois primeiros termos da série, temos:

$$F(x) \approx F(x_0) + (x - x_0)F^{(1)}(x_0). \tag{2.5}$$

Observe que a equação (2.5) representa uma reta que passa pelo ponto $F(x_0)$ e tem inclinação $F^{(1)}(x_0)$, ou seja, é uma reta tangente ao gráfico no ponto x_0.

O ponto que esta reta tangente intercepta o eixo das abscissas estará próximo do ponto que anula a função F, considerando que a função F seja bem aproximada por uma reta. Logo, temos o ponto x que anula F:

$$\begin{aligned} F(x_0) + (x - x_0)F^{(1)}(x_0) &= 0, \\ (x - x_0)F^{(1)}(x_0) &= -F(x_0), \\ x &= x_0 - \frac{F(x_0)}{F^{(1)}(x_0)}. \end{aligned}$$

Podemos melhorar a aproximação substituindo o ponto x_0 pelo ponto x e repetindo o processo. Desta forma, obtemos o método descrito pela equação (2.4), ou seja, o método de Newton para uma variável.

Agora, vamos generalizar o método de Newton para uma função $F(x)$ de várias variáveis. Assim, a função $F(x)$ passa a ser uma função vetorial, vejamos:

$$F(x) = \begin{pmatrix} F_1(x) \\ F_2(x) \\ \vdots \\ F_m(x) \end{pmatrix},$$

onde $F_i(x)$ é diferenciável $\forall i = 1, \ldots, m$.

Obter x que anule a função $F(x)$ é o mesmo que obter x que anule simultaneamente cada uma das funções $F_i(x)$. Usaremos a fórmula de Taylor, em torno de um ponto x_0, para cada uma das funções $F_i(x)$, como fizemos para uma função de uma variável, ou seja:

$$F_i(x) \approx F_i(x_0) + \nabla F(x)^t(x - x_0). \tag{2.6}$$

Escrevendo a equação (2.6) para cada função $F_i(x)$, com $i = 1, \ldots, m$, e fazendo $F_i(x) = 0$ obtemos o seguinte sistema de equações:

$$\begin{cases} \nabla F_1(x)^t(x - x_0) = -F_1(x_0) \\ \nabla F_2(x)^t(x - x_0) = -F_2(x_0) \\ \qquad \vdots \\ \nabla F_m(x)^t(x - x_0) = -F_m(x_0). \end{cases}$$

Agora, podemos reescrever esse sistema utilizando a matriz Jacobiana J da função F no ponto x_0, assim:

$$J(x_0)(x - x_0) = -F(x_0),$$

e, analogamente ao que fizemos para funções de uma variável, obtemos o método de Newton para funções de várias variáveis:

$$x_{j+1} = x_j - J(x_j)^{-1}F(x_j).$$

As equações deste sistema são conhecidas como equações de Newton e, fazendo $dx_j = -J(x_j)^{-1}F(x_j)$, obtemos as direções de Newton dx_j. Desta forma, podemos escrever o novo ponto x_j em função de dx_j:

$$x_{j+1} = x_j + dx_j.$$

2.2.2 Método Primal-Dual Afim Escala

Podemos, agora, voltar à construção do método primal-dual afim escala. Nos métodos primais-duais [36] partimos de um ponto interior (x_0, y_0, z_0), não obrigatoriamente factível, e resolvenos simultaneamente o problema primal e o problema dual. Definindo o resíduo primal por $r_p = b - Ax$, o resíduo dual por $r_d = c - A^t y - z$, resíduo afim escala por $r_a = -XZe$ e observando o sistema (2.3), temos:

$$-F(x, y, z) = \begin{pmatrix} r_p \\ r_d \\ r_a \end{pmatrix} = \begin{pmatrix} b - Ax \\ c - A^t y - z \\ -XZe \end{pmatrix}.$$

Assim, queremos que os resíduos (r_p, r_d, r_a) sejam nulos para obter uma solução ótima.

Portanto, se aplicarmos o método de Newton a (2.3) temos que dado um ponto inicial (x_0, y_0, z_0) o próximo ponto será $(x_1, y_1, z_1) = (x_0, y_0, z_0) - J(x_0, y_0, z_0)^{-1}F(x_0, y_0, z_0)$ onde,

$$J = \begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \text{ e } F(x_0, y_0, z_0) = \begin{pmatrix} Ax_0 - b \\ A^t y + z_0 - c \\ X_0 Z_0 e \end{pmatrix} = -r(x_0, y_0, z_0).$$

Desta forma podemos definir a direção de descida d como $d = J^{-1}r$, obtendo $(x_1, y_1, z_1) = (x_0, y_0, z_0) + \alpha(dx, dy, dz)$, onde α é o tamanho do passo. Calculamos α de tal forma que cada novo ponto seja interior a partir do ponto interior anterior. Definimos um tamanho de passo primal (α_p) e um tamanho de passo dual (α_d) que calculamos da seguinte forma:

$$\alpha_p = \tau min \left\{ -\frac{x_i}{dx_i} \right\}, \forall dx_i < 0,$$

$$\alpha_d = \tau min \left\{ -\frac{z_i}{dz_i} \right\}, \forall dz_i < 0,$$

onde, na prática, $\tau = 0.99995$ e os tamanhos de passo α_p e α_d têm valor máximo igual a 1 (passo de Newton) pois queremos que novo ponto continue sendo um ponto interior. Caso o tamanho do passo fosse sempre igual a 1, poderíamos ter um novo ponto fora da região interior do problema, o que não queremos. Por essa razão, o valor máximo do passo é igual a 1.

Quando fazemos o produto do Jacobiano J pela direção $d = (dx, dy, dz)$ obtemos os resíduos (r_p, r_d, r_a), isto é, $Jd = r$.

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^t & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} r_p \\ r_d \\ r_a \end{pmatrix}. \tag{2.7}$$

Neste momento, precisamos resolver esse sistema para obter as direções de busca. Assim, temos duas opções: eliminar a direção dz e construir um sistema chamado de sistema aumentado ou, a segunda opção, eliminar também dx e obter um conjunto de equações onde a direção dy é dada pela solução de um sistema linear. Neste caso, a matriz dos coeficientes do sistema obtido, pode ser escrita como o produto de uma matriz por sua transposta. Todo sistema que possui essa característica é chamado de sistema de equações normais. Portanto, quando utilizamos a segunda opção apresentada para resolver o sistema (2.7), obtemos um sistema de equações normais [33] para determinar a direção dy.

Primeiramente vamos eliminar dz no sistema (2.7). A partir da terceira equação do sistema (2.7) podemos escrever $dz = X^{-1}(r_a - Zdx)$ e substituir na segunda equação desse mesmo sistema, obtendo o sistema aumentado:

$$\begin{pmatrix} -X^{-1}Z & A^t \\ A & 0 \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} r_d - X^{-1}r_a \\ r_p \end{pmatrix}.$$

Fazendo $D = X^{-1}Z$ e eliminando agora dx, obtemos a segunda opção, o sistema de equações normais. Para isto, basta isolarmos dx na segunda equação do sistema aumentado e substituir na primeira equação do mesmo sistema. Com isso, obtemos o seguinte sistema de equações:

$$\begin{aligned} dy &= (AD^{-1}A^t)^{-1}(r_p + A(D^{-1}r_d - Z^{-1}r_a)) \\ dx &= D^{-1}(A^t dy - r_d + X^{-1}r_a) \\ dz &= X^{-1}(r_a - Zdx). \end{aligned}$$

Observe que a direção dy é determinada quando resolvemos o sistema

$$(AD^{-1}A^t)dy = r_p + A(D^{-1}r_d - Z^{-1}r_a),$$

onde a $AD^{-1}A^t$ é a matriz dos coeficientes. Esta matriz pode ser escrita como o produto de uma matriz pela sua transposta, isto é, como GG^t ou ainda G^tG. Por exemplo, se tomarmos $G = AD^{-\frac{1}{2}}$ teremos:

$$GG^t = (AD^{-\frac{1}{2}})(AD^{-\frac{1}{2}})^t = (AD^{-\frac{1}{2}})(D^{-\frac{1}{2}}A^t) = AD^{-1}A^t.$$

Como dissemos anteriormente, sistemas que possuem esta propriedade são chamados de sistemas de equações normais. No Capítulo 3 mostraremos como obter um sistema de equações normais. Portanto a direção dy, neste caso, é dada pela solução de um sistema de equações normais.

Com estas direções temos o método primal-dual afim escala [26].

Algoritmo 2.1 *Método Primal-Dual Afim Escala*
Para $k = 0, 1, \ldots, max$

$$
\begin{aligned}
&r_p = b - Ax \\
&r_d = c - A^t y - z \\
&r_a = -XZe \\
&D = X^{-1}Z \\
&dy = (AD^{-1}A^t)^{-1}(r_p + A(D^{-1}r_d - Z^{-1}r_a)) \\
&dx = D^{-1}(A^t dy - r_d + X^{-1}r_a) \\
&dz = X^{-1}(r_a - Zdx) \\
&\alpha_p = min(1, -\tau/min(dx_i/x_i)) \ para \ dx_i < 0 \\
&\alpha_d = min(1, -\tau/min(dz_i/z_i)) \ para \ dz_i < 0 \\
&x = x + \alpha_p dx \\
&y = y + \alpha_d dy \\
&z = z + \alpha_d dz
\end{aligned}
$$

até convergir.

Neste método não precisamos de um ponto inicial factível, basta apenas que ele seja interior. O método primal-dual afim escala não converge ou converge muito lentamente para diversos problemas pois os produtos $x_i z_i$ se aproximam de zero com taxas muito diferentes entre si [36].

2.3 Método Primal-Dual Seguidor de Caminhos

No método apresentado anteriormente vimos que os valores de x e z se aproximam rapidamente dos seus limites. Uma solução para esta dificuldade consiste em perturbar o lado direito das restrições $XZe = 0$. O valor médio dos pares $x_i z_i$ é $\sum_{i=1}^{n} \frac{x_i z_i}{n} = \frac{\gamma}{n}$. Este seria o valor ideal para cada par $x_i z_i$.

Portanto, a perturbação pode ser uma função deste valor. Simultaneamente, reduzimos o valor médio de $x_i z_i$ multiplicando $\frac{\gamma}{n}$ por $\sigma \in (0, 1)$. Desta forma temos a perturbação $\mu^k = \sigma^k \frac{\gamma^k}{n}$ e $\{\mu^k\} \to 0$ e obtemos assim a solução do problema original quando o método converge.

Esta classe de métodos é chamada de método seguidor de caminho.

Algoritmo 2.2 *Método Primal-Dual Seguidor de Caminho*
Dados (x_0, y_0, z_0), $x_0 > 0$, $z_0 > 0$, $\tau \in (0,1)$ *e* $\sigma \in (0,1)$
Para $k = 0, 1, \ldots, max$

$$
\begin{aligned}
&r_p = b - Ax \\
&r_d = c - A^t y - z \\
&\gamma = x^t z \\
&\mu = \sigma \frac{\gamma}{n} \\
&r_c = \mu e - XZe \\
&D = X^{-1} Z \\
&dy = (AD^{-1}A^t)^{-1}(r_p + A(D^{-1}r_d - Z^{-1}r_c)) \\
&dx = D^{-1}(A^t dy - r_d + X^{-1}r_c) \\
&dz = X^{-1}(r_c - Zdx) \\
&\alpha_p = min(1, -\tau/min(dx_i/x_i)) \ para \ dx_i < 0 \\
&\alpha_d = min(1, -\tau/min(dz_i/z_i)) \ para \ dz_i < 0 \\
&x = x + \alpha_p dx \\
&y = y + \alpha_d dy \\
&z = z + \alpha_d dz
\end{aligned}
$$

até convergir.

Se usarmos $\sigma = 0$ obtemos o método afim escala. Com $\sigma = 1$ obtemos a direção de centragem. Combinações de σ_k e τ_k diferentes levam a propriedades [36] tais como:

- convergência superlinear;
- complexidade polinomial;
- ambos.

2.4 Método Preditor-Corretor

A partir do método primal-dual afim escala podemos construir o método preditor-corretor [26, 25]. Calculamos o resíduo da complementaridade na iteração seguinte quando $\alpha_p = \alpha_d = 1$:

$$
\begin{aligned}
x_{j+1} &= x_j + dx, \\
z_{j+1} &= z_j + dz.
\end{aligned}
$$

No método primal-dual afim escala temos

$$X_{j+1}Z_{j+1}e = (X + Dx)(Z + Dz)e =$$

$$= XZe + XDze + DxZe + DxDze =$$

$$= -r_a + \underbrace{Xdz + Zdx}_{r_a} + DxDze =$$
$$= DxDze = -r_{aj+1}.$$

Utilizando este resultado o método preditor-corretor calcula duas direções:

1. direção afim escala,
2. direção de correção, composta de:
   a) correção não-linear,
   b) perturbação (centragem).

A perturbação é calculada na segunda direção pois podemos verificar a qualidade da primeira direção e calcular a perturbação de acordo com ela.

A primeira direção é a direção afim escala $(\widetilde{dx}, \widetilde{dy}, \widetilde{dz})$:

$$\begin{cases} A\widetilde{dx} = r_p \\ A^t\widetilde{dy} + \widetilde{dz} = r_d \\ X\widetilde{dz} + Z\widetilde{dx} = r_a. \end{cases}$$

A segunda direção $(\hat{dx}, \hat{dy}, \hat{dz})$ é calculada assumindo $\alpha_p = \alpha_d = 1$ e com perturbação μ mas utilizando a mesma matriz do sistema anterior, ou seja, usamos X e Z no lugar de $X + \widetilde{D}x$ e $Z + \widetilde{D}z$. Assim:

$$\begin{cases} A\hat{dx} = 0 \\ A^t\hat{dy} + \hat{dz} = 0 \\ X\hat{dz} + Z\hat{dx} = \mu e - \widetilde{D}x\widetilde{D}ze. \end{cases}$$

A perturbação pode ser melhor calculada de acordo com a direção afim escala. Baseado nisto Mehrotra [25] desenvolveu a seguinte heurística: a segunda direção $(\hat{dx}, \hat{dy}, \hat{dz})$ é calculada com a perturbação $\mu = (\widetilde{\gamma}/\gamma)^3(\gamma/n)$, onde $\widetilde{\gamma} = (x + \widetilde{\alpha}_p\widetilde{dx})^t(z + \widetilde{\alpha}_d\widetilde{dz})$.

Desta forma, a direção final é dada pela soma das duas direções:

$$(dx, dy, dz) = (\widetilde{dx}, \widetilde{dy}, \widetilde{dz}) + (\hat{dx}, \hat{dy}, \hat{dz}).$$

Note que somando os dois sistemas obtemos:

$$\begin{cases} Adx = r_p \\ A^tdy + dz = r_d \\ Xdz + Zdx = r_a + \mu e - \widetilde{D}x\widetilde{D}ze = r_s. \end{cases}$$

Podemos calcular a direção final resolvendo diretamente este sistema.

Algoritmo 2.3 *Método Preditor-Corretor*
Para $k = 0, 1, \ldots, max$

$$
\begin{aligned}
&r_p = b - Ax \\
&r_d = c - A^t y - z \\
&r_a = -XZe \\
&D = X^{-1}Z \\
&\widetilde{d}y = (AD^{-1}A^t)^{-1}(r_p + A(D^{-1}r_d - Z^{-1}r_a)) \\
&\widetilde{d}x = D^{-1}(A^t\widetilde{d}y - r_d + X^{-1}r_a) \\
&\widetilde{d}z = X^{-1}(r_a - Z\widetilde{d}x) \\
&\widetilde{\alpha}_p = min(1, -\tau/min(\widetilde{d}x_i/x_i)) \ \textit{para} \ \widetilde{d}x_i < 0 \\
&\widetilde{\alpha}_d = min(1, -\tau/min(\widetilde{d}z_i/z_i)) \ \textit{para} \ \widetilde{d}z_i < 0 \\
&\widetilde{\gamma} = (x + \widetilde{\alpha}_p\widetilde{d}x)^t(z + \widetilde{\alpha}_d\widetilde{d}z) \\
&\gamma = x^t z \\
&\mu = (\widetilde{\gamma}/\gamma)^3(\gamma/n) \\
&r_s = \mu e + r_a - \widetilde{D}x\widetilde{D}ze \\
&dy = (AD^{-1}A^t)^{-1}(r_p + A(D^{-1}r_d - Z^{-1}r_s)) \\
&dx = D^{-1}(A^t dy - r_d + X^{-1}r_s) \\
&dz = X^{-1}(r_s - Zdx) \\
&\alpha_p = min(1, -\tau/min(dx_i/x_i)) \ \textit{para} \ dx_i < 0 \\
&\alpha_d = min(1, -\tau/min(dz_i/z_i)) \ \textit{para} \ dz_i < 0 \\
&x = x + \alpha_p dx \\
&y = y + \alpha_d dy \\
&z = z + \alpha_d dz
\end{aligned}
$$

até convergir.

O método preditor-corretor gera uma sequência de vetores (x_j, y_j, z_j) que se aproximam da solução do PPL (x^*, y^*, z^*). Dizemos que essa sequência converge, isto é, que o método preditor-corretor converge quando $\lim_{j\to\infty} \|(x_j, y_j, z_j) - (x^*, y^*, z^*)\| = 0$, ou seja, se existe uma constante $M \in [0, 1)$ e um inteiro $\hat{j} \geq 0$ tais que para todo $j \geq \hat{j}$,

$$\|(x_{j+1}, y_{j+1}, z_{j+1}) - (x^*, y^*, z^*)\| \leq M \|(x_j, y_j, z_j) - (x^*, y^*, z^*)\| . \tag{2.8}$$

Quando o método converge como em (2.8) dizemos que o método tem convergência linear. Porém o método preditor-corretor tem uma taxa de convergência mais rápida, isto é, converge quadraticamente:

$$\|(x_{j+1}, y_{j+1}, z_{j+1}) - (x^*, y^*, z^*)\| \leq M \|(x_j, y_j, z_j) - (x^*, y^*, z^*)\|^2 . \tag{2.9}$$

Portanto o método preditor-corretor tem taxa de convergência quadrática desde que o cálculo das direções seja considerado como uma única iteração [36]. Devido as suas características e, na nomeclatura da programação não linear, ele poderia ser chamado de método de Newton perturbado (μ), modificado $(x, z) > 0$, de ordem 1 (correção não-linear).

O critério de convergência que utilizaremos será baseado nas condições de otimalidade:

$$\frac{\|b - Ax\|}{\|b\| + 1} \leq \epsilon,$$

$$\frac{\|c - A^t y - z\|}{\|c\| + 1} \leq \epsilon \text{ e}$$

$$\frac{|c^t x - b^t y|}{|c^t x|\,|b^t y| + 1} \leq \epsilon, \text{ onde } \epsilon = 10^{-8}, \text{por exemplo.}$$

Neste trabalho utilizamos o método preditor-corretor para obter a solução de um problema de programação linear.

2.5 O Sistema de Equações Normais

Observamos na seção anterior que o cálculo das direções $\widetilde{dy}$ e dy no método preditor-corretor envolve a solução de um sistema de equações lineares. Obter a solução destes sistemas é o ponto crítico do método, visto que este é o passo que envolve maior esforço e tempo computacional. Portanto precisamos buscar formas eficientes de solucionar estes sistemas.

Duas abordagens podem ser utilizadas para obter as direções do método, uma baseada no sistema aumentado e outra, no sistema de equações normais. No primeiro caso, temos que a matriz dos coeficientes do sistema é uma matriz esparsa, simétrica ($A^t = A$) e indefinida [1] e, no segundo caso, a matriz é simétrica e definida positiva [2]. Em ambos os casos existem métodos diretos e iterativos para obter a solução do sistema. Na prática, as implementações que trabalham com o sistema aumentado, tipicamente se mostram inferiores quando comparadas com implementações que utilizam o sistema de equações normais, no ponto de vista do tempo do processamento. O sistema aumentado é mais interessante na solução de problemas de programação quadrática e não-linear.

Já a segunda abordagem, apresentada na Seção 2.2.2, para obter as direções de Newton resulta em um sistema de equações normais, isto é, a matriz dos coeficientes desse sistema pode ser escrita como o produto de uma matriz por sua transposta:

$$(AD^{-\frac{1}{2}})(D^{-\frac{1}{2}}A^t)dy = (AD^{-1}A^t)dy = (r_p + A(D^{-1}r_d - Z^{-1}r_a))^3. \tag{2.10}$$

As formas de se obter um sistema de equações normais serão apresentadas no Capítulo 3. Esses sistemas tem matrizes com algumas propriedades: são simétricas e definidas positivas. Portanto, uma opção para encontrar a solução de sistemas de equações normais, como o sistema (2.10), é através de métodos diretos, por exemplo, a fatoração de Cholesky. Entretanto, quando temos problemas com variáveis livres ocorrem instabilidades numéricas.

[1] $A \in \Re^{nxn}$ é indefinida se, somente se, existem vetores $x \in \Re^n$ tais que $x^t Ax > 0$ e, existem também, vetores $y \in \Re^n$ tais que $y^t Ay < 0$.

[2] $A \in \Re^{nxn}$ é definida positiva se, somente se, $x^t Ax > 0$ para todos vetores não nulos $x \in \Re^n$.

[3] Sistema de equações normais do Método Preditor-Corretor.

Outro grande problema, quando utilizamos métodos ditetos, é a perda da esparsidade da matriz $AD^{-1}A^t$ quando temos colunas densas em A.

A perda de esparsidade pode ser contornada com o uso de métodos iterativos. Nestes métodos, a matriz $AD^{-1}A^t$ não precisa ser construída, a menos que seja necessária para construção do precondicionador. Um método iterativo muito utilizado é o método dos gradientes conjugados precondicionado. No Capítulo 3 apresentaremos o método dos gradientes conjugados e algumas de suas versões. Duas das versões que apresentaremos serão utilizadas nos experimentos númericos e estas serão comparadas com a versão clássica do método dos gradientes conjugados precondicionado.

2.6 Considerações Finais

Apresentamos neste capítulo o Método Preditor-Corretor. Este método de pontos interiores será o método utilizado nos experimentos numéricos. Iremos determinar as direções de Newton deste método a partir da solução do sistema de equações normais. Obter essas direções é o ponto crítico desse método e para isso iremos trabalhar com métodos iterativos na busca dessas soluções. No próximo capítulo, trataremos dos métodos iterativos adotados nos experimentos numéricos, no caso, o método de gradientes conjugados (MGC) e suas diferentes versões. O MGC resolve sistemas com a matriz dos coeficientes simétrica e definida positiva e, o sistema que desejamos resolver, possui uma matriz que tem essas propriedades. Mais adiante discutiremos também o uso dos precondicionadores nos métodos iterativos.

Capítulo 3

Métodos Iterativos para Solução de Sistemas Lineares

Existem diferentes formas de encontrar a solução de um sistema linear do tipo $Ax = b$. Podemos utilizar métodos diretos ou métodos iterativos e existem diversos para as duas abordagens.

Dentre os métodos diretos para se resolver um sistema do tipo $Ax = b$, os mais utilizados são baseados na fatoração da matriz do sistema linear em fatores triangulares, por exemplo, a fatoração LU [16]. Porém, à medida em que a dimensão dos problemas aumenta, o custo computacional pode ficar muito alto. Uma boa alternativa para solucionar esses sistemas é o uso de métodos iterativos. Uma forma simples de se obter um método iterativo seria, por exemplo:

$$x_{j+1} = x_j - \tau dx_j, \tag{3.1}$$

onde τ é um parâmetro e dx_j uma direção de busca. Porém, neste caso, não temos garantia da convergência do método. Vamos supor agora que desejamos resolver o sistema $Ax = b$ utilizando o processo iterativo descrito pela equação (3.1). No entanto, faremos $dx_j = Ax_j - b$ e o ponto inicial x_0 é dado. Logo, nosso método iterativo assume a seguinte forma:

$$x_{j+1} = x_j - \tau(Ax_j - b),\ j = 0, 1, \ldots,$$

onde é τ é um parâmetro. Se tivermos $0 < \tau < \dfrac{2}{\lambda_{max}}$, onde λ_{max} é o maior autovalor da matriz $(I - \tau A)$, a sequência x_k converge para qualquer x_0 inicial [32].

Portanto, uma vantagem de se trabalhar com métodos iterativos consiste na simplicidade computacional pois trabalhamos apenas com produtos de matriz por vetor e soma de vetores. A desvantagem é que os métodos iterativos podem ter uma taxa de convergência lenta ou até mesmo divergir.

Neste capítulo daremos uma breve introdução aos métodos de projeção e, em seguida, apresentaremos o método dos gradientes conjugados e algumas de suas versões.

3.1 Métodos de Projeção

Muitas das técnicas iterativas para resolver sistemas lineares de grande porte utilizam um processo de projeção. Assim, dado um sistema $Ax = b$, $A : n \times n$, os métodos de projeção [32] buscam uma solução aproximada em um subspaço de $\Re^n$. Este subespaço é chamado de subespaço de busca e será denotado por K. Se a dimensão de K for m então m restrições de ortogonalidade necessitam ser impostas para que consigamos a aproximação desejada. Isto é, o vetor resíduo, $r = b - Ax$ é construído de forma que seja ortogonal a m vetores linearmente independentes. Assim definimos outro subspaço de dimensão m chamado de subespaço de restrições e este será denotado por L.

Existem duas possibilidades para a escolha do subespaço L. Se L for o mesmo subespaço que K teremos a classe dos métodos de projeção ortogonal; caso contrário obtemos médotos de projeção oblíqua.

Os métodos de projeção procuram encontrar um vetor $\tilde{x}$ em K tal que o vetor resíduo $r = b - A\tilde{x}$ seja ortogonal à L. Dada uma solução inicial x_0 temos que $\tilde{x} = x_0 + \delta$, com $\delta \in K$, e $r_0 = b - Ax_0$. Com isso a solução aproximada pode ser definida como:

$$\tilde{x} = x_0 + \delta, \delta \in K$$

$$w^t(r_0 - A\delta) = \langle r_0 - A\delta, w\rangle = 0, \forall w \in L$$

onde $\langle a, b\rangle$ é o produto interno entre os vetores a e b.

Escolhidos os subespaços K e L e chamando de $V = \{v_1, \ldots, v_m\}$ e $W = \{w_1, \ldots, w_m\}$ as bases de K e L respectivamente, podemos escrever:

$$\tilde{x} = x_0 + Vy,$$

e, fazendo $\delta = Vy$ na restrição de ortogonalidade, obtemos:

$$W^tAVy = W^tr_0.$$

A partir deste ponto podemos escrever um algoritmo de um método de projeções, considerando que W^tAV seja não-singular:

Algoritmo 3.1 *Método de Projeção*
Até convergir, faça:

$$r = b - Ax$$
$$y = (W^tAVy)^{-1}W^tr$$
$$x = x + Vy$$

Fim

Vamos considerar o caso em que $K = L$, isto é, um método de projeção ortogonal. Assim, considerando V e W bases de K e L respectivamente, podemos escrever $W = VF$, onde F é uma matriz não-singular. Portanto, denotando B como a matriz W^tAV, temos:

$$B = F^tV^tAV.$$

Desde que a matriz A seja definida positiva a matriz V^tAV também será e isso garante que a matriz B será inversível. A partir daqui, vamos considerar que a matriz A seja definida positiva.

Com essa consideração e denotando por x^* na solução exata do sistema linear $Ax = b$ e x_0 uma solução inicial, um método de projeção ortogonal minimiza, na norma A, o vetor erro $x^* - \tilde{x}$ em $x_0 + K$, isto é:

$$E(\tilde{x}) = min_{x \in x_0+K} E(x) = min_{x \in x_0+K} \langle A(x^* - x), x^* - x \rangle^{\frac{1}{2}}.$$

Observemos que para $\tilde{x}$ ser o vetor que minimiza o erro é necessário e suficiente que $x^* - \tilde{x}$ seja A-ortogonal a todo vetor do subespaço K, ou seja:

$$\langle A(x^* - \tilde{x}), v \rangle = \langle b - A\tilde{x}, v \rangle = 0, \forall v \in K.$$

Como estamos no caso $K = L$, $\langle b - A\tilde{x}, v \rangle = \langle \tilde{r}, v \rangle = 0$ é exatamente a condição imposta para o resíduo em um método de projeção.

No nosso trabalho utilizaremos $K = L$ e K será o subespaço de Krylov definido por $K_m(v, A) = span\{v, Av, A^2v, \ldots, A^{m-1}v\}$.

3.2 Método dos Gradientes Conjugados

Existem formas diferentes de se resolver um sistema linear com a matriz dos coeficientes simétrica definida positiva (SDP), uma delas é o método de gradientes conjugados (MGC). Em poucas palavras, podemos dizer que o método realiza uma projeção ortogonal no subespaço de Krylov $K_m(r_0, A) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\}$ [18], onde r_0 é o resíduo inicial.

Existem outros métodos de projeção ortogonal, podemos citar: o método de Arnoldi [18], método do resíduo mínimo generalizado [18], o método de Lanczos [32] e o método de D-Lanczos [32].

Uma forma de se obter o método dos gradientes conjugados é baseada no método de máxima descida e das direções conjugadas. Estes métodos se baseiam na minimização de uma função quadrática. Assim, dado um sistema de equações lineares $Ax = b$, onde $A \in \Re^{n \times n}$ é simétrica e definida positiva, uma forma quadrática é uma função $f : \Re^{n \times n} \to \Re$, dada por:

$$f(x) = \frac{1}{2}x^tAx - b^tx + c,$$

com c constante.

Sabemos que resolver o sistema $Ax = b$ é equivalente a encontrar um ponto estacionário da função quadrática $f(x)$. Assim dado um ponto arbitrário x, a direção de maior crescimento da função $f(x)$ é dada pela direção do vetor gradiente. Os pontos críticos da função são os vetores $\bar{x}$ tais que $\nabla f(\bar{x}) = A\bar{x} - b = 0$. Assim todo vetor $\bar{x}$ é uma solução do sistema $Ax = b$. Entretanto, a matriz A é simétrica e definida positiva, portanto, o sistema $Ax = b$

tem solução única e esta solução é o mínimo de $f(x)$. Logo resolver o sistema $Ax = b$ e encontrar a solução que minimiza a função $f(x)$ são cálculos equivalentes.

O método de máxima descida, também chamado de método do gradiente ou método de Cauchy, é uma abordagem em que as iterações de x_j são escolhidas na direção oposta ao do gradiente, isto é, na direção em que f decresce mais rapidamente. Temos na iteração j a seguinte relação:

$$d_j = -\nabla f(x_j) = b - Ax_j = r_j.$$

Caso o resíduo r_j não seja nulo, tomamos $\alpha_j = \dfrac{r_j^t r_j}{r_j^t A r_j}$ positivo, tal que, $f(x_{j+1}) = f(x_j + \alpha_j r_j) < f(x_j)$.

O método das direções conjugadas é um método bem similar ao método de máxima descida. A diferença entre os métodos ocorre na escolha dos vetores direção. No método de máxima descida, estas direções são ortogonais e no método das direções conjugadas os vetores direção são A-conjugados, isto é, $\langle Ad_j, d_k\rangle = d_k^t A d_j = 0$, $k \geq 1$, $j \leq n$ e $k \neq j$.

Fazendo $\alpha_j = -\dfrac{d_j^t r_j}{d_j^t A d_j}$ temos o valor de α_j que irá minimizar a função f no ponto x_{j+1}. As direções conjugadas serão geradas de forma que sejam linearmente independentes utilizando o processo de ortogonalização de Gram-Schmidt.

Visto esses dois métodos, o método dos gradientes conjugados será o método das direções conjugadas com as direções de busca sendo construídas por conjugação dos resíduos.

Chamando agora as direções de busca por p, o vetor x_j pode ser expresso como

$$x_{j+1} = x_j + \alpha_j p_j.$$

O vetor resíduo pode ser calculado recursivamente

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{3.2}$$

Como os vetores resíduo, r_j, são ortogonais, teremos necessariamente que $\langle r_j - \alpha_j A p_j, r_j\rangle = 0$ e, como resultado

$$\alpha_j = \frac{\langle r_j, r_j\rangle}{\langle Ap_j, r_j\rangle}.$$

As direções de busca p_{j+1} são combinações lineares de r_{j+1} e p_j, assim

$$p_{j+1} = r_{j+1} + \beta_j p_j.$$

Usando esta última relação temos que

$$\langle Ap_j, r_j\rangle = \langle Ap_j, p_j\rangle\,,$$

pois Ap_j é ortogonal a p_{j-1}. Temos ainda que p_{j+1} deve ser ortogonal a Ap_j. Impondo esta condição temos que:

$$\beta_j = -\frac{\langle r_{j+1}, Ap_j\rangle}{\langle p_j, Ap_j\rangle}. \tag{3.3}$$

Combinado as equações (3.2) e (3.3) obtemos

$$\beta_j = \frac{1}{\alpha_j}\frac{\langle r_{j+1}, (r_{j+1} - r_j)\rangle}{\langle Ap_j, p_j\rangle} = \frac{\langle r_{j+1}, r_{j+1}\rangle}{\langle r_j, r_j\rangle}.$$

Com estas relações temos o método de gradientes conjugados.

Algoritmo 3.2 *Método dos Gradientes Conjugados*
Calcule $r_0 = b - Ax_0, p_0 = r_0$
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$\begin{aligned}
\alpha_j &= \frac{\langle r_j, r_j\rangle}{\langle Ap_j, p_j\rangle}\\
x_{j+1} &= x_j + \alpha_j p_j\\
r_{j+1} &= r_j - \alpha_j A p_j\\
\beta_j &= \frac{\langle r_{j+1}, r_{j+1}\rangle}{\langle r_j, r_j\rangle}\\
p_{j+1} &= r_{j+1} + \beta_j p_j
\end{aligned}$$

Fim

É importante observar que o método acima necessita de quatro vetores (x, p, Ap, r) além da matriz A.

3.3 Método dos Gradientes Conjugados e Equações Normais

Um sistema de equações lineares, em diversos casos, não tem a sua matriz dos coeficientes simétrica ou, nem mesmo, quadrada. Estas matrizes podem ter maior ou menor número de linhas ou colunas. Nestes casos, onde a matriz dos coeficientes é retangular, não podemos utilizar o método dos gradientes conjugados. Como vimos na seção anterior, este método necessita que a matriz A do sistema $Ax = b$ seja simétrica e definida positiva. Assim, precisamos obter um sistema equilavente no qual a matriz dos coeficientes atenda estas condições.

Assim, dado um sistema qualquer $Ax = b$, existem duas possibilidades para se obter o novo sistema nas condições desejadas, são elas:

- premultiplicando o sistema original por A^t ou
- substituindo a variável x por $A^t u$.

Desta forma, o novo sistema será simétrico e definido positivo, satisfazendo as condições necessárias para aplicarmos o método dos gradientes conjugados. Observe ainda, que esses sistemas são sistemas de equações normais.

3.3.1 Versão Normal Residual -NR

Como o apresentado na seção anterior, um sistema qualquer $Ax = b$ pode ser transformado em um sistema com a matriz dos coeficientes simétrica e definida positiva. Existem duas formas de se obter esse sistema, apresentaremos agora a primeira delas, na qual premultiplicamos o sistema original $Ax = b$ por A^t.

Quando premultiplicamos o sistema original por A^t, podemos aplicar diretamente o método de gradientes conjugados no novo sistema, $A^tAx = A^tb$, e obter o algoritmo GCNR (N de "Normal" e R de "Residual" [32]). Assim, denotando por z_j o vetor resíduo no passo j, obtemos a seguinte sequência de operações:

- $\alpha_j = \dfrac{\langle z_j, z_j\rangle}{\langle A^tAp_j, p_j\rangle} = \dfrac{\langle z_j, z_j\rangle}{\langle Ap_j, Ap_j\rangle}$,
- $x_{j+1} = x_j + \alpha_j p_j$,
- $z_{j+1} = z_j - \alpha_j A^tAp_j$,
- $\beta_j = \dfrac{\langle z_{j+1}, z_{j+1}\rangle}{\langle z_j, z_j\rangle}$,
- $p_{j+1} = z_{j+1} + \beta_j p_j$.

O resíduo original continua a ser calculado em cada passo e o novo resíduo, das equações normais, em duas etapas: $r_{j+1} = r_j - \alpha_j Ap_j$ e então $z_{j+1} = A^tr_{j+1}$. Também é conveniente introduzir o vetor $w_j = Ap_i$. Com estas definições, o GCNR tem a seguinte forma:

Algoritmo 3.3 *Método dos Gradientes Conjugados - Normal Residual - (GCNR)*
Calcule $r_0 = b - Ax_0$, $z_0 = A^tr_0$, $p_0 = z_0$.
Para $j = 0, \ldots$ *até convergir, faça:*

$$
\begin{aligned}
w_j &= Ap_j \\
\alpha_j &= \frac{\langle z_j, z_j\rangle}{\langle w_j, w_j\rangle} \\
x_{j+1} &= x_j + \alpha_j p_j \\
r_{j+1} &= r_j - \alpha_j w_j \\
z_{j+1} &= A^tr_{j+1} \\
\beta_j &= \frac{\langle z_{j+1}, z_{j+1}\rangle}{\langle z_j, z_j\rangle} \\
p_{j+1} &= z_{j+1} + \beta_j p_j
\end{aligned}
$$

Fim

Assim temos o método de gradientes conjugados para o sistema de equações normais do tipo $A^tAx = A^tb$. A seguir veremos a versão NE deste método.

3.3.2 Versão Normal Error - NE

Apresentaremos agora a versão do método dos gradientes conjugados quando o sistema de equações normais é obtido pela substituição da variável x por $A^t u$ no sistema $Ax = b$, ou seja, a segunda forma de se obter o sistema de equações normais. Portanto, a versão NE (N de "Normal" e E de "Error" [32]) é obtida de forma similar quando aplicamos o método de gradientes conjugados no sistema $AA^t u = b$ onde $x = A^t u$. Denotando por q_j as direções conjugadas, uma iteração do método de gradientes conjugados na variável u pode ser escrita da seguinte forma:

- $\alpha_j = \dfrac{\langle r_j, r_j \rangle}{\langle AA^t q_j, q_j \rangle} = \dfrac{\langle r_j, r_j \rangle}{\langle A^t q_j, A^t q_j \rangle}$,
- $u_{j+1} = u_j + \alpha_j q_j$,
- $r_{j+1} = r_j - \alpha_j AA^t q_j$,
- $\beta_j = \dfrac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$,
- $q_{j+1} = r_{j+1} + \beta_j q_j$.

Podemos, portanto, escrever o método dos gradientes conjugados (versão NE) para a variável u.

Algoritmo 3.4 *Método dos Gradientes Conjugados - Normal Error na variável u*
Calcule $r_0 = b - AA^t u$, $q_0 = r_0$.
Para $j = 0, 1, \ldots$ *até convergir, faça:*

$$
\begin{aligned}
p_j &= A^t q_j \\
\alpha_j &= \frac{\langle r_j, r_j \rangle}{\langle q_j, q_j \rangle} \\
u_{j+1} &= u_j + \alpha_j q_j \\
r_{j+1} &= r_j - \alpha_j A q_j \\
\beta_j &= \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle} \\
q_{j+1} &= r_{j+1} + \beta_j q_j
\end{aligned}
$$

Fim

O valor da variável x pode ser calculado, ao final deste método, fazendo o produto $x = A^t u$.

Outra opção poder ser obtida escrevendo cada iteração na variável original $x_j = x_0 + A^t(u_j - u_0)$. Para tanto introduziremos o vetor $p_j = A^t q_j$ no cálculo inicial das variáveis. Assim os vetores resíduo de x_j e u_j são os mesmos. Os vetores q_j já não são necessários pois os vetores p_j podem ser obtidos como $p_{j+1} = A^t r_{j+1} + \beta_j p_j$. O método que resulta destes passos é o GCNE, também conhecido como método de Craig [32].

Algoritmo 3.5 *Método dos Gradientes Conjugados - Normal Error (GCNE) - Método de Craig*
Calcule $r_0 = b - Ax_0, p_0 = A^t r_0$.
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$\alpha_j = \frac{\langle r_j, r_j \rangle}{\langle p_j, p_j \rangle}$$
$$x_{j+1} = x_j + \alpha_j p_j$$
$$r_{j+1} = r_j - \alpha_j A p_j$$
$$\beta_j = \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$$
$$p_{j+1} = A^t r_{j+1} + \beta_j p_j$$

Fim

Os dois métodos, GCNR e GCNE, buscam aproximar a solução ao longo do mesmo subespaço

$$x_0 + A^t K_m(AA^t, r_0) = x_0 + K_m(A^t A, A^t r_0)$$

mas com diferentes propriedades de otimalidade:

- mínimo resíduo para GCNR e
- mínimo erro para GCNE.

3.4 Precondicionadores

Pode acontecer que os autovalores da matriz A, na resolução de um sistema linear $Ax = b$, estejam muito dispersos [16]. Isso pode prejudicar o desempenho do método aplicado para buscar a solução desse sistema. Neste caso, é importante o uso dos precondicionadores. O precondicionamento é uma estratégia aplicada com o objetivo de melhorar as características de convergência do método utilizado para que este possa ser mais eficiente. O precondicionamento pode ser feito premultiplicando o sistema $Ax = b$ pela inversa de uma matriz auxiliar M. Desta forma teremos o seguinte sistema:

$$M^{-1}Ax = M^{-1}b.$$

Para um precondicionamento ser eficiente é importante que algumas condições sejam atendidas, são elas:

- as operações para a construção da matriz precondicionadora M devem ser de baixo custo, para que a matriz possa ser construída facilmente,
- M deve ter esparsidade próxima da matriz A e
- os autovalores de $M^{-1}A$ devem estar agrupados ou próximos da unidade.

Exitem formas diferentes para se construir um precondicionador e, dependendo da escolha, teremos uma boa contribuição para a convergência do método. De maneira geral, temos duas classes. Os precondicionadores construídos baseados em técnicas algébricas e aqueles projetados para aplicações particulares. Os primeiros, por exemplo, podem ser construídos utilizando fatorações incompletas da matriz e assim podem ser utilizados em muitas aplicações pois não é necessária nenhuma informação específica do problema que gostaríamos de resolver. Já a segunda classe é construída explorando detalhes do problema, como a estrutura, por exemplo [8].

Apresentaremos a seguir algumas versões precondicionadas do método dos gradientes conjugados. No Capítulo 4 voltaremos a tratar dos precondicionadores e apresentaremos aqueles utilizados neste trabalho.

3.5 Método dos Gradientes Conjugados Precondicionado

Como vimos, uma forma de aumentar a eficiência e a robustez dos métodos iterativos consiste em utilizar precondicionadores. Um precondicionador M válido para o método de gradientes conjugados deve ser simétrico definido positivo. Observe que a matriz $M^{-1}A$ é auto-adjunta usando o produto interno em relação à matriz M:

$$\langle x, y\rangle_M \equiv \langle Mx, y\rangle = \langle x, My\rangle \,,$$

então

$$\left\langle M^{-1}Ax, y\right\rangle_M = \langle Ax, y\rangle = \langle x, Ay\rangle = \left\langle x, M(M^{-1}A)y\right\rangle = \left\langle x, M^{-1}Ay\right\rangle_M.$$

Usando o produto interno em relação à matriz M e denotando por $r_j = b - Ax_j$ o resíduo original e por $z_j = M^{-1}r_j$ o resíduo para o sistema precondicionado, temos a seguinte sequência de operações do método de gradientes conjugados precondicionado:

- $\alpha_j = \dfrac{\langle z_j, z_j\rangle_M}{\langle M^{-1}Ap_j, p_j\rangle_M}$,
- $x_{j+1} = x_j + \alpha_j p_j$,
- $r_{j+1} = r_j - \alpha_j Ap_j$ e $z_{j+1} = M^{-1}r_{j+1}$,
- $\beta_j = \dfrac{\langle z_{j+1}, z_{j+1}\rangle_M}{\langle z_j, z_j\rangle_M}$,
- $p_{j+1} = z_{j+1} + \beta_j p_j$.

Como $\langle z_j, z_j\rangle_M = \langle r_j, z_j\rangle$ e $\langle M^{-1}Ap_j, p_j\rangle_M = \langle Ap_j, p_j\rangle$, não precisamos utilizar o produto interno em relação à matriz M. Desta forma obtemos o seguinte método:

Algoritmo 3.6 *Método dos Gradientes Conjugados Precondicionado (GCP)*
Calcule $r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$ *e* $p_0 = z_0$.
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$\begin{aligned}
\alpha_j &= \frac{\langle r_j, z_j \rangle}{\langle Ap_j, p_j \rangle} \\
x_{j+1} &= x_j + \alpha_j p_j \\
r_{j+1} &= r_j - \alpha_j A p_j \\
z_{j+1} &= M^{-1} r_{j+1} \\
\beta_j &= \frac{\langle r_{j+1}, z_{j+1} \rangle}{\langle r_j, z_j \rangle} \\
p_{j+1} &= z_{j+1} + \beta_j p_j
\end{aligned}$$

Fim

3.5.1 Gradientes Conjugados Precondicionado para Equações Normais

Existem diferentes versões do GCP para equações normais: precondicionada à esquerda para o GCNR e para o GCNE e, ainda, precondicionada à direita para o GCNR e para o GCNE. Vamos destacar apenas duas: com o precondicionador à esquerda para o GCNR e GCNE. Denominaremos de PGCNR e PGCNE, respectivamente, cada uma destas versões.

O PGCNR é facilmente derivado do Algoritmo 3.6. Lembrando que nesta versão do método consideramos que o sistema original $Ax = b$ foi premultiplicado pela matriz A^t, isto é, aplicaremos o precondicionamento ao sistema de equações normais $A^tAx = A^tb$. Vamos denotar por $r_j = b - Ax_j$, o resíduo do sistema original, $\tilde{r}_j = A^t r_j$ o resíduo para o sistema de equações normais e, finalmente, o resíduo do sistema precondicionado por $z_j = M^{-1}\tilde{r}_j$. O escalar α_j será dado por:

$$\alpha_j = \frac{\langle \tilde{r}_j, z_j \rangle}{\langle A^t A p_j, p_j \rangle} = \frac{\langle \tilde{r}_j, z_j \rangle}{\langle Ap_j, Ap_j \rangle}.$$

As relações acima nos sugerem o uso de um vetor auxiliar $w_j = Ap_j$ no método que assume a seguinte forma.

Algoritmo 3.7 *Método dos Gradientes Conjugados Precondicionado - Normal Residual (PGCNR)*
Calcule $r_0 = b - Ax_0$, $\tilde{r}_0 = A^t r_0$, $z_0 = M^{-1}\tilde{r}_0$, $p_0 = z_0$.
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$\begin{aligned}
w_j &= Ap_j \\
\alpha_j &= \frac{\langle z_j, \tilde{r}_j \rangle}{\langle w_j, w_j \rangle} \\
x_{j+1} &= x_j + \alpha_j p_j
\end{aligned}$$

$$
\begin{aligned}
r_{j+1} &= r_j - \alpha_j w_j \\
\tilde{r}_{j+1} &= A^t r_{j+1} \\
z_{j+1} &= M^{-1}\tilde{r}_{j+1} \\
\beta_j &= \frac{\langle z_{j+1}, \tilde{r}_{j+1}\rangle}{\langle z_j, \tilde{r}_j\rangle} \\
p_{j+1} &= z_{j+1} + \beta_j p_j
\end{aligned}
$$

Fim

Analogamente, o sistema $AA^t u = b$ com $x = A^t u$ pode ser precondicionado à esquerda e resolvido aplicando o método GCP. Observe que neste caso a atualização da variável u está associada à variável x logo os resíduos têm a seguinte forma:

- $\alpha_j = \dfrac{\langle r_j, z_j\rangle}{\langle AA^t p_j, p_j\rangle} = \dfrac{\langle r_j, z_j\rangle}{\langle A^t p_j, A^t p_j\rangle}$,
- $u_{j+1} = u_j + \alpha_j p_j \leftrightarrow x_{j+1} = x_j + \alpha_j A^t p_j$,
- $r_{j+1} = r_j - \alpha_j AA^t p_j$,
- $z_{j+1} = M^{-1} r_{j+1}$.

Usando como vetor auxiliar $w = A^t p_j$, segue a versão precondicionada à esquerda do GCNE na variável auxiliar u, isto é, o PGCNE - variável u.

Algoritmo 3.8 *Método dos Gradientes Conjugados Precondicionado na variável u - Normal Error (PGCNE)*
Calcule $r_0 = b - AA^t u_0$, $z_0 = M^{-1} r_0$, $p_0 = z_0$
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$
\begin{aligned}
w_j &= A^t p_j \\
\alpha_j &= \frac{\langle z_j, r_j\rangle}{\langle w_j, w_j\rangle} \\
u_{j+1} &= u_j + \alpha_j p_j \\
r_{j+1} &= r_j - \alpha_j A w_j \\
z_{j+1} &= M^{-1} r_{j+1} \\
\beta_j &= \frac{\langle z_{j+1}, r_{j+1}\rangle}{\langle z_j, r_j\rangle} \\
p_{j+1} &= z_{j+1} + \beta_j p_j
\end{aligned}
$$

Fim

Podemos, usando este último método, recuperar a variável x fazendo o produto $x = A^t u$ ou escrever o método diretamente na variável original x.

Algoritmo 3.9 *Método dos Gradientes Conjugados Precondicionado - Normal Error*
Calcule $r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$, $p_0 = A^t z_0$
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$
\begin{aligned}
w_j &= Ap_j \\
\alpha_j &= \frac{\langle z_j, r_j \rangle}{\langle p_j, p_j \rangle} \\
x_{j+1} &= x_j + \alpha_j p_j \\
r_{j+1} &= r_j - \alpha_j w_j \\
z_{j+1} &= M^{-1} r_{j+1} \\
\beta_j &= \frac{\langle z_{j+1}, r_{j+1} \rangle}{\langle z_j, r_j \rangle} \\
p_{j+1} &= A^t z_{j+1} + \beta_j p_j
\end{aligned}
$$

Fim

Temos assim a versão precondicionada dos dois métodos de gradientes conjugados para equações normais diretamente na variável original x.

Neste trabalho, estamos interessados em resolver problemas de programação linear utilizando os métodos de pontos interiores. Uma etapa destes métodos consiste em obter a solução de um sistema de equações normais. Assim resolveremos iterativamente este sistema utilizando os Algoritmos 3.7 e 3.8.

3.6 Considerações Finais

Apresentamos neste capítulo o método dos gradientes conjugados e algumas de suas verões. Este método iterativo será utilizado nos experimentos numéricos. As direções dy e $\tilde{dy}$ do método preditor-corretor serão determinas utilizando versões precondicionadas do método de gradientes conjugados. A versão precondicionada clássica será comparada com as versões precondicionadas que consideram a estrutura de equações normais dos sistemas que determinam as direções dy e $\tilde{dy}$. Testes computacionais foram realizados utilizando a versão clássica do método dos gradientes conjugados para determinar essas direções. Porém, essa versão não considera a estrutura de equações normais dos sistemas. Assim, realizaremos testes com as versões de gradientes conjugados que consideram a estrutura de equações normais do sistema. Esta é a nossa motivação, avaliar o método de gradientes conjugados precondicionado quando ele considera, em sua formulação, que o sistema em resolução é um sistema de equações normais.

No próximo capítulo, apresentaremos os precondicionadores que utizaremos nas versões de gradientes conjugados adotadas.

Capítulo 4

Precondicionadores para o Sistema de Equações Normais

A operação mais cara do método de pontos interiores consiste em encontrar as direções de busca resolvendo dois sistemas lineares conforme visto no Capítulo 2. Estes sistemas podem ser escritos na forma de equações normais e então resolvidos por métodos diretos como, por exemplo, a fatoração de Cholesky [6, 12, 17, 22] uma vez que são simétricos definidos positivos. Algumas vezes o uso de métodos diretos pode levar a problemas de armazenamento ou tempo de processamento, isto é, apesar de estarmos trabalhando com matrizes esparsas, quando calculamos, por exemplo, o fator de Cholesky, este geralmente será menos esparso. Essa situação pode se tornar um problema quando trabalhamos com sistemas de grande porte. Com isso, o uso de métodos iterativos pode ser a única alternativa de solução para estes casos.

O sucesso da implementação, usando métodos iterativos, depende de uma boa escolha do precondicionador. Diversos precondicionadores tem sido usados para resolver sistemas de equações normais originados do método de pontos interiores [5, 24, 29, 35]. Alguns destes precondicionadores são baseados na fatoração incompleta de Cholesky. Esta classe de precondicionadores é mais eficiente nas iterações iniciais do método de pontos interiores e perde eficiência quando o método se aproxima de uma solução. Por outro lado, próximo da convergência, o precondicionador proposto em [27], baseado na fatoração LU, tem comportamento oposto comparado à fatoração incompleta de Cholesky, isto é, sua eficiência é melhor próximo a uma solução do problema de programação linear, quando as matrizes são muito mal-condicionadas.

Nas iterações iniciais, usaremos uma classe de precondicionadores chamada de fatoração controlada de Cholesky, proposta por Campos [10] e, nas iterações finais, quando as matrizes se tornam muito mal-condicionadas, usaremos o precondicionador separador proposto por Oliveira [28]. Uma heurística irá determinar a regra para mudança de um precondicionador para outro.

4.1 Precondicionador Híbrido

Nas iterações de métodos de pontos interiores, a matriz $Z^{-1}X$ se altera muito rapidamente e, nas iterações finais, se torna extremamente mal-condicionada. Assim é difícil encontrar um precondicionador que obtenha bom desempenho para todas as iterações.

Trabalharemos com um precondicionador híbrido M, utilizando o método dos gradientes conjugados, para resolver o sistema de equações normais:

$$M^{-1}(AZ^{-1}XA^t)M^{-t}\bar{y} = M^{-1}(AZ^{-1}X(r_d - X^{-1}r_a) + r_p),$$

onde $\bar{y} = M^t dy$. Desta forma teremos duas fases no método de pontos interiores. Usaremos a fatoração controlada de Cholesky na primeira fase para construir a matriz M e o precondicionador separador para construir a matriz M na segunda fase.

4.1.1 Fatoração Controlada de Cholesky

Consideremos a fatoração de Cholesky $AZ^{-1}XA^t = LL^t = \tilde{L}\tilde{L}^t + R$ onde L é o fator obtido quando a fatoração é completa, $\tilde{L}$ quando é incompleta e R é a matriz resíduo. Definindo $E = L - \tilde{L}$ temos:

$$\tilde{L}^{-1}(AZ^{-1}XA^t)\tilde{L}^{-t} = (\tilde{L}^{-1}L)(\tilde{L}^{-1}L)^t = (I + \tilde{L}^{-1}E)(I + \tilde{L}^{-1}E)^t.$$

Podemos ver que quando a matriz $\tilde{L}$ se aproxima da matriz L, a matriz E se aproxima da matriz nula e então $\tilde{L}^{-1}(AZ^{-1}XA^t)\tilde{L}^{-t} \approx I$. A fatoração controlada de Cholesky [11] é baseada na minimização da norma de Frobenius da matriz E. Consideremos o seguinte problema:

$$\text{minimizar} \quad \|E\|_F^2 = \sum_{j=1}^{m} c_j \text{ com } c_j = \sum_{i=1}^{m} |l_{ij} - \tilde{l}_{ij}|^2.$$

Agora podemos dividir c_j em dois somatórios:

$$c_j = \sum_{k=1}^{t_j+\eta} |l_{i_k j} - \tilde{l}_{i_k j}|^2 + \sum_{k=t_j+\eta+1}^{m} |l_{i_k j}|^2,$$

onde t_j representa o número de entradas não nulas abaixo da diagonal da j-ésima coluna da matriz $AZ^{-1}XA^t$ e η representa o número de entradas extras permitidas por coluna. O primeiro somatório contém todas as $t_j + \eta$ entradas não nulas da j-ésima coluna de $\tilde{L}$. O segundo somatório contém apenas as entradas restantes do fator completo L que não têm entradas correspondentes em $\tilde{L}$. Considerando que $\tilde{l}_{ij} \approx l_{ij}$ quando $\eta \to m$ e l_{ij} não é calculado, $\|E\|_F$ é minimizado baseado em uma heurística que consiste em modificar o primeiro somatório.

Aumentando η, isto é, permitindo um maior preenchimento, c_j irá diminuir pois o primeiro somatório contém um número maior de termos e o segundo um número menor. Além disso, $\|E\|_F$ é minimizado escolhendo $t_j + \eta$ que contém as maiores entradas em valor absoluto de $\tilde{L}$ eliminando as maiores entradas correspondentes em L e, deixando apenas os menores valores de l_{ij} no segundo somatório.

O precondicionador M é construído por colunas, portanto, necessitamos apenas da j-ésima coluna de $AZ^{-1}XA^t$ a cada passo.

4.1.2 Precondicionador Separador

O precondicionador separador foi proposto para sistemas aumentados originados do método de pontos interiores [28] para problemas de programação linear. Esse precondicionador é uma generalização do proposto por Resende e Veiga [31] em um contexto de problemas de fluxo de custo mínimo em rede. A principal característica deste precondicionador é que ele trabalha bem próximo a uma solução do problema de programação linear onde este já está muito mal-condicionado e encontrar uma solução através de métodos iterativos se torna um problema muito difícil. O precondicionador separador evita ainda o armazenamento das equações normais. Seu uso é eficiente para as iterações finais do método de pontos interiores. Entretanto, quando usado nas iterações iniciais, o método não converge para diversos problemas de programação linear.

Usaremos a versão do precondicionador separador para o sistema de equações normais [27] conforme descrito a seguir.
Seja $A = [B\ N]P$ onde P é uma matriz de permutação tal que B é não singular. Fazendo $D = Z^{-1}X$ temos:

$$ADA^t = BD_BA^t + ND_NN^t.$$

Multiplicando por $D_B^{-\frac{1}{2}}B^{-1}$ e pós-multiplicando pela transposta chegamos a

$$T = D_B^{-\frac{1}{2}}B^{-1}(ADA^t)B^{-t}D_B^{-\frac{1}{2}} = I + WW^t$$

onde $W = D_B^{-\frac{1}{2}}B^{-1}ND_N^{\frac{1}{2}}$.

Observe que a matriz precondicionada é definida positiva e tem autovalores maiores ou iguais a um, isto é, não tem autovalores próximos de zero.

Próximo a uma solução do problema de programação linear pelo menos $n - m$ entradas da matriz D são pequenas. Assim, com uma escolha adequada das colunas de B, a diagonal das matrizes D_B^{-1} e D_N são muito pequenas próximas a solução. Nesta situação a matriz W se aproxima da matriz nula e, então, a matriz T se aproxima da matriz identidade.

4.1.3 Mudança de Fase

A mudança de fase é um aspecto muito importante para o bom desempenho desta abordagem. Usaremos a heurística para a mudança de fase desenvolvida em [34] que é uma melhora em relação a heurística de [9]. Isto é, se o número de iterações necessárias para a convergência do

método de gradientes conjugados é maior que $m/5$, o parâmetro η na fatoração controlada de Cholesky é incrementado em $\eta = \eta + 10$. A mudança de precondicionador acontece quando η ultrapassa um valor máximo fixado.

4.2 Métodos de Pontos Interiores e Métodos Iterativos

O passo mais caro de uma iteração de pontos interiores é a solução dos seguintes sistemas lineares:

$$(AD^{-1}A^t)\widetilde{dy} = r_p + A(D^{-1}r_d - Z^{-1}r_a) \text{ e} \tag{4.1}$$

$$(AD^{-1}A^t)dy = (r_p + A(D^{-1}r_d - Z^{-1}r_s)). \tag{4.2}$$

Para resolver os sistemas (4.1) e (4.2) vamos compará-los com os seguintes sistemas de equações normais apresentados no Capítulo 3:

- $G^tGx = G^tb$ e
- $GG^tu = b$ com $x = G^tu$.

Inicialmente, vamos comparar os sistemas (4.1) e (4.2) com o sistema de equações normais $G^tGx = G^tb$. Podemos perceber que os sistemas (4.1) e (4.2) são, também, sistemas de equações normais. Assim, podemos observar as seguintes características entre o sistema $G^tGx = G^tb$ e os sistemas (4.1) e (4.2):

- a matriz G será substituida por $D^{-\frac{1}{2}}A^t$,
- vamos manter a variável x mas esta estará indicando os valores de $\widetilde{dy}$ e dy,
- o vetor $r_p + A(D^{-1}r_d - Z^{-1}r_a)$ do sistema (4.1) é representado por G^tb no sistema $G^tGx = G^tb$ e, ainda, o vetor $r_p + A(D^{-1}r_d - Z^{-1}r_s)$ do sistema (4.2) é representado por G^tb no sistema $G^tGx = G^tb$. Portanto o lado direito dos sistemas (4.1) e (4.2) são o resultado do produto de $AD^{-\frac{1}{2}}$ pelo vetor b. Precisamos recuperar o vetor b dos sistemas (4.1) e (4.2).

Desta forma, primeiramente para o sistema (4.1) temos:

$$AD^{-\frac{1}{2}}b = r_p + A(D^{-1}r_d - Z^{-1}r_a) = b - Ax + A(D^{-1}r_d - Z^{-1}r_a).$$

Fazendo $\tilde{b} = A^t(AA^t)^{-1}b$ temos:

$$A\tilde{b} - Ax + A(D^{-1}r_d - Z^{-1}r_a) = AD^{-1}(D\tilde{b} - Dx + r_d - DZ^{-1}r_a).$$

Portanto $b = D^{-\frac{1}{2}}(D\tilde{b} - Dx + r_d - DZ^{-1}r_a)$ para o sistema (4.1).

Analogamente, o vetor b do sistema (4.2) pode ser recuperado. Assim

$$b = D^{-\frac{1}{2}}(D\tilde{b} - Dx + r_d - DZ^{-1}r_s)$$

para o sistema (4.2).

Notemos que para recuperar o vetor, operações extras são realizadas. Em particular, necessitamos de resolver outro sistema linear a cada vez que precisamos calcular o vetor b.

Portanto, com estas observações, podemos aplicar o Algoritmo 3.7 para resolver os sistemas (4.1) e (4.2).

Algoritmo 4.1 *Método dos Gradientes Conjugados Precondicionado - Normal Residual (PGCNR) - Versão para o Sistema do Método de Pontos Interiores*
Calcule $r_0 = b - D^{\frac{1}{2}}A^t x_0$, $\tilde{r}_0 = AD^{\frac{1}{2}}r_0$, $z_0 = M^{-1}\tilde{r}_0$, $p_0 = z_0$.
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$\begin{aligned}
w_j &= D^{\frac{1}{2}}A^t p_j \\
\alpha_j &= \frac{\langle z_j, \tilde{r}_j \rangle}{\langle w_j, w_j \rangle} \\
x_{j+1} &= x_j + \alpha_j p_j \\
r_{j+1} &= r_j - \alpha_j w_j \\
\tilde{r}_{j+1} &= AD^{\frac{1}{2}}r_{j+1} \\
z_{j+1} &= M^{-1}\tilde{r}_{j+1} \\
\beta_j &= \frac{\langle z_{j+1}, \tilde{r}_{j+1} \rangle}{\langle z_j, \tilde{r}_j \rangle} \\
p_{j+1} &= z_{j+1} + \beta_j p_j
\end{aligned}$$

Fim

O precondicionador que utilizaremos será o dado pela abordagem híbrida apresentado na Seção 4.1. Agora, vamos comparar os sistemas (4.1) e (4.2) com o segundo tipo de equações normais apresentado: $GG^t u = b$ com $x = G^t u$, porém iremos trabalhar com o sistema na variável u. Isto se justifica pois como queremos obter os valores dos vetores $\widetilde{dy}$ e dy nos sistemas (4.1) e (4.2), estes se comparam ao valor do vetor u no sistema $GG^t u = b$. Notemos também que a matriz G se compara à matriz $AD^{-\frac{1}{2}}$ nos sistemas (4.1) e (4.2) e, o vetor b, aos vetores $r_p + A(D^{-1}r_d - Z^{-1}r_a)$ e $r_p + A(D^{-1}r_d - Z^{-1}r_s)$.

Portanto podemos aplicar o Algoritmo 3.8 para resolver os sistemas (4.1) e (4.2) da seguinte forma:

Algoritmo 4.2 *Método dos Gradientes Conjugados Precondicionado na variável u - Normal Error (PGCNE) - Versão para o Sistema do Método de Pontos Interiores*
Calcule $r_0 = b - ADA^t u_0$, $z_0 = M^{-1}r_0$, $p_0 = z_0$
Para $j = 0, 1, \ldots,$ *até convergir, faça:*

$$
\begin{aligned}
& w_j = D^{\frac{1}{2}} A^t p_j \\
& \alpha_j = \frac{\langle r_j, z_j \rangle}{\langle w_j, w_j \rangle} \\
& u_{j+1} = u_j + \alpha_j p_j \\
& r_{j+1} = r_j - \alpha_j A D^{\frac{1}{2}} w_j \\
& z_{j+1} = M^{-1} r_{j+1} \\
& \beta_j = \frac{\langle r_{j+1}, z_{j+1} \rangle}{\langle r_j, z_j \rangle} \\
& p_{j+1} = z_{j+1} + \beta_j p_j
\end{aligned}
$$

Fim

Usaremos, também, o precondicionador apresentado na Seção 4.1.

Serão estes dois últimos algoritmos que utilizaremos no método de pontos interiores para resolver os sistemas (4.1) e (4.2).

4.3 Considerações Finais

Neste capítulo, apresentamos os precondicionadores que utilizaremos nos experimentos numéricos para o método de gradientes conjugados e, também, a heurística que adotaremos para a troca de precondicionadores. O uso de precondicionadores eficientes, muitas vezes, determina o sucesso do método iterativo. Por esta razão, utilizamos dois precondicionadores: a fatoração controlada de Cholesky e o precondicionador separador. Esses precondicionadores tem melhor eficiência em momentos diferentes do processo iterativo. A fatoração incompleta de Cholesky é mais eficiente nas iterações iniciais e, o precondicionador separador, nas iterações finais. Destacamos, ainda, a heurística adotada que determina a troca de precondicionadores. Assim, concluímos a descrição do trabalho. Resolveremos PPL, utilizando o método de pontos interiores preditor-corretor. As direções dy e $\tilde{dy}$ serão determinadas pela solução de sistemas de equações normais através de diferentes versões do método de gradientes conjugados precondicionado. Finalmente, o precondicionador adotado será o precondicionador híbrido apresentado neste capítulo.

Capítulo 5

Experimentos Numéricos

Os experimentos numéricos foram realizados utilizando o PCx [12] desenvolvido no Optimization Technology Center at Argonne National Laboratory and Northwestern University. O PCx foi implementado utilizando o método preditor-corretor com múltiplas correções e, para a resolução dos sistemas lineares que surgem do método de pontos interiores, um método direto foi adotado. As rotinas são implementadas em C, com exceção àquelas responsáveis pela solução dos sistemas lineares que foram desenvolvidas em FORTRAN77. O PCx trabalha com problemas de programação linear no formato MPS. Estes problemas podem conter restrições de igualdade e desigualdade, variávies livres ou limitadas.

5.1 Modificações no PCx

O código PCx, em seu formato original [12], resolve problemas de programação linear utilizando o método preditor-corretor com múltiplas correções. Os sistemas lineares que surguem ao decorrer do método são resolvidos através da fatoração de Cholesky. Portanto, o código PCx necessita ser modificado para o formato proposto em nosso trabalho, isto é, resolver problemas de programação linear utilizando o método preditor-corretor e, ainda, os sistemas lineares devem ser resolvidos por versões precondicionadas do método dos gradientes conjugados. Contudo, trabalhos anteriores realizaram algumas modificações [28, 27, 9, 34].

Primeiramente, as múltiplas correções foram eliminadas em [28] e o código passou a utilizar o método preditor-corretor, isto é, passou a resolver apenas dois sistemas lineares a cada iteração de pontos interiores. Ainda em [28], a fatoração de Cholesky foi substituída pelo método dos gradientes conjugados precondicionado pelo precondicionador separador. Já no trabalho [9], o precondicionador híbrido foi incorporado ao PCx utilizando uma heurística que foi modificada em [34].

Assim, o código PCx que iniciamos o trabalho já havia sido modificado, como descrevemos. Com isso, os testes realizados foram comparados aos resultados desta versão modificada do PCx. Nossas modificações no código foram a substituição da versão clássica do método dos gradientes conjugados precondicionado (3.6) pelas versões que consideram a estrutura de equações normais do sistema, isto é, os Algoritmos (3.7) e (3.8). Todos os testes foram

realizados em um processador Intel Core 2 Duo 2.2GHz com 2Gb de memória, em ambiente Linux usando os compiladores gcc e gfortran.

Utilizamos nos testes o precondicinador híbrido descrito no Capítulo 4. O número máximo de iterações para os métodos de gradientes conjugados foi dado pela dimensão do sitema (4.1).

5.2 Problemas Teste

Os problemas utilizados são de domínio público extraídos de diversas bibliotecas: NETLIB [3], STOCHPL [4], MISC [1, 2]. Os modelos QAP são da coleção QAPLIB [30]. O problema KBAPAH2 [7] pertence a uma coleção de problemas altamente mal-condicionados nos quais implementações robustas como o PCx não atingem a solução ótima. As Tabelas 5.1 e 5.2 indicam o número de linhas e de colunas dos problemas testados, o número de elementos não nulos da matriz A (NZA) e, também, a qual biblioteca pertence cada problema.

5.3 Resultados Computacionais

Para avaliar a eficiência dos métodos sugeridos comparamos o total de iterações de pontos interiores, o total de iterações do método de gradientes conjugados e o tempo total para resolver os problemas. Apresentaremos tabelas contento essas informações, onde cada tabela irá apresentar os dados obtidos em duas versões dos métodos iterativos testados. Os métodos iterativos testados consistem de duas versões de gradientes conjugados descritos no Capítulo 3, isto é, os Algoritmos (3.7) e (3.8) foram comparados ao Algoritmo (3.6). Destacamos que os Algoritmos (3.7) e (3.8) estão escritos para um sistema de equações normais qualquer. Assim, as versões implementadas no trabalho, foram escritas para os sistemas de equações normais que surgem do método de pontos interiores utilizado, isto é, implementamos as versões (4.1) e (4.2). Após apresentar os dados obtidos nos testes para cada uma das versões utilizadas, apresentaremos três gráficos contruídos a partir de uma ferramenta para medida de desempenho de algoritmos denominada *perfil de desempenho* [15].

Primeiramente, apresentaremos os resultados e as análises obtidas quando testamos a versão (4.1) e a comparamos com a versão (3.6). Nesta primeira versão testada encontramos problemas que convergiram com os mesmos parâmetros adotados para a versão (3.6), outros que convergiram após o ajuste de um parâmetro e, finalmente, problemas que não atingiram seu valor ótimo nem mesmo com estes ajustes. Quando nos referimos à ajuste de parâmetro, queremos dizer, o ajuste do precondicinador, isto é, ajuste no valor de η na fatoração controlada de Cholesky. O ajuste no precondicionador da fatoração controlada de Cholesky permitiu encontrar uma solução ótima para 31 dos 50 problemas testados. Na Tabela 5.3 apresentamos os valores máximos de η com os quais os problemas convergiram e, nas linhas indicadas por (-), não encontramos um valor de η para que o problema convergisse.

Este valor de η descrito na Tabela 5.3, é o máximo valor permitido para η. Quando chegamos a este máximo acontece a mudança de fase, ou seja, a troca de precondicionador,

Problema	Linhas	Colunas	NZA	Biblioteca
share1b	117	225	1148	NETLIB
share2b	96	79	777	NETLIB
vtpbase	198	203	283	NETLIB
sctap1	300	480	1802	NETLIB
sctap2	1090	1880	7052	NETLIB
sctap3	1480	2480	9383	NETLIB
scsd1	77	760	2388	NETLIB
scsd6	147	1350	4316	NETLIB
scsd8	397	2750	8584	NETLIB
ship04s	402	1458	2875	NETLIB
ship04l	402	2118	4290	NETLIB
ship12s	1151	2763	4297	NETLIB
ship12l	1164	5427	9254	NETLIB
seba	515	1028	4102	NETLIB
pilot4	410	1000	5001	NETLIB
pilotwe	722	2789	8841	NETLIB
pilotnov	975	2172	11934	NETLIB
scfxm2	660	914	4919	NETLIB
scfxm3	990	1371	7381	NETLIB
fit1p	627	1677	9868	NETLIB
fit2p	3000	13525	50284	NETLIB
fit1d	24	1026	13427	NETLIB
fit2d	25	10500	129042	NETLIB
25fv47	821	1571	10538	NETLIB
kbapah2	15	30	299	KBAPAH

Tabela 5.1: Dados dos Problemas Teste - Parte 1

Problema	Linhas	Colunas	NZA	Biblioteca
scsd8-2b-64	5130	35910	112770	STOCHLP
scsd8-2c-64	5130	35910	112770	STOCHLP
scsd8-2r-432	8650	60550	190210	STOCHLP
stocfor3	16675	23541	62960	NETLIB
els19	4350	13186	50882	QAP
chr22b	5587	10417	36520	QAP
scr15	2234	6210	24060	QAP
scr20	5079	15980	61780	QAP
rou20	7359	37640	152980	QAP
ste36a	27683	131076	512640	QAP
ste36b	27683	131076	512640	QAP
qap12	2794	8856	33528	NETLIB
qap15	5698	22275	85470	NETLIB
nug12	3192	8856	33528	QAP
nug15	6330	22275	85470	QAP
pds-10	15648	48780	103725	MISC
pds-20	32276	106180	226494	MISC
pds-30	49944	158489	333260	MISC
pds-40	64265	214385	457538	MISC
pds-50	80339	272513	581152	MISC
pds-60	96503	332862	709178	MISC
pds-70	114944	390005	822526	MISC
pds-80	126109	430800	916852	MISC
pds-90	142823	475448	1002902	MISC
pds-100	156243	514577	1096002	MISC

Tabela 5.2: Dados dos Problemas Teste Parte - 2

Problema	η	Problema	η
share1b	30	scsd8-2b-64	10
share2b	10	scsd8-2c-64	30
vtpbase	10	scsd8-2r-432	70
sctap1	10	stocfor3	-
sctap2	10	els19	-
sctap3	10	chr22b	-
scsd1	10	scr15	-
scsd6	10	scr20	-
scsd8	30	rou20	-
ship04s	30	ste36a	-
ship04l	30	ste36b	-
ship12s	10	qap12	-
ship12l	60	qap15	-
seba	10	nug12	-
pilot4	60	nug15	-
pilotwe	40	pds-10	10
pilotnov	10	pds-20	-
scfxm2	50	pds-30	10
scfxm3	40	pds-40	30
fit1p	10	pds-50	-
fit2p	50	pds-60	-
fit1d	10	pds-70	-
fit2d	10	pds-80	-
25fv47	40	pds-90	-
kbapah2	10	pds-100	-

Tabela 5.3: Parâmetro η para versão PGCNR

como apresentado no Capítulo 4. Portanto, quanto maior o valor máximo de η, mais tarde acontecerá a mudança de fase. Enquanto η não atinge seu valor máximo ele vai sendo incrementado e, assim, a cada momento em que ele aumenta de valor, o sistema ficará melhor precondicionado. Assim, com um melhor precondicionador, o sistema que foi resolvido necessitaria de um menor número de iterações de gradientes conjugados. Como, no método utilizado, a matriz dos coeficientes do sistema fica mais mal-condicionada a cada iteração, o aumento do η auxilia na resolução deste novo sistema.

Observamos que existiram 19 problemas que não atingem um valor ótimo para esta versão de gradientes conjugados. Assim, já podemos dizer que a versão que encontramos no código PCx modificado, PGC, tem maior eficiência quando nos referimos ao número de problemas que atingem seu valor ótimo.

Na Tabela 5.4, apresentamos apenas os problemas que a versão testada, versão PGCNR, encontrou o valor ótimo e comparamos com a versão PGC. Nesta tabela encontramos o total de iterações realizadas no método de pontos interiores, o total de iterações realizadas pelo método iterativo, isto é, o método de gradientes conjugados utilizado, e tempo total de convergência de cada problema. O total de iterações de gradientes conjugados foi dado pela soma do número de iterações realizadas para o cálculo de cada uma das duas direções do método de pontos interiores e assim foi feito para todas as iterações de pontos interiores. O total de iterações de pontos interiores estará indicado entre parênteses e o tempo é dado em segundos.

Na grande parte dos problemas da Tabela 5.4 a versão PGC se mostrou totalmente superior, isto é, convergiu em menor tempo, com menor número de iterações de pontos interiores e, também, de gradientes conjugados. Contudo, para os problemas *ship04l*, *pilotnov*, *fit1p* e *fit2p* o comportamento foi oposto, a versão PGCNR foi superior para pelo menos um dos itens analisados. Existiram também problemas em que o comportamento das duas versões foram idênticos e casos nos quais, algum dos itens analisados, em uma das versões se mostrou superior apesar de nos itens restantes tenha sido inferior.

Os resultados apresentados na Tabela 5.4 foram construídos a partir dos problemas que convergiram para a versão PGCNR. Assim, todos os outros problemas apresentados nas Tabelas 5.1 e 5.2 não atigiram seu valor ótimo para a versão PGCNR. Diversos valores para η foram testados e ainda assim não conseguimos, com a versão PGCNR, encontrar o valor ótimo para os problemas não apresentados na Tabela 5.4.

A principal motivação para testarmos estas duas versões de gradientes conjugados se baseia no fato de elas considerarem a estrutura de equações normais do sistema, aproveitando assim sua estrutura natural, no caso, os sistemas apresentados para o cálculo das direções do método de pontos interiores. Porém, para a versão PGCNR, necessitamos de um esforço extra, isto é, recuperar o lado direito do sistema como descrito na Seção 4.2. Ao longo do método o lado direito deste sistema é calculado e quando necessitamos determinar a solução desse sistema, usando o método PGCNR, precisamos recuperar o lado direito. Portanto, fazendo essas operações, acumulamos muitos erros númericos, visto que temos até mesmo o cálculo de um sistema linear extra para determinar uma matriz inversa. Por isso, talvez, a versão PGCNR perca a eficiência no ponto de vista do total de problemas em que atingiu

Problema	Iterações do PGC	Iterações do PGCNR	Tempo - PGC	Tempo - PGCNR
share1b	45 (19)	273 (19)	0,04	0,06
share2b	142 (17)	143 (17)	0,02	0,02
vtpbase	214 (11)	214 (11)	0,02	0,02
sctap1	362 (16)	585 (21)	0,08	0,11
sctap2	93 (14)	188 (20)	0,16	0,27
sctap3	191 (16)	607 (24)	0,31	0,64
scsd1	146 (9)	163 (12)	0,03	0,03
scsd6	263 (13)	263 (13)	0,05	0,06
scsd8	272 (12)	272 (12)	0,13	0,17
ship04s	495 (15)	544 (18)	0,15	0,23
ship04l	609 (19)	630 (18)	0,36	0,32
ship12s	885 (14)	907 (13)	0,31	0,33
ship12l	1133 (17)	1194 (17)	1,01	1,23
seba	5091 (16)	6162 (18)	2,48	5,28
pilot4	1022 (48)	2084 (48)	0,97	1,42
pilotwe	2419 (48)	2429 (49)	1,58	1,63
pilotnov	3675 (18)	3673 (18)	2,13	2,40
scfxm2	884 (20)	868 (21)	0,49	0,52
scfxm3	1348 (20)	1376 (22)	0,90	0,99
fit1p	37074 (43)	36242 (40)	56,46	49,98
fit2p	171213 (48)	170759 (47)	2558,26	2863,63
fit1d	58 (19)	58 (19)	0,08	0,10
fit2d	56 (25)	56 (25)	0,98	1,03
25fv47	4185 (26)	4630 (28)	4,19	5,02
kbapah2	18 (8)	18 (8)	0,00	0,00
scsd8-2b-64	386 (7)	878 (8)	3,39	6,06
scsd8-2c-64	382 (7)	5864 (10)	4,04	20,00
scsd8-2r-432	6519 (18)	11397 (22)	39,28	180,96
pds-10	6872 (47)	8851 (49)	63,36	77,26
pds-30	31129 (73)	70946 (79)	741,93	1748,39
pds-40	43516 (79)	127153 (84)	1463,64	4813,55

Tabela 5.4: Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NR (PGCNR) - Tempo

um valor ótimo. Portanto, apesar da versão PGCNR tirar proveito da estrutura de equações normais, esta versão fica muito comprometida pois necessita recuperar um vetor para a sua utilização.

A versão PGCNE já não necessita deste esforço extra. Esta versão aproveita a estrutura de equações normais do sistema sem a necessidade de recuperar nenhum vetor. Isto acontece pois, na versão PGCNE, o método iterativo é construído considerando que o vetor de incógnitas do sistema linear é substituido pelo produto da matriz transposta do sistema por um vetor auxiliar, como descrito na Seção 4.2. Assim o sistema de equações normais construído não alterou o lado direito do sistema original.

Os problemas testados com a versão PGCNE atingiram um valor ótimo para praticamente todos os problemas. Nesta versão utilizamos o precondicionador híbrido, como em todas as outras. Porém, os parâmetros adotados para o precondicionador, utilizando a versão PGCNE, foram os mesmos utilizados para a versão modificada do PCx que trabalha com a versão PGC para a solução dos sistemas lineares. Logo, não foi necessário fazer ajustes no precondicionador, isto é, modificações no paramêtro η para que os problemas convergissem. Isto mostra uma maior competitividade da versão PGCNE com a versão PGC.

Visto que praticamente todos os problemas apresentados nas Tabelas 5.1 e 5.2 atingem um valor ótimo, selecionamos os problemas de maior porte para apresentar os resultados e destacamos que nos problemas omitidos os dois métodos se comportam praticamente da mesma forma quando comparamos as iterações de pontos interiores, gradientes conjugados e o tempo total. A Tabela 5.5 mostra os resultados obtidos para os 22 problemas selecionados. Na Tabela 5.5 exibimos o total de iterações de pontos interiores entre parênteses, o total de iterações de gradientes conjugados e o tempo total medido em segundos.

Observamos que os métodos comparados se comportam de forma muito parecida em parte dos problemas testados. Apesar disso ocorrem diferenças consideráveis entre os métodos. Variações no tempo e no total de iterações de gradientes conjugados ocorreram em todos os problemas e, em alguns casos, até mesmo no total de iterações de pontos interiores. Destacamos os problemas *ste36b* e *nug15*. O primeiro atingiu a solução ótima apenas quando foi usado o método de gradientes conjugados precondicionado que considera a estrutura de equações normais, isto é, o PGCNE. Entretanto, o segundo problema atingiu uma solução ótima apenas quando o método de gradientes conjugados precondicionado não considera a estrutura de equações normais do sistema, isto é, o PGC.

Finalmente, vamos observar os resultados obtidos quando comparamos as duas versões testadas, PGCNE e PGCNR.

A Tabela 5.6 contém o total de iterações de pontos interiores, gradientes conjugados e o tempo total. Para estes testes, também selecionamos somente os problemas em que a versão PGCNR encontrou o valor ótimo. Com isso, a versão PGCNE já apresenta superioridade, visto que encontrou a solução ótima para a maior parte dos problemas apresentados nas Tabelas 5.1 e 5.2. É importante lembrar que os problemas que não constam na Tabela 5.6 são problemas em que a dimensão é grande, isto é, problemas de grande porte. O que reforça ainda mais o melhor desempenho da versão PGCNE quando comparada com a versão PGCNR.

Problema	Iterações do PGC	Iterações do PGCNE	Tempo - PGC	Tempo - PGCNE
stocfor3	69507 (32)	69828 (32)	339,93	345,72
els19	26294 (31)	26315 (31)	167,90	172,48
chr22b	25580 (29)	25688 (29)	71,28	72,98
scr15	15973 (24)	15886 (24)	28,77	28,99
scr20	36360 (21)	35789 (21)	224,31	254,20
rou20	54326 (24)	54606 (24)	3165,60	3156,59
ste36a	312988 (38)	305066 (38)	32303,34	32256,71
ste36b	-	375810 (37)	-	41786,72
qap12	29962 (20)	30519 (20)	290,59	305,59
qap15	113695 (24)	92876 (23)	4870,09	4160,35
nug12	23677 (20)	24111 (20)	245,41	249,23
nug15	91134 (23)	-	4447,09	-
pds-10	6872 (47)	6765 (47)	62,63	63,01
pds-20	52372 (60)	53759 (60)	745,72	782,26
pds-30	31769 (74)	31324 (74)	791,48	824,34
pds-40	43516 (79)	43797 (79)	1462,24	1564,04
pds-50	71481 (79)	69471 (79)	2894,95	2959,64
pds-60	83095 (85)	82774 (85)	4116,21	4337,98
pds-70	83050 (84)	77579 (85)	4978,68	4994,45
pds-80	80742 (83)	75626 (83)	5743,66	5658,51
pds-90	97909 (82)	82992 (81)	7521,79	6841,65
pds-100	122237 (86)	126769 (87)	9708,21	10493,45

(-) significa que o método falhou.

Tabela 5.5: Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NE (PGCNE) - Tempo

Problema	Ite. PGCNE	Ite. PGCNR	Tempo - PGCNE	Tempo - PGCNR
share1b	45 (19)	273 (19)	0,03	0,06
share2b	142 (17)	143 (17)	0,02	0,02
vtpbase	215 (11)	214 (11)	0,02	0,02
sctap1	363 (16)	585 (21)	0,07	0,11
sctap2	93 (14)	188 (20)	0,16	0,27
sctap3	191 (16)	607 (24)	0,32	0,64
scsd1	146 (9)	163 (12)	0,02	0,03
scsd6	263 (13)	263 (13)	0,06	0,06
scsd8	271 (12)	272 (12)	0,13	0,17
ship04s	608 (16)	544 (18)	0,20	0,23
ship04l	602 (19)	630 (18)	0,36	0,32
ship12s	884 (13)	907 (13)	0,27	0,33
ship12l	1066 (17)	1194 (17)	0,96	1,23
seba	5875 (15)	6162 (18)	3,28	5,28
pilot4	1022 (48)	2084 (48)	0,99	1,42
pilotwe	2422 (48)	2429 (49)	1,58	1,63
pilotnov	3655 (18)	3673 (18)	2,04	2,40
scfxm2	874 (20)	868 (21)	0,49	0,52
scfxm3	1143 (20)	1376 (22)	0,93	0,99
fit1p	37188 (44)	36242 (40)	60,33	49,98
fit2p	174704 (48)	170759 (47)	3178,70	2863,63
fit1d	58 (19)	58 (19)	0,08	0,10
fit2d	56 (25)	56 (25)	0,89	1,03
25fv47	4176 (26)	4630 (28)	4,30	5,02
kbapah2	18 (8)	18 (8)	0,00	0,00
scsd8-2b-64	395 (7)	878 (8)	3,76	6,06
scsd8-2c-64	389 (7)	5864 (10)	4,03	20,00
scsd8-2r-432	6914 (18)	11397 (22)	42,26	180,96
pds-10	7041 (47)	8851 (49)	62,22	77,26
pds-30	32017 (73)	70946 (79)	785,43	1748,39
pds-40	43509 (79)	127153 (84)	1537,23	4813,55

Tabela 5.6: Total de Iterações de Pontos Interiores e Gradientes Conjugados Precondicionado - versão NE x NR - Tempo

Analisando a Tabela 5.6 encontramos diferenças relevantes entre as versões. Primeiramente, destacamos a variação no total de iterações de pontos interiores. Considerando somente os problemas em que as duas versões encontraram uma solução ótima, apenas nos problemas *ship04l*, *fit1p* e *fit2p* a versão PGCNR obteve melhor desempenho. Ainda considerando o total de iterações de pontos interiores, a versão PGCNE se mostrou superior em 16 dos problemas apresentados, sendo que, em grande parte destes problemas, o total de iterações de pontos interiores foi consideravelmente menor para a versão PGCNE. Destacamos, por exemplo, o problema *sctap3* que atingiu um valor ótimo em 8 iterações de pontos interiores a menos para a versão PGCNE. Apesar destas observações, encontramos o problema *fit1p* que chegou ao valor ótimo com 4 iterações a menos na versão PGCNR.

Notamos que alguns problemas se comportaram de maneira idêntica quando observamos apenas o total de iterações de pontos interiores. Para a maioria destes casos, o total de iterações de gradientes conjugados foi menor para a versão PGCNE.

O tempo total foi melhor para a versão PGCNE na maioria dos problemas. Ainda assim, destacamos os problemas *ship04l* ,*fit1p* e *fit2p*, pois convergiram com melhor tempo para versão PGCNR. Observe que o total de iterações de pontos interiores, para estes 3 problemas, é menor para a versão PGCNR, fato que pode justificar o melhor desempenho no tempo total para estes problemas.

Apresenteremos agora uma ferramenta para a análise do desempenho de algoritmos, o *perfil de desempenho* [15].

5.3.1 Perfil de Desempenho

O perfil de desempenho, introduzido por Dolan e Moré [15], é uma ferramenta para comparar o desempenho de n_s algoritmos quando desejamos que cada um desses algoritmos resolva um conjunto com n_p problemas. No caso deste trabalho, teremos $n_s = 3$ e $n_p = 50$, isto é, vamos comparar o desempenho de 3 algoritmos (PGC, PGCNR, PGCNE) na resolução dos 50 problemas apresentados nas Tabelas 5.1 e 5.2. Podemos ter diferentes medidas de desempenho, por exemplo, tempo de processamento ou números de avaliações de função. Como analisado anteriormente, iremos utilizar como medida de desempenho:

- o total de iterações de pontos interiores,
- o total de iterações de gradientes conjugados e,
- o tempo total para resolução de cada problema.

O perfil de desempenho produzirá gráficos comparando os métodos para cada medida de desempenho adotada. Vejamos, agora, como iremos produzir esses gráficos.

A medida de desempenho para resolver o problema p através do método s é denotada por m_{sp}. Assim, sendo S o conjunto de métodos e P o conjunto de problemas, teremos para cada problema p e método s a taxa de desempenho r_{sp} calculada da seguinte forma:

$$r_{sp} = \frac{m_{sp}}{min\ \{m_{sp} \mid s \in S\}},$$

se o método s tem sucesso na resolução do problema p, caso contrário,

$$r_{sp} = r_M,$$

onde r_M é um parâmetro pré-fixado e suficientemente grande.

Construímos agora, para cada s, a função de distribuição acumulativa ρ_s, para a taxa de desempenho r_{sp}, dada por:

$$\begin{aligned} &\rho_s : \Re \to [0,1] \\ &\rho_s(\tau) = \frac{1}{n_p} \left| p \in P \mid r_s \leq \tau \right|, \end{aligned}$$

onde $|A|$ é a cardinalidade do conjunto A.

A função ρ_s é não decrescente e constante por partes e representa o desempenho do método s na resolução de p problemas. Esta função nos fornece duas informações importantes:

- $\rho_s(1)$ representa a porcentagem de problemas resolvidos em menor tempo pelo método s, ou seja, a eficiência do método s e,
- a robustez do método s, representada pelo valor de $\tilde{\tau}$ tal que $\rho_s(\tilde{\tau}) = 1$, se existe tal valor para $\tau \leq r_M$.

Assim, o método que fornece o máximo valor para $\rho_s(1)$, resolve o maior número de problemas com o menor valor para a medida m e, então, é considerado o mais eficiente. O método $\hat{s}$ será o mais robusto quando $\tilde{\tau}_{\hat{s}} = min_{s \in S}\ \tau_s$.

Com isso, temos mais uma ferramenta para a análise do desempenho dos métodos testados. Assim, apresentaremos os gráficos gerados pelo perfil de desempenho para cada medida de desempenho adotada. As medidas de desempenho adotas foram: o total de iterações de pontos interiores, o total de iterações de gradientes conjugados e o tempo total de resolução dos problemas e, os gráficos gerados, serão apresentados nesta ordem. Diferente dos dados apresentados nas Tabelas 5.4, 5.5 e 5.6 que não apresentam todos os 50 problemas selecionados para os testes, os gráficos do perfil de desempenho foram construídos com os resultados de todos os problemas apresentados nas Tabelas 5.1 e 5.2.

A Figura 5.1 apresenta o gráfico do perfil de desempenho dos métodos PGC, PGCNR e PGCNE, utilizando como medida de desempenho o total de iterações de pontos interiores. Observando o gráfico percebemos que o Algoritmo PGCNR é o menos eficiente e robusto. Já os Algoritmos PGC e PGCNE, tem a mesma eficiência e, a robustez, apresenta uma diferença muito pequena.

A Figura 5.2 apresenta o gráfico do perfil de desempenho dos métodos PGC, PGCNR e PGCNE, utilizando como medida de desempenho o total de iterações de gradientes conjugados. Neste gráfico, observamos praticamente as mesmas características da Figura 5.1 porém, a pequena diferença que existia na robustez dos Algoritmos PGC e PGCNE, já não existe no gráfico da Figura 5.2.

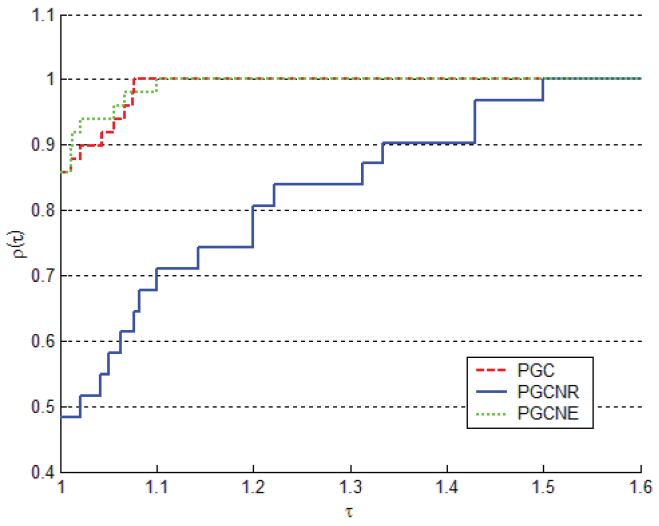

Figura 5.1: Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Iterações de pontos interiores

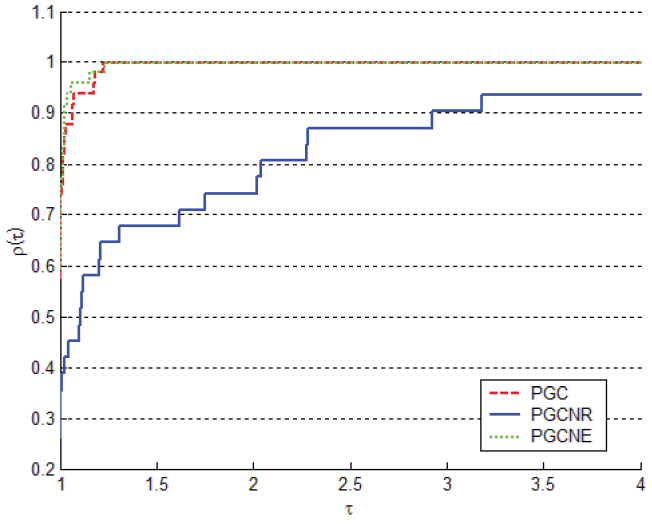

Figura 5.2: Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Iterações de gradientes conjugados

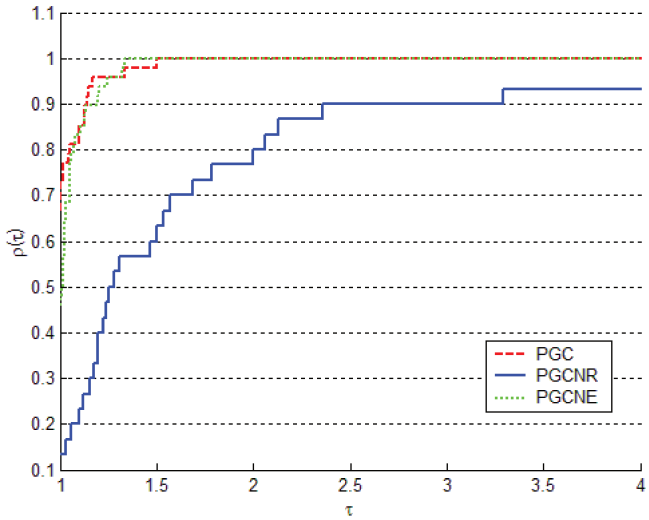

Figura 5.3: Perfil de desempenho para PGC, PGCNR e PGCNE - Medida de desempenho: Tempo de resolução

A Figura 5.3 apresenta o gráfico do perfil de desempenho dos métodos PGC, PGCNR e PGCNE, utilizando como medida de desempenho o tempo total de resolução dos problemas. Neste gráfico o método PGCNR ainda é o menos eficiente e menos robusto contudo, o método PGCNE é o mais robusto e o método PGC o mais eficiente.

Utilizando o perfil de desempenho observamos que os métodos PGC e PGCNE tem desempenho muito semelhante entre si e são superiores quando comparados com o método PGCNR.

5.4 Considerações Finais

Existem grandes diferenças nos resultados apresentados para as versões testadas. Ocorreram variações em todos os itens avaliados, ou seja, até mesmo no total de iterações nos métodos de pontos interiores. Isto é importante de destacar pois a cada iteração de pontos interiores resolvemos dois sistemas lineares e encontrar a solução destes sistemas é a etapa que exige um grande esforço computacional. Portanto sempre buscamos reduzir o total de iterações nos métodos de pontos interiores, sobre esse ponto de vista, a versão PGCNE é a opção indicada.

Como dissemos anteriormente, a motivação para testar essas duas versões de gradientes conjugados baseava-se em aproveitar a estrutura de equações normais do sistema de equações lineares. A versão PGCNE aproveita essa estrutura sem custos adicionais e obteve ótimos resuldados quando comparada com a versão encontrada no PCx modificado, isto é, a versão PGC. Porém, a versão PGCNR, apesar de também aproveitar a estrutura de equações normais do sistema, não obteve resultados tão satisfatórios. Isto ocorreu porque

esta versão necessitou recuperar o lado direito do sistema. Com isso acumulamos muitos erros numéricos e isto custou um pior desempenho para muitos problemas e, para os problemas de maior porte, não conseguimos obter a solução ótima.

Selecionamos os problemas da coleção KBAPAH para teste porque eles são altamente mal-condicionados. Porém para todas as versões de gradientes conjugados apresentadas neste trabalho, apenas para o problema *KBAPAH2* encontramos a solução ótima. Por essa razão somente este problema foi apresentado nos resultados. Nos demais problemas desta coleção não conseguimos encontrar a solução ótima com nenhuma das versões, nem mesmo com a versão PGC. Portanto para esta coleção de problemas o PCx modificado ainda não apresenta um bom desempenho.

Enfim, para os problemas testados, a versão PGCNE se mostrou extremamente competitiva com a versão PGC encontrada no PCx modificado.

Capítulo 6

Conclusões e Trabalhos Futuros

Os métodos de pontos interiores são uma excelente opção quando trabalhamos com problemas de programação linear de grande porte. O ponto crítico destes métodos consiste na solução de dois sistemas de equações lineares que surguem a cada iteração. A matriz dos coeficientes deste sistema muda a cada iteração dos métodos de pontos interiores, o que significa que a cada iteração temos uma nova matriz para trabalhar. Assim, estratégias eficientes devem ser adotadas para a solução destes sistemas que consiste no passo mais caro do método. Restrições de tempo e memória impedem, muitas vezes, o uso de métodos diretos. Portanto, o uso de métodos iterativos pode ser a única opção em alguns casos. Contudo, o sucesso do método iterativo adotado, depende de um bom precondicionador.

Neste trabalho, apresentamos resultados no contexto descrito. Necessitavamos resolver problemas de programação linear e, para isso, utilizamos um método de pontos interiores no qual as soluções dos sistemas que surgem foram obtidas utilizando um método iterativo precondicionado. Avaliamos a eficiência de diferentes versões do método de gradientes conjugados precondicionado utilizando um precondicionador híbrido.

A principal motivação para testar diferentes versões de gradientes conjugados precondicionado vem da estrutura natural que os sistemas obtidos no método preditor-corretor aparecem, isto é, sistemas de equações normais. Motivados por esse fato, o uso de versões do método iterativo que considera essa estrutura em sua formulação foram avaliados. Estudos anteriores já haviam sido realizados com uma versão de gradientes conjugados precondicionado mas, neste caso, a versão não considerava a estrutura de equações normais dos sistemas envolvidos.

Para avaliar a eficiência de cada uma das versões consideramos o total de iterações dos métodos de pontos interiores, de gradientes conjugados e o tempo de processamento total. Inicialmente, testamos os problemas selecionados para a versão de gradientes conjugados que foi nossa base para comparação, PGC. O próximo passo foi implementar e testar as outras versões, PGCNE e PGCNR.

A implementação da versão PGCNE no código PCx ocorreu naturalmente visto que a estrutura do método se adequava muito bem aos sistemas encontrados. Os resultados obtidos com essa versão foram muito semelhantes aos da versão PGC. Em poucos casos ocorreram

variações no total de iterações de pontos interiores. De maneira geral, o tempo e total de iterações do método dos gradientes conjugados também não apresentaram grandes diferenças. Ainda para essas versões, PGC e PGCNE, não foi necessário fazer ajustes nos parâmetros do precondicionador, visto que, usando ambas as versões, o método de pontos interiores encontra uma solução ótima para a quase totalidade dos problemas testados. Portanto, a versão PGCNE obteve bom desempenho e se mostrou competitiva com a versão PGC.

A versão PGCNR, já apresentou dificuldades teóricas antes da implementação. Foi necessário desenvolver adequações para serem incorporadas aos métodos de pontos interiores. Isto ocorreu pois a versão PGCNR considera que o sistema de equações normais é obtido a partir de um sistema original que foi premultiplicado pela matriz transposta dos coeficientes do sistema. Assim, o lado direito do sistema também será premultiplicado por esta mesma matriz. Este foi o ponto que necessitou de maiores ajustes para a implementação, uma vez que, o sistema originado pelo método de pontos interiores, já se encontrava nesta forma. Portanto, foi necessário modificar o lado direito destes sistemas, isto é, recuperar um vetor que havia sido premultiplicado por uma matriz. Essas adaptações envolvem muitas operações que incluem até mesmo a resolução de um sistema adicional. Com isso, acumulamos erros númericos e assim, o método ficou prejudicado perdendo eficiência. Esta perda de eficiência foi confirmada nos testes realizados. Diversos problemas testados não atingiram uma solução ótima sendo a maioria problemas de grande porte. Mesmo os problemas que convergiram para uma solução ótima, muitos deles necessitaram que o precondicionador fosse ajustado. Assim, observamos muitas variações no total de iterações dos métodos de pontos interiores e de gradientes conjugados e, também, no tempo total de convergência.

O fato de não encontramos uma solução ótima para os problemas de maior dimensão nos mostrou uma pior eficiência do PGCNR em relação ao PGC e ao PGCNE. Ainda assim, em alguns problemas, o PGCNR obteve um bom desempenho. Portanto, o seu uso pode ser mais adequado em outras aplicações, por exemplo, aplicações onde não seja necessário recuperar o lado direito do sistema como no problema aqui tratado.

Neste trabalho, testamos métodos iterativos que trabalham com uma das duas formulações algébricas mais comuns dos sistemas obtidos do método de pontos interiores, isto é, sistemas de equações normais onde a matriz dos coeficientes é simétrica e definida positiva. Em trabalhos futuros, podemos avaliar métodos iterativos que trabalhem com a outra formulação comum destes sistemas, o sistema aumentado e, neste caso, a matriz dos coeficientes será simétrica e indefinida.

Referências Bibliográficas

[1] Miscellaneous lp models. Technical report, Hungarian Academy of Sciences OR Lab. Online at http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc.

[2] Mittelmann - lp models. Technical report, Miscellaneous LP models collect by Hans D. Mittelmann. Online at ftp://plato.asu.edu/pub/lptestset/pds.

[3] Netlib collection lp test sets. Technical report, NETLIB LP repository. Online at http://www.netlib.org/lp/data.

[4] Stochastic lp test sets. Technical report, Hungarian Academy of Sciences OR Lab. Online at http://www.sztaki.hu/~meszaros/public_ftp/lptestset/stochlp.

[5] Ilan Adler, Narendra Karmarkar, Mauricio G. C. Resende, and Geraldo Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA Journal on Computing*, 1:84–106, 1989.

[6] Ilan Adler, Mauricio G. C. Resende, Geraldo Veiga, and Narendra Karmarkar. An implementation of Karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[7] M. D. Ašić and V. V. Kovačević-Vujčić. Ill-conditionedness and interior-point methods. *UNIV. BEOGRAD. PUBL ELEKTROTEHN. FAK. SER. MAT.*, 11:53–58, 2000.

[8] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[9] S. Bocanegra, F. F. Campos, and A. R. L. Oliveira. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, 36(1–2):149–164, 2007. Special issue on "Linear Algebra Issues arising in Interior Point Methods".

[10] F. F. Campos. *Analysis of Conjugate Gradients - type methods for solving linear equations*. PhD thesis, Oxford University Computing Laboratory, Oxford, 1995.

[11] F. F. Campos and N. R. C. Birkett. An efficient solver for multi-right hand side linear systems based on the CCCG(η) method with applications to implicit time-dependent partial differential equations. *SIAM J. Sci. Comput.*, 19(1):126–138, 1998.

[12] Joseph Czyzyk, S. Mehrotra, Michael Wagner, and Stephen J. Wright. PCx an interior point code for linear programming. *Optimization Methods & Software*, 11-2(1-4):397–430, 1999.

[13] G. B. Dantzig. *Maximization of a Linear Function of Variables subject to Linear Inequalities*, pages 339–347. Activity Analysis of Production and Allocation, (T. C. Koopmans ed.), Wiley, New York, 1951.

[14] John E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, PA, 1996.

[15] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[16] Gene H. Golub and Charles F. Van Loan. *Matrix Computations 2nd Edition*. The Johns Hopkins University Press, Baltimore, Maryland, 1989.

[17] J. Gondzio. HOPDM (Version 2.12) – A fast LP solver based on a primal–dual interior point method. *European Journal of Operational Research*, 85:221–225, 1995.

[18] A. Greenbaum. *Iterative Methos for Solving Linear Systems*. SIAM Publications, SIAM, Philadelphia, PA, USA, 1997.

[19] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[20] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point method for linear programming. *Progress in Mathematical Programming, Interior-Point and Related Methods, Spring-Verlag, New York*, pages 29–47, 1989.

[21] David G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, 1984.

[22] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization*, 2:435–449, 1992.

[23] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6:1–22, 1994.

[24] S. Mehrotra. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. *ORSA Journal on Computing*, 4:103–118, 1992.

[25] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[26] Renato D. C. Monteiro, Ilan Adler, and Mauricio G. C. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15:191–214, 1990.

[27] Aurelio R. L. Oliveira and D. C. Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra and Its Applications*, 394:1–24, 2005.

[28] Aurelio Ribeiro Leite Oliveira. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Technical report, PhD Thesis, TR97-11, Department of Computational and Applied Mathematics, Rice University, Houston TX, 1997.

[29] L. F. Portugal, M.G. C. Resende, G. Veiga, and J. J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. *Networks*, 35:91–108, 2000.

[30] S. Karisch R. S. Burkard and F. Rendl. Qaplib - a quadratic assignment problem library. *Europen Journal of Operations Research*, 55:115–119, 1991.

[31] Mauricio G. C. Resende and Geraldo Veiga. An efficient implementation of a network interior point method. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 12:299–348, 1993.

[32] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM Publications, SIAM, Philadelphia, PA, USA, 2003.

[33] R. J. Vanderbei. *Linear Programming – Foundations and Extensions*. Kluwer Academics Publishers, Boston, USA, 1996.

[34] M. I. Velazco, A. R. L. Oliveira, and F. F. Campos. A note on hybrid preconditions for large scale normal equations arising from interior-point methods. *Optimization Methods and Software*, 25:321–332, 2010.

[35] W. Wang and D. P. O'Leary. Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming. *Numerical Algorithms*, 25(1–4):387–406, 2000.

[36] S. J. Wright. *Primal–Dual Interior–Point Methods*. SIAM Publications, SIAM, Philadelphia, PA, USA, 1997.