Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Algoritmos de Newton-Krylov precondicionados para métodos de pontos interiores

Silvana Bocanegra

Belo Horizonte

Dezembro de 2005

Silvana Bocanegra

Algoritmos de Newton-Krylov precondicionados para métodos de pontos interiores

Orientador: Frederico Ferreira Campos, filho

Tese de doutorado apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Belo Horizonte

Dezembro de 2005

À
minha querida mãe, Romilda,
meu pai, Lamerch (*in memorian*).

Agradecimentos

Agradeço imensamente ao Prof. Frederico por sua inestimável orientação, tanto científica quanto pessoal, e acima de tudo por sua amizade.

Ao Prof. Marcos, por me transmitir sólidos conhecimentos na área de pontos interiores, e pelo apoio nos momentos difíceis encontrados durante esse período.

Ao Prof. Aurélio por sua valiosa contribuição, fornecendo o código e me orientando no desenvolvimento do trabalho.

À minha mãe, principal responsável por essa vitória, pela sua imensa luta para me proporcionar uma formação digna e pelo carinho e amor incondicionais em todos os momentos de minha vida.

Aos meus irmãos Sandra e Sandro, por serem sempre o meu porto seguro e por colocarem em minha vida os sobrinhos maravilhosos Marcela, Willian e Isabela, que tanto amo.

À minha madrinha Cizinha pelas orações a mim concedidas e aos agregados da família Silmara, Osmar e Diego pelos momentos divertidos em Araraquara.

A todos os colegas do curso de doutorado, em especial aos amigos Júlio, Umberto, Lula e Kissia que muito me ajudaram nas disciplinas da qualificação e às amigas Dri, Joyce e Karla pelo apoio e incentivo nestes últimos meses.

Ao amigo David pela sua grande ajuda na linguagem C e ao amigo Paulo pela companhia em muitas noites de laboratório.

Às amigas Dri, Lena, Kissia, Dani, Fernanda, Bianca, Mariana, Duda, Aninha e Claúdia por serem ótimas companheiras nos momentos de descontração.

Às amigas de apartamento, Nazaré, Simone, Dani, Kissia, Fernanda e Lena pela amizade e pelos momentos felizes que compartilhamos durante esses anos.

A todos os professores, que contribuíram com minha formação, em especial aos professores Frederico, Marcos e Christiano por compartilharem suas experiências como docentes. Aos funcionários do DCC, por serem sempre prestativos.

À CAPES e FAPEMIG, pelo imprescindível apoio financeiro.

A Deus, por tudo.

Resumo

Métodos de pontos interiores têm sido amplamente usados na solução de problemas de programação linear de grande porte. A cada iteração destes métodos é necessário resolver um sistema de equações lineares para calcular a direção de Newton. Esta é a tarefa que demanda a maior parte do tempo de processamento e deve ser feita de forma eficiente. A abordagem mais utilizada em métodos de pontos interiores implementa a fatoração de Cholesky. No entanto, esta fatoração pode ser densa em algumas classes de problemas. Nestes casos, o uso de métodos iterativos torna-se mais interessante, desde que sejam utilizados com precondicionadores eficientes. Devido à dificuldade de encontrar estratégias de precondicionamento que apresentam bons resultados durante todas as iterações de pontos interiores estamos propondo uma abordagem iterativa híbrida para resolver estes sistemas. O método do gradiente conjugado é precondicionado nas iterações iniciais (fase I) usando um tipo de fatoração incompleta de Cholesky na qual o preenchimento pode ser controlado em termos da memória disponível e nas últimas iterações (fase II) usando um precondicionador baseado na fatoração LU que apresenta melhores resultados nas proximidades da solução ótima. A troca de fases ocorre quando a fatoração controlada de Cholesky começa a perder eficiência tornando lenta a convergência pelo método do gradiente conjugado. Experimentos numéricos revelam que a abordagem híbrida apresenta desempenho superior quando comparada a abordagem direta na solução de algumas classes de problemas de programação linear de grande porte.

Abstract

Interior point methods have been widely used to solve large-scale linear programming problems. The bulk of the work in these methods is computing the search direction by solving one or more linear systems. The most commom approach in interior point solvers uses Cholesky sparse factorization to solve these systems. In some problems this factorization becomes prohibitive due to storage and time limitations. Iterative approaches are more interesting in these situations. Since these systems are ill-conditioned, it is crucial to develop efficient preconditioners. However it is difficult to find a preconditioning strategy that produces good performance of iterative methods over the entire course of the interior point iterations. We are proposing a hybrid approach to solve these systems. The preconditioned conjugate gradient method works in two phases. During phase I it uses a kind of incomplete Cholesky preconditioner such that fill-in can be controlled in terms of available memory. As the optimal solution of the problem is approached, the linear systems become highly ill-conditioned and the method changes to phase II. In this phase a preconditioner based on the LU factorization is found to work better near a solution of the LP problem. The numerical experiments reveal that the iterative hybrid approach works better than Cholesky factorization on some classes of large-scale problems.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Nos últimos anos, os métodos de pontos interiores, desenvolvidos a partir do trabalho de Karmarkar [55], têm atraído o interesse da comunidade científica. Além da contribuição teórica destes métodos, na prática, eles têm se mostrado eficientes, desde que convenientemente implementados [25, 45, 67]. Extensivos testes numéricos mostram que implementações robustas resolvem muitos problemas de grande porte substancialmente mais rápido que o melhor código do método simplex [63].

Cada iteração de um método de pontos interiores envolve a solução de sistemas lineares, conhecidos por sistemas de equações de Newton. Em aplicações reais esses sistemas quase sempre possuem dimensões elevadas e alto grau de esparsidade. Tipicamente, são realizadas algumas operações algébricas reduzindo os sistemas a duas formulações mais simples. Uma delas, conhecida por sistemas aumentados, envolve matrizes simétricas, indefinidas e geralmente esparsas. À outra, denominada sistemas de equações normais, estão associadas matrizes que possuem menor dimensão, são simétricas e definidas positivas.

Resolver os sistemas lineares é a fase que demanda a maior parte do tempo de processamento. Desta forma, a escolha do método de solução é de extrema importância para que se tenha uma implementação eficiente. Normalmente aplicam-se métodos diretos para resolver os sistemas de equações de Newton. A abordagem mais utilizada nos códigos de pontos interiores implementa a fatoração de Cholesky das matrizes simétricas e definidas positivas [7, 45, 62]. Muitas vezes, por limitações de tempo e memória seu uso torna-se proibitivo, fazendo com que abordagens iterativas usando métodos do subespaço de Krylov sejam mais

interessantes. O sucesso de implementações que utilizam esta estratégia depende do uso de bons precondicionadores, já que a matriz dos coeficientes torna-se muito malcondicionada nas proximidades de uma solução ótima.

Em muitas dessas implementações, são construídos precondicionadores baseados em fatorações incompletas da matriz definida positiva [6, 56, 66]. Tipicamente, essa classe de precondicionadores é eficiente nas primeiras iterações de pontos interiores e se deteriora a medida que a solução ótima do problema se aproxima. No entanto, foi proposto um precondicionador para os sistemas indefinidos baseado na fatoração LU [73], que trabalha melhor nas proximidades da solução ótima do problema, quando os sistemas já estão muito malcondicionados.

O objetivo desse trabalho é o desenvolvimento de uma abordagem iterativa híbrida para resolver os sistemas de equações normais originados na aplicação de métodos de pontos interiores em problemas de programação linear. O método do gradiente conjugado é precondicionado nas iterações iniciais (fase I) usando um tipo de fatoração incompleta, denominada Fatoração Controlada de Cholesky (FCC), proposta por Campos [19] e nas últimas iterações (fase II) usando o precondicionador separador proposto por Oliveira [73]. A FCC ainda não tinha sido usada em problemas de otimização, porém apresenta propriedades interessantes neste contexto. A principal delas é a possibilidade de controlar a quantidade de memória a ser usada durante as iterações de pontos interiores. Foi desenvolvida uma estratégia para aumentar o preenchimento a medida que os sistemas tornam-se malcondicionados. A troca dos precondicionadores ocorre quando a FCC perde eficiência tornando lenta a convergência pelo método do gradiente conjugado; este é um bom indicativo que os sistemas já estão malcondicionados e o precondicionador separador poderá obter melhor desempenho. A abordagem híbrida foi adicionada ao código PCx [25], uma implementação eficiente de pontos interiores.

Este trabalho é constituído por mais seis capítulos. No que se segue, tem-se uma breve descrição de métodos de pontos interiores para programação linear. O método primal-dual é apresentado e são descritas algumas etapas para sua implementação. No Capítulo 3 é feita uma revisão de técnicas usadas para resolver os sistemas de equações de Newton, como a fatoração de Cholesky e os métodos do subespaço de Krylov. Este capítulo foi desenvolvido com o intuito de apresentar as principais estratégias utilizadas na solução desses sistemas e fornecer o embasamento téorico dos métodos. O Capítulo 4 trata de técnicas de precondicionamento. São apresentados os precondicionadores desenvolvidos para aplicações gerais,

como o escala diagonal e os baseados em fatorações incompletas. Também é feita uma revisão de alguns precondicionadores especialmente projetados para métodos de pontos interiores. No Capítulo 5 é descrita a abordagem híbrida. Os experimentos numéricos obtidos com o precondicionador híbrido são apresentados no Capítulo 6. As conclusões e os trabalhos que poderão ser realizados futuramente encontram-se no Capítulo 7. Finalizando, têm-se as Referências bibliográficas.

Capítulo 2

Métodos de pontos interiores para programação linear

Os métodos de pontos interiores foram propostos na década de 1960 [26], mas tornaram-se um campo de pesquisa atrativo depois da publicação do trabalho de Karmarkar [55], onde foi apresentado um método polinomial para resolver problemas de programação linear. Estes métodos, como o próprio nome sugere, buscam a solução ótima de um problema de programação linear percorrendo o interior de uma região viável.

Os algoritmos de pontos interiores são descritos como primal, dual, ou primal-dual, dependendo do espaço em que estão sendo realizadas as iterações. Uma iteração é dita ser viável quando satisfaz o problema restrito e inviável caso contrário. Pode-se classificar os algoritmos de pontos interiores em três categorias: métodos afim escala [26], métodos de redução potencial [10] e métodos de trajetória central [48, 52]. Em nosso trabalho, restringiremos ao uso de um algoritmo primal-dual inviável, pertencente a categoria trajetória central. Esse algoritmo é considerado como o mais eficiente dentre as variantes de pontos interiores.

A teoria que fundamenta os procedimentos descritos neste capítulo é discutida detalhadamente em excelentes livros e artigos de métodos de pontos interiores publicados na década de 1990. Dentre eles estão Roos *et al.* [81], Vanderbei [88], Wright [91], Gondzio e Terlaky [47] e Andersen *et al.* [9]. A seguir, encontram-se definidos os problemas de programação linear. Uma breve descrição para o método primal-dual é apresentada na Seção 2.2. Na Seção 2.3 estão enumeradas as principais etapas para implementação de um algoritmo primal-dual.

2.1 Problemas de programação linear

Problemas de programação linear (PPL) têm como objetivo determinar uma solução que minimiza uma função, sujeita a um conjunto de equações e/ou inequações lineares [24]. Estão presentes em muitas áreas de aplicação, com ênfase em problemas de economia e engenharia.

Considere o seguinte modelo de programação linear no formato padrão,

$$\begin{aligned} \min \quad & c^T x \\ \text{s.a.} \quad & Ax = b \\ & x \geq 0, \end{aligned} \tag{2.1}$$

sendo $A \in \mathbb{R}^{m \times n}$ a matriz de restrições, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ e $x \in \mathbb{R}^n$ os vetores de termos independentes, custos e variáveis de decisão, respectivamente.

A este modelo, denominado primal, tem-se associado o seguinte problema no formato dual,

$$\begin{aligned} \max \quad & b^T y \\ \text{s.a.} \quad & A^T y + z = c \\ & z \geq 0, \end{aligned} \tag{2.2}$$

onde $y \in \mathbb{R}^m$ é um vetor de variáveis livres e $z \in \mathbb{R}^n$ representa as variáveis de folga.

O problema primal-dual é construído baseado nas condições de otimalidade de primeira ordem (Karush-Kuhn-Tucker) dos problemas (2.1) e (2.2), ou seja

$$\begin{aligned} Ax - b &= 0 \\ A^T y + z - c &= 0 \\ XZe &= 0 \\ (x, z) &\geq 0. \end{aligned} \tag{2.3}$$

sendo $X = \text{diag}(x)$, $Z = \text{diag}(z)$, $e \in \mathbb{R}^n$, tal que $e^T = (1, 1, \ldots, 1)$.

A solução deste problema é determinada resolvendo o sistema de equações não lineares (2.3). Se (x, y, z) for uma solução do problema primal-dual, então x e (y, z) são soluções ótimas

dos problemas (2.1) e (2.2), respectivamente. Um ponto (x, y, z) é dito ser viável se ele satisfizer o conjunto de restrições dos problemas primal e dual e, o ponto é dito ser interior se $(x, z) > 0$.

Os problemas de programação linear quando são formulados na prática, dificilmente se encontram no formato (2.1), considerado padrão. Em geral, apresentam variáveis que podem ser não negativas, livres ou limitadas e restrições na forma de equações ou inequações. Todos estes problemas podem ser reduzidos ao formato padrão acrescentando variáveis e restrições. Para simplificar a notação, optamos por apresentar os métodos de solução para problemas no formato padrão.

2.2 Método primal-dual

Os métodos primal-dual foram introduzidos por Megiddo [65], desenvolvidos por Kojima, Mizuno e Yoshise [58] e são considerados os mais eficientes dentre as inúmeras variantes de métodos de pontos interiores. Além de apresentarem melhores propriedades teóricas para a análise de complexidade [69] e convergência [93], os testes numéricos, mostram-se superiores aos realizados com abordagens primal e dual [61]. A maioria destes algoritmos utiliza o método de Newton em suas iterações. Antes de descrevermos um algoritmo pertencente a esta classe, vamos apresentar o método de Newton.

2.2.1 Método de Newton

Resolver um sistema de equações não lineares, consiste em determinar, simultaneamente, o zero de um conjunto de funções de n variáveis. A solução para este problema pode ser obtida pelo método de Newton-Raphson, que é uma generalização do método de Newton para cálculo de zero de função unidimensional.

Seja $F : \mathbb{R}^n \to \mathbb{R}^n$ um vetor de funções não lineares F_i definidas $\mathbb{R}^n \to \mathbb{R}$, com as seguintes propriedades:

(i) existe $v^* \in \mathbb{R}^n$, tal que $F(v^*) = 0$;

(ii) a função F é contínua e diferenciável na vizinhança de v^*;

(*iii*) para qualquer v diferente de v^*, a matriz das derivadas parciais de F, denominada matriz Jacobiana (J) é não singular.

O método de Newton para determinar $F(v^*) = 0$ pode ser derivado usando a série de Taylor para expandir F nas vizinhanças de um ponto v^k,

$$F(v^k + \triangle v^k) = F(v^k) + J(v^k)\triangle v^k + O(\triangle v^k)^2.$$

Eliminado os termos de ordem 2 e superior e fazendo $F(v^k + \triangle v^k) = 0$, tem-se:

$$J(v^k)\triangle v^k = -F(v^k).$$

A solução deste sistema de equações lineares fornece a direção que move, simultaneamente, cada função mais próxima de um zero. As equações deste sistema são conhecidas como equações de Newton e $\triangle v^k$ como direção de Newton.

O novo ponto é calculado por

$$v^{k+1} = v^k + \triangle v^k \ .$$

O processo continua até que um critério de parada seja atingido, por exemplo,

$$\sum_{i=1}^{n} \|F_i\| < \epsilon_1 \ \text{ e/ou } \ \sum_{i=1}^{n} |\triangle v^k| < \epsilon_2 \ ,$$

onde $\epsilon_1, \epsilon_2 > 0$ são as tolerâncias pré-estabelecidas.

A Figura 2.1 apresenta um algoritmo de Newton-Raphson para calcular a solução de um sistema de equações não lineares.

2.2.2 Algoritmo primal-dual

Os algoritmos de pontos interiores primal-dual podem ser vistos como a aplicação do método de Newton para calcular aproximações da solução de uma seqüência de sistemas não lineares (2.3) perturbados por um parâmetro μ, originado do uso da função barreira para relaxar as restrições de não negatividade,

$$F_\mu(x, y, z) = \begin{bmatrix} Ax - b \\ A^T y + z - c \\ XZe - \mu e \end{bmatrix} = 0, \ \ (x, z) \geq 0. \tag{2.4}$$

```
Algoritmo Newton_Raphson
{ Objetivo: Resolver sistemas não lineares. }
parâmetros de entrada v^0, toler, F
   { solução inicial, tolerância, vetor de funções não lineares }
parâmetros de saída v { solução }
   k ← 0
   enquanto critério de convergência > toler faça
      Calcule J(v^k), J_{i,j} = ∂F_i/∂v_j |_{v_j^k}, i,j = 1,...,n
      Determine △v^k tal que J(v^k)△v^k = −F(v^k)
      v^{k+1} ← v^k + △v^k
      k ← k + 1
   fim enquanto
   v ← v^{k+1}
fim algoritmo
```

Figura 2.1: Algoritmo de Newton-Raphson.

Observe que as duas primeiras equações são lineares e forçam a viabilidade primal e dual. A terceira equação é não linear e atende à condição de complementaridade quando $\mu = 0$. Note que, se $\mu = 0$ então os problemas (2.3) e (2.4) são equivalentes. Logo, a solução do problema (2.4) se aproxima da solução do problema primal-dual conforme $\mu \to 0$.

Um algoritmo primal-dual obtém uma solução aproximada para o problema gerando uma seqüência de pontos (x^k, y^k, z^k) e de parâmetros μ_k. O conjunto de pontos (x^k, y^k, z^k) satisfazendo (2.4) para todo $\mu > 0$ é denominado trajetória central e toda solução (x, y, z) não negativa é chamada centro analítico. Dada uma solução e um parâmetro μ associado, pode-se medir o erro na complementaridade calculando o chamado *gap de complementaridade*, dado por

$$g = x^T z \ .$$

Se a solução for viável, esse valor será reduzido ao usual *gap de dualidade* ($g = \mu e^T e = n\mu$).

A cada iteração do algoritmo é aplicado um passo do método de Newton para resolver o sistema (2.4), com um dado parâmetro μ_k. A direção de Newton é obtida da solução do

seguinte sistema de equações lineares:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x^k \\ \triangle y^k \\ \triangle z^k \end{bmatrix} = \begin{bmatrix} b - Ax^k \\ c - A^T y^k - z^k \\ \mu_k e - X^k Z^k e \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_{\mu_k} \end{bmatrix}. \tag{2.5}$$

O novo ponto é calculado fazendo,

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + (\alpha_P \triangle x^k, \alpha_D \triangle y^k, \alpha_D \triangle z^k),$$

sendo α_P e α_D os passos primal e dual, respectivamente. Esses valores são determinados para preservar a não negatividade das variáveis x e z.

O parâmetro μ_k é decrescido e o processo se repete até que seja encontrada uma solução suficientemente próxima da solução ótima ou seja detectada a inviabilidade do problema.

Na figura 2.2 é apresentado um algoritmo onde o parâmetro da barreira é

$$\mu_k = \lambda (x^k)^T z^k / n \ , \ \text{ sendo } \ \lambda \in [0, 1],$$

e os tamanhos máximos para os passos são determinados por

$$\alpha_P = \max\{\alpha \in [0,1] \mid x^k + \alpha \triangle x^k \geq 0\}, \quad \alpha_D = \max\{\alpha \in [0,1] \mid z^k + \alpha \triangle z^k \geq 0\}.$$

Este algoritmo é conhecido como primal-dual inviável. As iterações geradas são interiores mas não satisfazem, necessariamente, as restrições de igualdade.

A cada iteração do algoritmo é necessário resolver um sistema de equações lineares para determinar a direção de Newton. Quando métodos diretos estão sendo utilizados é feita uma fatoração da matriz e em seguida são resolvidos dois sistemas lineares mais simples, aos quais estão associadas matrizes triangulares. O processo de fatoração apresenta ordem de complexidade $O(n^3)$ enquanto a solução de sistemas triangulares pode ser feita com complexidade $O(n^2)$. Algumas variantes de métodos de pontos interiores resolvem a cada iteração vários sistemas com a mesma matriz dos coeficientes e com isso determinam uma melhor direção, reduzindo o número total de iterações e, conseqüentemente, o número de fatorações.

2.2.3 Algoritmo preditor-corretor

O algoritmo preditor-corretor proposto por Mehrotra [67] difere do apresentado na seção anterior, no cálculo das direções de Newton. A direção $\triangle = (\triangle x, \triangle y, \triangle z)$ é decomposta em

```
Algoritmo primal_dual
{ Objetivo: Resolver PPL usando algoritmo primal-dual. }
parâmetros de entrada (x^0, y^0, z^0),  toler
   { solução inicial, tolerância }
parâmetros de saída (x, y, z),  { solução }
   k ← 0
   Determine λ tal que λ ∈ (0, 1)
   μ^k ← λ(x^0)^T z^0/n
   enquanto critério de convergência > toler faça
      Calcule (△x^k, △y^k, △z^k) resolvendo o sistema (2.5)
      Determine α_P, α_D
      (x^{k+1}, y^{k+1}, z^{k+1}) ← (x^k, y^k, z^k) + (α_P △x^k, α_D △y^k, α_D △z^k)
      μ^{k+1} ← λ(x^{k+1})^T z^{k+1}/n
      k ← k + 1
   fim enquanto
   (x, y, z) ← (x^{k+1}, y^{k+1}, z^{k+1})
fim algoritmo
```

Figura 2.2: Algoritmo primal-dual.

duas partes, $\triangle = \triangle_a + \triangle_c$.

O termo $\triangle_a$ é obtido resolvendo o sistema (2.5) para $\mu = 0$, ou seja,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \triangle_a x^k \\ \triangle_a y^k \\ \triangle_a z^k \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ -X^k Z^k e \end{bmatrix}. \tag{2.6}$$

A componente $\triangle_a$, conhecida como direção afim-escala, é responsável por determinar a melhor predição para o parâmetro da barreira. Este parâmetro é escolhido usando a seguinte heurística:

$$\mu_k = \left(\frac{(x^k + \alpha_{pa} \triangle_a x^k)^T (z^k + \alpha_{da} \triangle_a z^k)}{(x^k)^T z^k} \right)^2 \left(\frac{(x^k + \alpha_{pa} \triangle_a x^k)^T (z^k + \alpha_{da} \triangle_a z^k)}{n} \right),$$

sendo α_{pa} e α_{da} os passos que preservam a não negatividade das variáveis x e z.

A direção corretora, $\triangle_c$ é dada por,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \triangle_c x^k \\ \triangle_c y^k \\ \triangle_c z^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mu_k e - \triangle_a X^k \triangle_a Z^k e \end{bmatrix}, \tag{2.7}$$

onde, $\triangle_a X^k = \text{diag}(\triangle_a x^k)$, $\triangle_a Z^k = \text{diag}(\triangle_a z^k)$. O termo $\triangle_c$ tem por objetivo fazer com que a nova iteração seja a mais centrada possível. Note que, o vetor de termos independentes depende da solução do sistema (2.6).

A direção de Newton pode ser obtida diretamente, resolvendo o sistema (2.7) com o vetor de termos independentes substituido por $[\ r_p \quad r_d \quad \mu_k e - \triangle_a X^k \triangle_a Z^k e - X^k Z^k e\ ]^T$. Dessa forma, evita-se o cálculo $\triangle = \triangle_a + \triangle_c$.

O acréscimo de trabalho ocorrido por resolver dois sistemas lineares a cada iteração é compensado considerando a redução significativa no número de iterações. Esta variante está implementada na maioria dos softwares livres (acadêmicos) e comerciais e está presente em muitos livros de métodos de pontos interiores.

2.3 Implementações de métodos pontos interiores

A seguir, encontram-se descritas as principais etapas para implementações de métodos de pontos interiores:

- Preprocessamento: modelos de programação linear freqüentemente contêm informações redundantes e apresentam estruturas que permitem que algumas componentes sejam facilmente identificadas. O preprocessamento tem por objetivo detectar e rearranjar essas informações para reduzir o tamanho do problema, tornando mais fácil sua solução. Por exemplo, podem ser eliminadas linhas e colunas duplicadas, algumas variáveis podem ser consideradas fixas, restrições redundantes podem ser removidas. Além disso, o preprocessamento também pode identificar inviabilidade primal e dual. Outras técnicas para melhorar o desempenho de implementações de pontos interiores podem ser feitas na fase de preprocessamento, tais como, determinar combinações de linhas que tornem a matriz A mais esparsa, tratar colunas densas separadamente, entre outras.

- Solução inicial: a escolha deste ponto é crucial para o bom desempenho de métodos de pontos interiores. Andersen *et al.* [9], seguindo as idéias de Mehrotra [67] propõem resolver o seguinte problema quadrático para obter a solução inicial

$$\begin{aligned} \min \quad & c^T x + \tfrac{\varrho}{2}(x^T x) \\ \text{s.a.} \quad & Ax = b \end{aligned} \tag{2.8}$$

sendo ϱ um parâmetro de peso pré-estabelecido. A solução de (2.8) pode ser calculada com custo comparável a uma iteração de pontos interiores, usando uma fórmula explícita. Se forem determinados resultados negativos para componentes em x eles serão substituídos por valores positivos suficientemente próximos de zero. Uma solução inicial dual (y, z) pode ser construída atribuindo $y = 0$.

- Ajuste dos parâmetros: no algoritmo apresentado na Figura 2.2, o parâmetro da barreira é dado por $\mu = \lambda(x^T z)/n$, sendo o passo $\lambda \in [0, 1]$. Para garantir que todas as iterações permaneçam próximas à trajetória central o parâmetro da barreira deve ser decrescido suavemente. Algoritmos que trabalham neste cenário, são denominados métodos de passo curto e são conhecidos por apresentarem boas propriedades numéricas, mas infelizmente, na prática, apresentam convergência lenta. Em métodos de passo longo, o parâmetro é reduzido rapidamente e para garantir que a iteração permaneça próxima à trajetória central é necessário que muitos passos de Newton sejam calculados. Implementações práticas ignoram este fato e calculam apenas um passo de Newton a cada iteração. Ainda que as iterações estejam em uma vizinhança relativamente grande da trajetória central é possível mostrar que o método pode ter convergência rápida. Outro fator que influencia o desempenho de implementações de pontos interiores é a escolha dos passos primal α_P e dual α_D. Primeiro é atribuido o tamanho máximo para esses passos no intervalo $[0, 1]$, e então é aplicado um fator de redução $\alpha_0 = 0{,}99995$ para garantir que o novo ponto seja estritamente positivo. Alguns códigos atribuem valores menores para o fator de redução nas iterações em que α_0 é considerado ser muito agressivo.

- Critério de parada: métodos de pontos interiores terminam quando as condições de otimalidade (2.3) são satisfeitas sob uma tolerância pré-determinada. Essas condições podem ser medidas calculando a viabilidade primal e dual relativa,

$$\mathsf{PriInf} = \frac{\|Ax - b\|}{1 + \|b\|} \leq \epsilon \,, \qquad \mathsf{DuInf} = \frac{\|A^T y + z - c\|}{1 + \|c\|} \leq \epsilon \,,$$

e o gap de dualidade relativo,

$$\mathsf{Gap} = \frac{|c^T x - b^T y|}{1 + |b^T y|} \leq \epsilon \ .$$

É comum encontrar na literatura $\epsilon = 10^{-8}$. Na prática, as condições de viabilidade primal e dual são freqüentemente satisfeitas antes do gap relativo. Segundo Andersen *et al.* [9], o gap é a condição mais importante e provavelmente é a única que deve realmente ser considerada.

Uma outra etapa que influencia diretamente o desempenho de implementações de métodos de pontos interiores é a solução dos sistemas de equações de Newton. No capítulo seguinte são descritas as principais estratégias usadas na solução desses sistemas.

Capítulo 3

Solução do sistema de equações de Newton

A cada iteração de um método de pontos interiores é necessário resolver um ou mais sistemas lineares. O processo utilizado na solução é de extrema importância para o desempenho dos códigos, já que é a tarefa que demanda a maior parte do tempo de processamento.

Neste capítulo serão descritas as principais estratégias de solução, as quais envolvem duas formulações distintas. A primeira delas trata com matrizes simétricas e definidas positivas e à outra estão associadas matrizes simétricas indefinidas. Neste cenário, serão apresentados métodos diretos e iterativos para solução dos sistemas.

3.1 Estratégias de solução

Como visto no Capítulo 2, as direções de Newton são obtidas resolvendo um ou mais sistemas lineares. Uma vez que esses sistemas compartilham a mesma matriz de coeficientes, restringiremos nossa discussão à solução do seguinte sistema linear,

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \\ \triangle z \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - z \\ \mu e - XZe \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_\mu \end{bmatrix}. \tag{3.1}$$

Em geral, as implementações de pontos interiores trabalham com sistemas de menor dimensão, obtidos com a eliminação de algumas variáveis. O sistema linear (3.1) pode ser transformado no chamado *sistema aumentado* usando a terceira equação para eliminar $\triangle z$ da segunda equação. Definindo $\Theta = (Z^{-1}X)$ tem-se o sistema resultante,

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1}r_\mu \\ r_p \end{bmatrix} . \tag{3.2}$$

Este sistema ainda pode ser reduzido às *equações normais*, usando a primeira equação para eliminar $\triangle x$ da segunda equação, ou seja,

$$(A\Theta A^T)\triangle y = A\Theta(r_d - X^{-1}r_\mu) + r_p \; . \tag{3.3}$$

As variáveis $\triangle x$ e $\triangle z$ são facilmente calculadas,

$$\triangle x = \Theta A^T \triangle y - \Theta(r_d - X^{-1}r_\mu)$$
$$\triangle z = X^{-1}(r_\mu - Z\triangle x) \; .$$

Todos os métodos de pontos interiores resolvem sistemas similares para calcular a direção de Newton, o que os diferencia é a maneira como a matriz Θ e o vetor de termos independentes são construídos. Para métodos de pontos interiores na programação linear, Θ é diagonal e possui todos os elementos positivos. Conforme a solução ótima se aproxima alguns valores de Θ tendem à infinito, enquanto outros se aproximam de zero, causando sérios problemas de instabilidade numérica.

3.1.1 Equações normais

A maioria das implementações de pontos interiores utilizam métodos diretos para resolver os sistemas de equações normais [7, 61, 62]. A abordagem mais comum de solução decompõe a matriz de (3.3) em LDL^T, sendo D uma matriz diagonal e L uma triangular inferior. Este método é um tipo de fatoração de Cholesky e está descrito na seção 3.2. Os sistemas de equações normais têm a vantagem de trabalhar com matrizes que apresentam boas propriedades numéricas: são simétricas e definidas positivas. Quando estão sendo aplicados métodos diretos, a decomposição dessas matrizes é estável e dispensa estratégias de pivotação. Entretanto, existem algumas desvantagens nesta formulação. Uma delas é a instabilidade originada em problemas que apresentam variáveis livres. Outro problema, ainda

mais grave, é a perda de esparsidade de $A\Theta A^T$, na simples presença de colunas densas em A. Para evitar o preenchimento, foram propostas estratégias que tratam estas colunas separadamente. O sucesso destas técnicas depende da escolha de boas heurísticas para determinar quais colunas devem ser consideradas densas, e esta não é uma tarefa fácil.

O problema da perda de esparsidade pode ser contornado com o uso de abordagens iterativas para resolver os sistemas. Nesses métodos, a matriz é requerida apenas para calcular o produto matriz-vetor e, portanto, $A\Theta A^T$ não precisa ser fisicamente construída, a menos que seja necessária para o cálculo do precondicionador. O método iterativo mais utilizado em algoritmos de pontos interiores é o gradiente conjugado precondicionado (ver por exemplo, [6, 66, 80, 90]). Algumas dessas implementações utilizam a fatoração incompleta da matriz $A\Theta A^T$ para construir os precondicionadores e, conseqüentemente, podem sofrer com a perda de esparsidade.

3.1.2 Sistemas aumentados

Os sistemas aumentados (3.2) possuem matrizes esparsas, simétricas e indefinidas. A fatoração de Cholesky não se aplica, já que não há uma forma numericamente estável para decompor uma matriz indefinida em LDL^T com D diagonal. Um método direto bastante utilizado em implementações de pontos interiores que tratam com sistemas aumentados é a fatoração de Bunch-Parlett [17]. Outra alternativa que tem sido usada, transforma o sistema indefinido em um quase definido utilizando o método de regularização primal e dual [8]. Neste caso, a fatoração LDL^T, com D diagonal existe, para qualquer permutação simétrica de linhas ou colunas da matriz. Estes sistemas apresentam algumas vantagens em relação às equações normais: os fatores resultantes na decomposição normalmente são mais esparsos e numericamente mais estáveis; variáveis livres e colunas densas são facilmente tratadas [89]. Estes fatos, motivaram muitos pesquisadores a incluírem esta formulação em seus códigos [34, 71].

Abordagens iterativas também são utilizadas para calcular a direção de Newton via sistema aumentado [15, 31, 42, 57]. O método do gradiente conjugado não é adequado, já que pode falhar na solução de sistemas com matrizes indefinidas. Entretanto, ele tem sido aplicado com sucesso usando precondicionadores adequados [15, 31]. Métodos como MINRES e SYMMLQ [76] são mais apropriados para resolver os sistemas indefinidos.

Na prática, as implementações que utilizam matrizes simétricas e definidas positivas, tipicamente, mostram-se superiores na solução de problemas de programação linear, quando comparadas às que tratam com os sistemas aumentados. Esta última abordagem é mais interessante na solução de problemas de programação quadrática e não linear.

A seguir são descritos métodos diretos e iterativos para resolver os sistemas de equações de Newton. Para simplificar a notação, vamos considerar o sistema como $Ax = b$.

3.2 Abordagem direta: fatoração de Cholesky

Seja $Ax = b$ um sistema de equações lineares. Quando $A \in \mathbb{R}^{n\times n}$ for uma matriz simétrica e definida positiva, isto é, $v^T Av > 0$, $\forall v \in \mathbb{R}^n$ e $v \neq 0$, ela pode ser decomposta em

$$A = LL^T$$

onde L é uma matriz triangular inferior.

A decomposição $A = LL^T$ é conhecida como decomposição de Cholesky e sua existência é garantida pelo teorema 3.1 ([43, teorema 4.2.5]).

Teorema 3.1. *Se $A \in \mathbb{R}^{n\times n}$ for uma matriz simétrica e definida positiva, então existe uma única matriz triangular $L \in \mathbb{R}^{n\times n}$ com elementos da diagonal positivos tal que $A = LL^T$.*

Para resolver o sistema $Ax = b$, usa-se a matriz na forma decomposta

$$Ax = b \Rightarrow (LL^T)x = b\,.$$

Fazendo,

$$Ly = b \;\text{ e }\; L^T x = y,$$

tem-se que a solução y do sistema triangular inferior é obtida pelas substituições sucessivas e a solução x do sistema triangular superior, pelas substituições retroativas.

Os elementos da matriz L são dados por:

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} {l_{jk}}^2}\,;$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}} \quad \text{para } i > j.$$

Uma forma alternativa para resolver o sistema $Ax = b$ de modo a evitar o uso da raíz quadrada, pode ser feita decompondo a matriz em

$$A = LDL^T$$

onde L é uma matriz triangular inferior unitária ($l_{ii} = 1, \forall i$) e D é uma matriz diagonal.

O sistema $Ax = b$ é resolvido usando a matriz A na forma decomposta,

$$Ax = b \Rightarrow (LDL^T)x = b\,.$$

Fazendo,

$$Ly = b, \quad Dz = y \ \text{ e } \ L^T x = z\,,$$

tem-se que a solução y do sistema é obtida pelas substituições sucessivas e a solução x pelas substituições retroativas. As matrizes L e D são calculadas por

$$d_{jj} = a_{jj} - \sum_{k=1}^{j-1} {l_{jk}}^2 d_{kk}\,;$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}}{d_{jj}}\,, \text{ para } i > j.$$

As matrizes definidas positivas quando decompostas possuem alta estabilidade numérica, dispensando com isto o uso de estratégias de pivotação. Já para matrizes indefinidas, o uso de estratégias de pivotação pode ser necessário, conforme [43].

Na prática, muitas estratégias devem ser utilizadas para que a fatoração de Cholesky tenha êxito ao ser aplicada na solução do sistema (3.3). São pontos cruciais, a manipulação de colunas densas e a escolha do reordenamento que produza fatores com preenchimento mínimo. Implementações eficientes podem ser encontradas em [40, 44].

Uma das vantagens de se trabalhar com métodos diretos, como a fatoração de Cholesky, é a precisão obtida na solução. No entanto, seu uso pode se tornar proibitivo quando a matriz L, obtida da fatoração, for muito mais densa que a matriz original. Nestes casos, métodos iterativos tornam-se mais atrativos, desde que sejam utilizados com precondicionadores adequados.

3.3 Abordagem iterativa: métodos do subespaço de Krylov

Os métodos baseados no subespaço de Krylov são utilizados tanto para resolver sistemas lineares $Ax = b$ quanto para encontrar autovalores de A. Em nosso trabalho será abordada a solução dos sistemas lineares. A idéia básica por trás desses métodos é construir aproximações ótimas da solução em uma seqüência de subespaços que aumentam de dimensão a cada iteração, denominados subespaços de Krylov.

O mais conhecido dos métodos do subespaço de Krylov é o gradiente conjugado (CG), proposto em 1952 por Hestenes e Stiefel [51]. Ele foi desenvolvido para resolver sistemas de equações lineares com matrizes simétricas e definidas positivas.

Em 1975 foram concebidos MINRES e SYMMLQ, por Paige e Saunders [76]. Esses métodos são baseados no algoritmo de Lanczos [59] e podem ser aplicados para resolver sistemas com matrizes simétricas e indefinidas. No ano seguinte, Fletcher [32] propôs o BiCG, baseado no algoritmo Bi-Lanczos [82], sendo portanto, adequado a matrizes não simétricas.

Uma forma alternativa para trabalhar com matrizes não simétricas e indefinidas é transformar o sistema em equações normais, ou seja, $A^T Ax = A^T b$. Dois métodos foram desenvolvidos baseados nesta abordagem, CGNE e CGNR [77]. O primeiro deles aplica o gradiente conjugado às equações normais e o outro usa minimização da norma do resíduo.

Em 1986 foi desenvolvido o GMRES, por Saad e Schultz [84]. Este método é uma extensão do MINRES para sistemas com matrizes não simétricas.

Sonneveld, em 1989, propôs o CGS [86], como uma tentativa para melhorar a convergência do BiCG e evitar o cálculo de A^T. Três anos depois, van der Vorst [87] desenvolveu o BiCGStab, introduzindo um parâmetro de estabilidade para suavizar o comportamento irregular de convergência do CGS.

Em 1991, Freund e Nachtigal [37], propuseram o QMR. Este método trabalha com o produto de duas normas e aplica a minimização apenas em uma delas usando o processo Bi-Lanczos. Freund, em 1993, fez uma adaptação à este método para evitar o uso de transpostas, produzindo o TFQMR [36]. Chan *et al.* propuseram o acréscimo do parâmetro de estabilidade

e desenvolveram o QMRCGStab [23].

3.3.1 Embasamento teórico dos métodos

Um subespaço de Krylov de dimensão m, denotado por $K_m(A; r_0)$, é definido pelo espaço gerado pelo vetores

$$r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0,$$

onde $r_0 = b - Ax_0$ denota o resíduo da solução inicial. Uma aproximação x_m é considerada ótima se o resíduo for ortogonal ao subespaço onde x_m é escolhida, ou seja

$$b - Ax_m \perp K_m \ .$$

Esta relação é conhecida por condição de Galerkin. A solução x_m será calculada no subespaço afim $x_0 + K_m$, onde x_0 denota a solução inicial.

A matriz original A pode ser projetada no subespaço de Krylov com o uso de uma transformação ortogonal, produzindo uma matriz mais simples H denominada Hessenberg superior,

$$H_m = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & \ldots & \vdots & \vdots \\ & h_{3,2} & \ldots & \vdots & \vdots \\ & & \ddots & \vdots & \vdots \\ & & & h_{m,m-1} & h_{m,m} \end{pmatrix}.$$

Essa transformação é obtida calculando bases ortonormais do subespaço de Krylov. A seguir serão vistos dois processos para calcular estas bases:

Algoritmo de Arnoldi

O método de Arnoldi foi proposto em 1951 [11] com o objetivo de transformar uma matriz densa em Hessenberg superior. Arnoldi sugeriu que os autovalores de H_m seriam boas aproximações dos autovalores da matriz original. Esta estratégia deu origem a muitos métodos para calcular autovalores de matrizes esparsas e com dimensões elevadas.

O algoritmo de Arnoldi é um procedimento para construir uma base ortonormal ao subespaço de Krylov. Considere um subespaço $K_1(A, r0) = \{r_0\}$. Uma base ortonormal para este subespaço é $V = \{v_1\} = \{r_0/\|r_0\|\}$. Para determinar uma base $V = \{v_1, v_2, \ldots, v_m\}$ de $K_m(A, r_0)$, pode ser utilizado o algoritmo apresentado na Figura 3.1.

```
Algoritmo Arnoldi
{ Objetivo: Determinar uma base ortonormal do subespaço de Krylov. }
parâmetros de entrada v_1,  m
   { vetor de norma um, dimensão do subespaço }
parâmetros de saída V = {v_1, v_2, ... v_m},  { base ortonormal de K_m }
   V ← {v_1}
   para j = 1 até m faça
      soma ← 0
      para i = 1 até j faça
         h_{i,j} ← < A * v_j, v_1 >  { <,>: produto interno , * : produto matriz-vetor  }
         soma ← soma+ < h_{i,j}, v_i >
      fim para
      w_j ← A * v_j − soma
      h_{j+1,j} ← ||w_j||_2
      se h_{j+1,j} = 0 então
         interrompa
      fim se
      v_{j+1} ← w_j/h_{j+1,j}
      V ← V ∪ {v_{j+1}}
   fim para
fim algoritmo
```

Figura 3.1: Algoritmo de Arnoldi.

Este método utiliza ortogonalização total, uma vez que os vetores v_{j+1} são ortogonalizados em relação a todos os outros da base. À medida que o tamanho da base aumenta, o espaço requerido para armazenamento e o tempo de processamento pode ser muito elevado.

A transformação ortogonal para reduzir A em H pode ser feita levando em consideração a proposição 3.2 ([83, proposição 6.5]).

Proposição 3.2. *Seja V_m uma matriz de dimensão $(n \times m)$ formada pelos vetores da base $v_1, v_2, \ldots, v_m$; $\tilde{H}_m$ a matriz Hessenberg construída com elementos $h_{i,j}$ calculados pelo algoritmo da Figura 3.1 e H_m obtida retirando a última linha de $\tilde{H}_m$. Então, tem-se:*

$$AV_m = V_m H_m + w_m {e_m}^T = V_{m+1}\tilde{H}_m,$$

$$V_m^T A V_m = H_m,$$

sendo ${e_m}^T = (1, 1, \ldots, 1) \in \mathbb{R}^m$.

Aproximações para solução do sistema $Ax = b$ em um espaço m-dimensional podem ser obtidas fazendo,

$$x_m = x_0 + V_m y_m,$$

sendo y_m a solução do sistema linear

$$H_m y_m = \|r_0\| e_1,$$

com $e_1^T = (1, 0, \ldots, 0)$, $r_0 = b - Ax_0$. Note que, além de H possuir dimensão muito menor que a da matriz A ela é estruturalmente mais simples.

Algoritmo de Lanczos

Lanczos [59] propôs seu método em 1950. Este pode ser visto como uma simplificação feita no algortimo de Arnoldi para trabalhar com matrizes simétricas. Quando A possui esta característica a situação passa a ser muito mais favorável, já que vários termos $h_{i,j}$ se anulam reduzindo a matriz H em uma matriz tridiagonal simétrica,

$$H_m = T_m = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-2} & \alpha_{m-1} & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \end{pmatrix}. \tag{3.4}$$

Na Figura 3.2 está descrito o algoritmo de Lanczos para gerar uma base ortonormal do subespaço de Krylov. Note que, a cada iteração do algoritmo é necessário armazenar apenas três vetores para construir a base ortonormal, fazendo com que o tempo computacional

```
Algoritmo Lanczos
{ Objetivo: Determinar uma base ortonormal do subespaço de Krylov. }
parâmetros de entrada v_1,  m
   { vetor de norma um, dimensão do subespaço }
parâmetros de saída V = {v_1, v_2, ... v_m},  { base ortonormal de K_m }
   V ← {v_1}
   β_1 ← 0 ;  v_0 ← 0
   para j = 1 até m faça
      w_j ← A * v_j − β_j * v_{j−1}
      α_j ← < w_j, v_j >  { <,>: produto interno }
      w_j ← w_j − α_j * v_j
      β_{j+1} ← ||w_j||_2
      se β_{j+1} = 0 então
         interrompa
      fim se
      v_{j+1} ← w_j/β_{j+1}
      V ← V ∪ {v_{j+1}}
   fim para
fim algoritmo
```

Figura 3.2: Algoritmo de Lanczos.

e a quantidade de armazenamento sejam reduzidos significativamente, em comparação ao algoritmo de Arnoldi.

Quando aritmética de precisão finita estiver sendo utilizada, os vetores que geram as bases tendem a perder ortogonalidade e conseqüentemente autovalores com o mesmo valor são gerados. Em geral, as repetições ocorrem com os autovalores mais isolados e isso faz com que a convergência dos métodos do subespaço de Krylov se torne lenta [78]. O processo utilizado para evitar esta perda de ortogonalidade é denominado reortogonalização. A seguir serão discutidas quatro formas distintas para fazê-lo.

Reortogonalização

- **Reortogonalização total:** A reortogonalização é feita pelo método de Gram-

Schmidt, sendo que cada novo vetor calculado é ortogonalizado em relação aos anteriores. Este processo evita a repetição dos autovalores e acelera a velocidade de convergência. Entretanto, com o aumento da base, o espaço de armazenamento requerido e o tempo de processamento pode ser muito elevado.

- **Reortogonalização seletiva:** Parlett e Scott [78] propuseram uma forma mais econômica para fazer a reortogonalização. Cada novo vetor da base é reortogonalizado somente em relação aos vetores de Ritz para os quais os autovalores já convergiram. Desta forma, a quantidade de armazenamento para os vetores e o tempo necessário por iteração são reduzidos, contudo é preciso calcular os vetores de Ritz. Este processo consome tempo de execução e espaço de armazenamento.

- **Reortogonalização parcial:** uma alternativa para reortogonalização foi feita por Simon [85], na qual calcula-se a cada iteração uma estimativa do nível de perda de ortogonalidade. Quando este nível atinge um valor crítico é feita a reortogonalização usando somente alguns vetores já calculados. Apesar da abordagem parcial ser mais rápida do que a total ela requer a mesma quantidade de armazenamento, já que todos os vetores anteriormente calculados precisam ser armazenados, pois a escolha dos vetores que serão usados na reortogonalização é feita durante o processo. Além do mais, o sucesso desta estratégia depende do cálculo de uma estimativa precisa para o nível de ortogonalidade e esta não é uma tarefa fácil.

- **Reortogonalização primária:** Em 1995, Campos [19] propôs uma estratégia para reortogonalizar que é ao mesmo tempo parcial e seletiva. A reortogonalização primária consiste em construir uma base ortonormal com os primeiros vetores resíduos até que alguns ou todos os vetores associados aos autovalores dominantes sejam incluídos nesta base. Conforme a base é construída, cada novo vetor é reortogonalizado em relação a todos os vetores que se encontram na base. O número de iterações para a convergência da solução decresce a medida que o tamanho da base aumenta. Este comportamento ocorre até que todos autovetores dominantes pertençam ao subespaço gerado por esta base. A partir deste momento o número de iterações para convergência é estabilizado. Quando a base for suficiente para gerar todos os autovalores requeridos a adição de novos vetores é interrompida. Os novos vetores resíduos serão reortogonalizados em relação à esta base de tamanho fixo.

Muitos métodos do subespaço de Krylov são baseados nos processos para gerar bases orto-

normais. A seguir serão apresentados dois desses métodos para resolver sistemas de equações lineares.

3.3.2 Método do Gradiente Conjugado

O método do gradiente conjugado é considerado o mais eficiente para resolver sistemas de equações lineares de grande porte, que apresenta a matriz dos coeficientes simétrica, definida positiva e esparsa. Existem várias formas para apresentar o método do gradiente conjugado, optamos por uma abordagem intuitiva e com boas interpretações geométricas. Para isto, serão revisados os métodos do máximo declive e das direções conjugadas, que são métodos menos eficientes, mas de grande utilidade teórica no desenvolvimento do gradiente conjugado. Estes métodos se baseiam na minimização de uma função f quadrática.

Considere um sistema de equações lineares $Ax = b$, onde $A \in \mathbb{R}^{n\times n}$ é uma matriz simétrica e definida positiva. Uma forma quadrática é uma função $f : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}$, definida por:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c\,, \tag{3.5}$$

com c constante.

Pode-se mostrar que resolver o sistema é equivalente a encontrar o ponto de mínimo de (3.5). Sabe-se que dado um ponto arbitrário x, o vetor gradiente aponta na direção de maior crescimento de $f(x)$. Fazendo $\bigtriangledown f(x) = Ax - b = 0$ encontra-se um ponto crítico de $f(x)$. Como A é simétrica e definida positiva, pode-se garantir que A possui inversa, e com isto o sistema tem solução única. Logo resolver este sistema é equivalente a encontrar x que minimiza $f(x)$.

Método de máximo declive

Também conhecido por método do gradiente ou de Cauchy, é um dos métodos mais antigos e conhecidos para minimização de uma função de n variáveis. Apesar de possuir convergência lenta, é de grande importância, devido ao seu embasamento teórico e sua simplicidade.

As iterações x_k são escolhidas na direção em que f decresce mais rapidamente, ou seja, na direção contrária ao gradiente, que no caso, é o resíduo. Na k-ésima iteração, tem-se:

$$d_k = - \bigtriangledown f(x_k) = b - Ax_k = r_k\,.$$

Se o resíduo for não nulo, existe $x_{k+1} = x_k + \alpha r_k$, com $\alpha_k = \frac{((r_k)^T r_k)}{((r_k)^T A r_k)}$ positivo, tal que,

$$f(x_k + \alpha_k r_k) < f(x_k) \, .$$

Método das direções conjugadas

O método das direções conjugadas é bastante similar ao método de máximo declive. No método de máximo declive, os vetores direção escolhidos são ortogonais, já no método das direções conjugadas, como o próprio nome diz, os vetores direção escolhidos serão conjugados.

Definição 3.3. *Seja $A \in \mathbb{R}^{n \times n}$, os vetores direção $d_0, d_1, d_2, ..., d_n$ pertencentes a R^n são conjugados se*

$$d_k{}^T A d_j = 0 \;\; k \geq 1 \; , \;\; j \leq n \; , \;\; k \neq j. \tag{3.6}$$

De modo similar ao método de máximo declive é necessário determinar α_k que minimiza f no ponto x_{k+1}. Este valor é dado por,

$$\alpha_k = -\frac{(d_k)^T r_k}{(d_k)^T A d_k} \, . \tag{3.7}$$

Para gerar um conjunto de n direções conjugadas linearmente independente será utilizado o processo de conjugação de Gram-Schmidt. Ele consiste em, a partir de um conjunto de vetores $u_0, u_1, u_2, ..., u_{n-1}$, construir direções conjugadas de modo que:

$$d_i = \begin{cases} u_0, & \text{se } i = 0 \\ u_i + \sum_{k=0}^{i-1} \beta_{i,k} d_k, & \text{se } i > 0 \end{cases} \tag{3.8}$$

onde os $\beta_{i,j}$ são obtidos por:

$$\beta_{i,j} = -\frac{(u_i)^T A d_j}{(d_j)^T A d_j} \, , \text{ para todo } i > j.$$

A dificuldade do uso da conjugação de Gram-Schmidt no método das direções conjugadas, é que todos os vetores direção tem que ser armazenados, pois os anteriores serão sempre utilizados na construção de cada novo vetor. O custo requerido das operações por este método é $O(n^3)$.

Método do gradiente conjugado

O método do gradiente conjugado é simplesmente o método das direções conjugadas, onde as direções de busca são construídas por conjugação dos resíduos.

Pode ser mostrado por indução que as seguintes relações ortogonais são sastisfeitas:

$$\begin{cases} r_i^T r_j = 0, & \text{para } i \neq j \\ r_i^T d_j = 0, & \text{para } i > j \ . \end{cases} \tag{3.9}$$

Proposição 3.4. *Seja $d_0 = r_0$, os termos $\beta_{i,j}$ se anulam para $i > j + 1$, com $i = 1, 2, \ldots, n-1$.*

Demonstração: O resíduo pode ser escrito como:

$$r_{j+1} = r_j - \alpha_j A d_j \ .$$

Multiplicando ambos os lados por r_i^T,

$$(r_i)^T r_{j+1} = r_i^T r_j - \alpha_j (r_i)^T A d_j \ .$$

Logo,

$$r_i^T A d_j = \frac{r_i^T r_j - (r_i)^T r_{j+1}}{\alpha_j} \ ,$$

Usando as relações ortogonais dos resíduos,

$$r_i^T A d_j = \begin{cases} \dfrac{1}{\alpha_i}(r_j^T r_j), & \text{se } i = j \\ -\dfrac{1}{\alpha_j}(r_{j+1}^T r_{j+1}), & \text{se } i = j+1 \\ 0, & \text{demais casos.} \end{cases}$$

Como $\beta_{i,j} = -\dfrac{(u_i)^T A d_j}{(d_j)^T A d_j}$ e $u_i = r_i$, tem-se

$$\beta_{i,j} = \begin{cases} \dfrac{(r_{j+1})^T r_{j+1}}{\alpha_j (d_j)^T A d_j} = \dfrac{(r_{j+1})^T r_{j+1}}{(r_j)^T r_j}, & \text{se } i = j+1 \\ 0, & \text{se } i > j+1. \end{cases}$$

Assim, para todo $i > j + 1$ tem-se que $\beta_{i,j} = 0$. ■

Com isto, muito dos termos $\beta_{i,j}$ se anulam, sendo necessário armazenar apenas a última direção para construir a nova direção conjugada, isto é:

$$d_{j+1} = r_{j+1} + \beta_{j+1,j} d_j \ ,$$

onde $\beta_{j+1,j} = \dfrac{(r_{j+1})^T r_{j+1}}{(rj)^T r_j}$.

As características de convergência dos métodos gradientes, dependem dos autovalores da matriz dos coeficientes. Se a magnitude de todos os autovalores for aproximadamente a mesma, as superfícies se tornam mais esféricas e a convergência será rápida. Para um caso mais geral, pode-se mostrar que em aritmética exata, o número de iterações necessárias para a convergência do método é o número de autovalores distintos.

A Figura 3.3 apresenta um algoritmo para calcular a solução de um sistema $Ax = b$ pelo método gradiente conjugado.

3.3.3 Método LQ simétrico - SYMMLQ

Em 1975, Paige e Saunders desenvolveram o algoritmo SYMMLQ [76] para resolver sistemas lineares com matrizes simétricas e indefinidas. Este método pode ser visto como uma aplicação do algoritmo de Lanczos nesta classe de problemas (Figura 3.4).

A solução do sistema $T_m y_m = \|r_0\| \beta_1 e_1$ deve ser obtida por métodos de fatoração numericamente estáveis, já que esta matriz pode ser indefinida. Em SYMMLQ é utilizada a fatoração ortogonal, ou seja

$$T_m = L_m Q_m \ , \quad Q_m^T Q_m = I \ ,$$

sendo L_m uma matriz triangular inferior. A forma como estes fatores são gerados está descrita em [76]. O custo computacional do SYMMLQ é superior ao gradiente conjugado.

O uso de métodos iterativos para resolver o sistema de equações de Newton só é aplicável se for possível construir bons precondicionadores, uma vez que a matriz dos coeficientes se torna altamente malcondicionada. Este assunto será discutido no próximo capítulo.

```
Algoritmo GC
{ Objetivo: Resolver o sistema Ax = b pelo método Gradiente Conjugado }
parâmetros de entrada A, b, x_0, kmax, toler
   { matriz dos coeficientes; vetor de termos independentes, solução inicial }
   { número máximo de iterações; tolerância máxima admitida }
parâmetros de saída x, Erro { solução do sistema, condição de erro }
   r_0 ← b − A * x_0
   k ← 0; d_0 ← 0; ρ_0 ← 1
   repita
      k ← k + 1
      ρ_k ← (r_{k−1})^T r_{k−1}
      β_k ← ρ_k/ρ_{k−1}
      d_k ← r_{k−1} + β_k d_{k−1}
      α_k ← ρ_k/(d_k)^T A d_k
      x_k ← x_{k−1} + α_k d_k
      r_k ← r_{k−1} − α_k A d_k
      escreva k, x_k, r_k
      se ||r_k||/||r_0|| < toler ou k = kmax
         então interrompa
      fim se
   fim repita
   Erro ← ||r_k||/||r_0|| ≥ toler
fim algoritmo
```

Figura 3.3: Algoritmo Gradiente Conjugado.

`Algoritmo Lanczos_Sol_sist`

{ `Objetivo:` Resolver um sistema linear $Ax = b$. }

`parâmetros de entrada` n A, b, x_0

{ ordem da matriz, matriz A, vetor b e solução inicial }

`parâmetros de saída` x, r { vetor solução, vetor resíduo }

$r_0 \leftarrow b - Ax_0$

$\beta_1 \leftarrow \|r_0\|_2$; $v_1 \leftarrow r_0/\beta_1$

{Calcule pelo algoritmo 3.2}

$V_m = \{v_1, v_2, \ldots, v_m\}$ { base ortonormal para $K_m(A, r_0)$ }

T_m { matriz tridiagonal 3.4 } , v_{m+1} , β_{m+1}

{Solução do sistema}

Calcule y_m tal que $T_m y_m = \|r_0\|\beta_1 e_1$

$x \leftarrow x_0 + V_m y_m$

$r \leftarrow -v_{m+1}\beta_{m+1}(e_m^T y_m)$

`fim algoritmo`

Figura 3.4: Algoritmo de Lanczos para solução de sistemas lineares.

Capítulo 4

Precondicionamento de sistemas lineares

O precondicionamento é um esquema que deve ser aplicado aos métodos iterativos para melhorar as características de convergência de sistemas que possuam a matriz de coeficientes com autovalores muito dispersos. Precondicionar um sistema linear é fazer com que a matriz dos coeficientes apresente as condições desejadas para que o método que está sendo aplicado para resolver o sistema seja eficiente. Isto pode ser feito pré-multiplicando a inversa de uma matriz auxiliar M, denominada precondicionadora, a um sistema de equações lineares $Ax = b$, da seguinte maneira

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

Muitos pesquisadores consideram que para o precondicionamento ser eficiente, suas operações devem ser de baixo custo, a matriz precondicionadora M deve ser facilmente construída, ter esparsidade próxima de A, além de, os autovalores de $M^{-1}A$ estarem próximos a unidade ou agrupados.

Existem várias formas para construir uma matriz precondicionadora M, e uma boa escolha, pode ter efeito dramático na convergência do método. De um modo geral, os precondicionadores podem ser classificados em duas classes. Em uma, estão os baseados unicamente em técnicas algébricas, como por exemplo, os precondicionadores construídos usando fatorações incompletas da matriz ou aproximações esparsas da matriz inversa. Esses precondicionadores podem ser usados em muitas aplicações, já que não é utilizada nenhuma informação

específica do problema que está sendo resolvido. Na outra classe, estão os precondicionadores projetados para aplicações particulares. Pelo fato de serem construídos explorando detalhes do problema, como estrutura, geometria, e propriedades dos coeficientes, tipicamente, apresentam desempenho superior. Benzi, Golub and Liesen [13] apresentam uma excelente revisão para essa classe de precondicionadores.

Neste capítulo são descritas algumas estratégias para precondicionar um sistema de equações lineares. Na Seção 4.1 são apresentados os precondicionadores escala diagonal e os baseados em fatorações incompletas da matriz. A Seção 4.2 traz algumas técnicas de reordenamento para minimizar o preenchimento no fator quando são usados métodos de fatoração. Uma revisão de precondicionadores usados em métodos de pontos interiores é apresentada na Seção 4.3.

4.1 Agrupamento dos autovalores

Nesta seção, serão vistos alguns precondicionadores utilizados para resolver sistemas com matrizes simétricas e definidas positiva [19]. Todos estes precondicionadores trabalham no sentido de agrupar os autovalores e assim acelerar a convergência.

4.1.1 Escala diagonal

Em 1955, Forsythe e Strauss [33], propuseram uma forma muito simples para precondicionar um sistema de equações lineares,

$$Ax = b \;\; \rightarrow \;\; (DAD)(D^{-1}x) = Db\,,$$

utilizando uma matriz diagonal D.

Para matrizes simétricas, quando $D = \mathrm{diag}(A)$, o sistema pode ser precondicionado por

$$(D^{-1/2}AD^{-1/2})y = D^{-1/2}b\,,$$

onde $(D^{-1/2}AD^{-1/2})$ é uma matriz com diagonal unitária e $y = D^{1/2}x$.

4.1.2 Fatoração incompleta de Cholesky

A fatoração incompleta de Cholesky é uma das estratégias mais importantes de se obter precondicionadores para sistemas de equações lineares, com matrizes esparsas, simétricas e definidas positivas. Muitas vezes, por limitações de memória, o uso da fatoração de Cholesky para resolver esses sistemas pode não ser possível. Em geral, o fator L será mais denso que a matriz dos coeficientes, devido ao preenchimento ocorrido durante o processo. Isto pode ser evitado utilizando a fatoração incompleta de Cholesky, ou seja, aplica-se a decomposição de Cholesky ignorando alguns elementos que causam preenchimento.

Seja $Ax = b$ um sistema de equações lineares. Uma forma fácil e eficiente para obter um precondicionador M para este sistema é determinar uma matriz triangular inferior $\tilde{L}$ que tenha os elementos não nulos nas mesmas posições dos elementos não nulos de A, tal que, $M = \tilde{L}\tilde{L}^T$. Desta forma, elimina-se elementos para reduzir o preenchimento, fazendo com que o espaço requerido para armazenar $\tilde{L}$ seja o mesmo da matriz original A. Como A é simétrica, são armazenados apenas os elementos não nulos da diagonal para baixo.

Foram desenvolvidos vários precondicionadores baseados em fatorizações incompletas. Em 1977, Meijerink e van der Vorst [68], propuseram uma classe de matrizes precondicionadoras com níveis de preenchimento, denominadas decomposição LU incompleta. Munksgaard [70] propôs um esquema de retirada por tolerância (*drop tolerance*), no qual, elimina-se somente os elementos de $\tilde{L}$ que são menores que um valor pré-estabelecido. Jones e Plassmann [53] propuseram $\tilde{L}$, com um número fixo de elementos não nulos por linha ou coluna. Este número é fixado como o número de elementos não nulos de cada linha ou coluna de A e os elementos escolhidos são os maiores em módulo, ignorando o padrão de esparsidade da matriz. Conseqüentemente, o espaço requerido para armazenar $\tilde{L}$ é fixo e pode ser determinado a priori. Além do mais, por armazenar somente os maiores elementos torna-se possível obter os mesmos benefícios do esquema de retirada por tolerância. Mais recentemente, Campos [19] propôs a fatoração controlada de Cholesky (FCC), que pode ser vista como uma generalização do esquema desenvolvido por Jones e Plassmann, na qual o preenchimento também pode variar. Na FCC o número de elementos nas colunas de $\tilde{L}$ será controlado por um parâmetro η, de tal forma que, $\tilde{L}$ poderá ter mais ou menos elementos que A. Esse parâmetro pode ser dado em função da quantidade de espaço disponível. Se a demanda por memória não for crítica, pode ser construído um precondicionador com mais elementos que A, acelerando desta forma a convergência. Por outro lado, se o espaço para armazenamento for limitado,

pode ser construído um precondicionador com menos elementos que A, que ao menos seja melhor que o escala diagonal. A FCC é descrita em detalhes na Seção 5.1.

4.2 Reordenamento

A eficiência de implementações que usam a fatoração de Cholesky para resolver sistemas de equações lineares com matrizes esparsas e de dimensões elevadas depende das técnicas de reordenamento que são usadas para preservar a esparsidade. O objetivo é encontrar uma matriz de permutação P que torne a fatoração de PAP^T a mais esparsa possível. O problema de encontrar a permutação ótima é NP-completo [92]. Entretanto, algumas heurísticas têm sido usadas com sucesso em implementações de métodos de pontos interiores.

No código HOPDM [45] são usadas duas heurísticas para os sistemas de equações normais, denominadas ordenamento de mínimo grau e preenchimento mínimo local [27]. Ambas são heurísticas locais e agem no sentido de determinar o pivô que cause o menor preenchimento na fatoração. A primeira delas usa uma função de custo para sobre-estimar o número de operações de ponto flutuante e a quantidade de preenchimento que será causada na escolha de um determinado pivô. A outra, analisa a quantidade exata de preenchimento na escolha do pivô, mas isto é feito a um custo elevado. Mészáros [71] propõe uma técnica eficiente para selecionar o pivô, na qual o número de candidatos é reduzido a um conjunto menor, antes de analisar o que causa o menor preenchimento. No código PCx [25] é utilizada uma implementação da heurística de mínimo grau múltiplo [60] para reduzir o preenchimento e conseqüentemente o trabalho requerido na fatoração de Cholesky.

Métodos de reordenamento também podem ser utilizados para trabalhar com abordagens iterativas, desde que o precondicionador seja contruído baseado em algum tipo de fatoração. Carmo [21] utilizou algumas dessas estratégias para trabalhar com o algoritmo gradiente conjugado precondicionado pela fatoração controlada de Cholesky [19]. Foram testados os métodos de contagem de colunas, Cuthill-Mckee reverso [39] e o método de mínimo grau aproximado [41]. O método de contagem de colunas é baseado na ordenação das colunas pelo número de elementos não nulos. Cuthill-Mckee reverso é um algoritmo bastante simples aplicado ao grafo da matriz, no qual os nós são numerados em ordem reversa utilizando busca em largura. Esta heurística tem por objetivo reduzir a largura de banda por mover todos

elementos não nulos para as proximidades da diagonal. O método de mínimo grau é mais elaborado e tem como idéia básica minimizar localmente o preenchimento. A cada iteração, o vértice escolhido para ser eliminado é o que possui o menor grau em um determinado grafo. Este método produz uma matriz com estrutura baseada em fractal, apresentando grandes blocos de zeros. Essas três estratégias foram testadas com matrizes esparsas pertencentes a coleção Harwell-Boeing [28]. Os resultados obtidos são promissores. A estrutura da matriz original tem mostrado influência sobre os métodos de reordenamento; o número de iterações do gradiente conjugado diminui consideravelmente em matrizes com estrutura irregular, por outro lado, em matrizes com estrutura banda ou próxima de banda, pouco ganho é obtido.

Em algoritmos de pontos interiores, as estratégias de reordenamento, geralmente são implementadas na fase inicial. O padrão de esparsidade da matriz não é alterado durante as iterações de Newton, já que as mudanças ocorrem apenas nos valores numéricos dos elementos da matriz Θ. Em função disso, as estratégias de reordenamento podem ser aplicadas uma única vez, sendo realizadas no início do processo.

4.3 Precondicionadores projetados para métodos de pontos interiores

Os sistemas de Newton oriundos em métodos de pontos interiores, freqüentemente, tornam-se malcondicionados à medida que as iterações prosseguem. Abordagens iterativas são promissoras desde que existam precondicionadores eficientes. Este estudo vem despertando o interesse da comunidade científica e muitos métodos iterativos precondicionados estão sendo propostos.

A primeira implementação de métodos de pontos interiores com abordagem iterativa para resolver os sistemas de equações de Newton foi apresentada por Adler, Karmarkar, Resende e Veiga [6]. Esta abordagem utiliza o método do gradiente conjugado precondicionado pela fatoração incompleta de Cholesky da matriz $A\Theta A^T$. Karmarkar e Ramakrishnan [56] também utilizaram o método do gradiente conjugado precondicionado para resolver o sistema de equações normais resultantes da aplicação do algoritmo de projeção na solução de sistemas lineares de grande porte. Em média, a cada três iterações foi calculada a fatoração incompleta de Cholesky da matriz $A\Theta A^T$ e utilizada como precondicionador nas iterações

subseqüentes. Mehrotra [66] utiliza o mesmo precondicionador para resolver os sistemas de equações normais em um algoritmo afim escala dual. A fatoração é calculada em todas as iterações. Resende e Veiga [80] utilizaram o método do gradiente conjugado precondicionado para resolver os sistemas lineares presentes na aplicação de um algoritmo afim escala dual especializado na solução de problemas de fluxo de custo mínimo em redes não capacitadas bipartidas. Foram usados dois precondicionadores; um diagonal e outro baseado na matriz de incidência de uma árvore geradora de peso máximo. Carpenter e Shanno [22] resolvem problemas de programação linear e quadrática por um algoritmo de pontos interiores que utiliza o método do gradiente conjugado com um precondicionador diagonal. Portugal, Veiga, Resende e Júdice [79], propuseram o método truncado primal-inviável dual-viável, focando a solução de problemas de fluxo em redes. Nas primeiras iterações, o precondicionador é construído com a diagonal da matriz $A\Theta A^T$ e depois substituído por um precondicionador árvore geradora. Wang e O'Leary [90] propõem o uso de um algoritmo adaptativo para determinar as direções de Newton presentes na aplicação de métodos de pontos interiores à problemas de programação linear. É utilizado um procedimento para decidir entre métodos diretos ou iterativos. Quando métodos iterativos são escolhidos é possível determinar se o precondicionador deve ser atualizado ou reinicializado. Júdice *et al* [54] fizeram um estudo comparativo de uma classe de precondicionadores utilizados na solução de problemas de redes por métodos de pontos interiores. Mais recentemente, Frangioni e Gentile [35] propuseram uma classe de precondicionadores para resolver sistemas com matrizes simétricas originadas em problemas de fluxo de custo mínimo. Estes precondicionadores são construídos baseados na idéia de extrair da matriz de incidência um subgrafo que contenha uma árvore geradora, para a qual é possível determinar a fatoração de Cholesky com preenchimento mínimo. Todos os trabalhos citados resolvem o sistema de equações normais.

Gill, Murray, Ponceléon e Saunders [42] desenvolveram uma série de precondicionadores para tratar com matrizes simétricas indefinidas originadas em problemas de otimização. Os problemas são resolvidos utilizando SYMMLQ com a adição de precondicionadores definidos positivos. A seguir, são descritos quatro desses precondicionadores.

Considere $Kx = z$ o sistema linear (3.2). Seja $M = CC^T$ o precondicionador requerido pelo SYMMLQ. A solução é obtida resolvendo implicitamente o sistema

$$C^{-1}KC^{-T}y = C^{-1}z \ , \ \text{ sendo } \ y = C^T x \ .$$

Os precondicionadores foram desenvolvidos baseados em partições das matrizes A e Θ, a

fim de evitar o malcondicionamento ocorrido pelas grandes variações nos elementos de Θ conforme as iterações do método primal-dual prosseguem.

Sejam $A = (A_N, A_B)$ e $\Theta = \text{diag}(\Theta_N, \Theta_B)$. O símbolo N relaciona as variáveis que estão próximas ao limite, enquanto B representa as restantes. A partição ocorre levando em consideração que A_B deve possuir posto completo e ser quadrada. O primeiro precondicionador foi projetado para eliminar as maiores entradas em Θ^{-1},

$$M_1 = C_1 C_1^T \; ; \qquad C_1 = \begin{pmatrix} \Theta_N^{-\frac{1}{2}} & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \; , \tag{4.1}$$

$$C_1^{-1} K C_1^{-T} = \begin{pmatrix} I & 0 & (A_N \Theta_N^{\frac{1}{2}})^T \\ 0 & \Theta_B & A_B^T \\ (A_N \Theta_N^{\frac{1}{2}}) & A_B & 0 \end{pmatrix} \; ,$$

Θ_N^{-1} representa os $(n-m)$ maiores elementos em Θ^{-1}, sendo n e m o número de variáveis e restrições do problema. O segundo é um precondicionador bloco diagonal,

$$M_2 = C_2 C_2^T \; ; \qquad C_2 = \begin{pmatrix} \Theta_N^{-\frac{1}{2}} & 0 & 0 \\ 0 & A_B^T & 0 \\ 0 & 0 & I \end{pmatrix} \; , \tag{4.2}$$

$$C_2^{-1} K C_2^{-T} = \begin{pmatrix} I & 0 & (A_N \Theta_N^{\frac{1}{2}})^T \\ 0 & A_B^{-1} \Theta_B^{-1} A_B^{-1} & I \\ (A_N \Theta_N^{\frac{1}{2}}) & I & 0 \end{pmatrix} \; .$$

A solução do sistema com a matriz precondicionada envolve alguma fatoração esparsa da matriz A_B. Os dois precondicionadores restantes foram desenvolvidos para tratar com problemas degenerados,

$$M_3 = C_3 C_3^T \; ; \qquad C_3 = \begin{pmatrix} \Theta_N^{-\frac{1}{2}} & 0 & 0 \\ 0 & A_B^T & \frac{1}{2}\Theta_B^{-1} A_B^{-1} \\ 0 & 0 & I \end{pmatrix} \; , \tag{4.3}$$

$$C_3^{-1} K C_3^{-T} = \begin{pmatrix} I & -\frac{1}{2}(A_N \Theta_N^{\frac{1}{2}})^T (A_B^{-T} \Theta_B^{-1} A_B^{-1}) & (A_N \Theta_N^{\frac{1}{2}})^T \\ -\frac{1}{2}(A_B^{-T} \Theta_B^{-1} A_B^{-1})(A_N \Theta_N^{\frac{1}{2}}) & 0 & I \\ (A_N \Theta_N^{\frac{1}{2}}) & I & 0 \end{pmatrix} .$$

Em geral, conforme as iterações evoluem, as parcelas $(A_N\Theta_N^{\frac{1}{2}})$ e $(A_B^{-T}\Theta_B^{-1}A_B^{-1})$ tornam-se pequenas. O precondicionador M_4 tem a seguinte forma:

$$M_4 = C_4C_4^T\ ; \qquad C_4 = \begin{pmatrix} \Theta_N^{-\frac{1}{2}} & 0 & 0 \\ 0 & L & \frac{1}{2}\Theta_B^{-1}L^{-T} \\ 0 & 0 & U^T \end{pmatrix}\ , \ \text{sendo}\ \ A_B^T = LU\ ; \tag{4.4}$$

$$C_4^{-1}KC_4^{-T} = \begin{pmatrix} I & -\frac{1}{2}(UA_N\Theta_N^{\frac{1}{2}})^T(L^{-1}\Theta_N^{-1}L^{-T}) & (U^{-T}A_N\Theta_N^{\frac{1}{2}})^T \\ -\frac{1}{2}(L^{-1}\Theta_N^{-1}L^{-T})(U^{-T}A_N\Theta_N^{\frac{1}{2}}) & 0 & I \\ (U^{-T}A_N\Theta_N^{\frac{1}{2}}) & I & 0 \end{pmatrix}.$$

Baseado nestas idéias, Oliveira [73] propôs dois precondicionadores para tratar com problemas degenerados. O autor considera muito rígido eliminar um número fixo de variáveis com o precondicionador C_1 e propõe,

$$\tilde{C}_1 = \begin{pmatrix} \Theta_{\tilde{N}}^{-\frac{1}{2}} & 0 & 0 \\ 0 & \Theta_{\tilde{B}}^{-\frac{1}{2}} & 0 \\ 0 & 0 & 0 \end{pmatrix}\ , \tag{4.5}$$

tal que, $\Theta_{\tilde{N}}^{-\frac{1}{2}}$ é formado pelos elementos em Θ^{-1} maiores que $10^{-8}\bar{\delta}$, sendo $\bar{\delta}$ a maior entrada de Θ^{-1}. Os elementos restantes de Θ^{-1} são fixados para $10^{-8}\bar{\delta}$ e armazenados em $\Theta_{\tilde{B}}^{-\frac{1}{2}}$.

O outro precondicionador é baseado na fatoração incompleta de Cholesky da matriz $A\Theta A^T$,

$$\tilde{C}_4 = \begin{pmatrix} \Theta_{\tilde{N}}^{-\frac{1}{2}} & 0 & 0 \\ 0 & \Theta_{\tilde{B}}^{-\frac{1}{2}} & 0 \\ 0 & 0 & \tilde{L} \end{pmatrix}\ , \ \text{sendo}\ \ A\Theta A^T = \tilde{L}\tilde{L}^T. \tag{4.6}$$

Os resultados computacionais obtidos com MINRES precondicionado por $\tilde{C}_4$ foram superiores em relação a C_4.

Durrazzi e Ruggiero [31] resolvem o sistema aumentado originado na solução de problemas de programação quadrática utilizando gradiente conjugado. Dois precondicionadores para os sistemas indefinidos são projetados e as propriedades teóricas do sistema precondicionado são analisadas. Uma nova classe de precondicionadores para os sistemas aumentados originados em problemas de programação linear e não lineares é investigada por Keller *et al.* [57]. Haws e Meyer [50] propõem três precondicionadores para o sistemas aumentado. Os dois

primeiros aplicam fatorações incompletas no sistema de equações normais. O outro é baseado na fatoração incompleta LU da matriz indefinida, após ser realizado um pré-processamento.

Mais recentemente, Bergamaschi, Gondzio e Zilli [15] desenvolveram precondicionadores para resolver os sistemas indefinidos originados em problemas de programação quadrática e não linear, ou seja

$$\begin{bmatrix} -Q - \Theta_1^{-1} & A^T \\ A & \Theta_2^{-1} \end{bmatrix} \begin{bmatrix} \triangle x \\ \triangle y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

sendo $\Theta_1 \in \mathbb{R}^{n \times n}$ e $\Theta_2 \in \mathbb{R}^{m \times m}$ matrizes diagonais com elementos positivos, definidas de acordo com o problema que está sendo resolvido. Para facilitar a notação os resultados foram apresentados para programação quadrática com restrições de igualdade ($\Theta_2 = 0$), e depois extendidos ao caso não linear.

O precondicionador foi projetado eliminando todos os elementos fora da diagonal de Q,

$$P = \begin{bmatrix} E & A^T \\ A & 0 \end{bmatrix}, \quad E = -\text{diag}(Q) - \Theta_1^{-1} ;$$

e calculado, por duas formas distintas:

$$P_1 = \begin{bmatrix} E & A^T \\ A & 0 \end{bmatrix} = LDL^T \quad \text{e} \tag{4.7}$$

$$P_2 = \begin{bmatrix} E & A^T \\ A & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ AE^{-1} & L_0 \end{bmatrix} \begin{bmatrix} E & 0 \\ 0 & -D_0 \end{bmatrix} \begin{bmatrix} I & E^{-1}A^T \\ 0 & L_0^T \end{bmatrix}, \tag{4.8}$$

sendo $AE^{-1}A^T = L_0 D_0 L_0^T$.

Estes precondicionadores são matematicamente equivalentes e diferem apenas na forma como são implementados. Eles foram desenvolvidos com o objetivo de tornar a fatoração mais esparsa, sem perder as propriedades numéricas do problema. Note que, esta estratégia não é interessante para programação linear, pois o custo da fatoração será o mesmo que em abordagens diretas. Os autores mostraram que em problemas de programação convexa, a matriz precondicionada possui todos os autovalores positivos. Foram realizados testes com os métodos do Gradiente Conjugado [51], BiCGSTAB [87], GMRES [84] e QMR [38].

Muitos dos precondicionadores propostos para os sistemas indefinidos, originados em métodos de pontos interiores, foram desenvolvidos para resolver problemas de programação

quadrática e não linear. Normalmente, esses precondicionadores não são eficientes na solução de problemas de programação linear de grande porte. Entretanto, Oliveira [73] propôs uma classe de precondicionadores, denominada precondicionador separador, especialmente projetada para os sistemas indefinidos originados em problemas de programação linear. Uma característica importante dessa classe é a opção de reduzir o sistema precondicionado indefinido a um definido positivo, similar as equações normais, permitindo o uso do gradiente conjugado. O precondicionador separador apresenta melhores resultados nas proximidades da solução ótima, quando os sistemas já são muito malcondicionados. A Tabela 4.1 ilustra o comportamento do gradiente conjugado precondicionado com a fatoração incompleta de Cholesky e com precondicionador separador, na solução do problema KEN13 presente na NETLIB.

É interessante observar que o precondicionador com a fatoração incompleta apresenta melhores resultados nas primeiras iterações e se deteriora nas últimas, quando o sistema se torna altamente malcondicionado. O precondicionador separador tem comportamento inverso; as últimas iterações são as que apresentam o melhor desempenho.

Baseado nestes fatos, propomos o desenvolvimento de um precondicionador híbrido para resolver os sistemas lineares originados em problemas de programação linear. Este precondicionador será descrito no próximo capítulo.

Tabela 4.1: KEN13: Iterações de método gradiente conjugado. [74, Tabela 1, pg20].

	Iterações internas	
iterações PI	Cholesky Incompleto	Precondicionador separador
0	49	195
1	49	203
2	45	258
3	39	190
4	24	171
5	24	185
6	20	128
7	22	130
8	22	133
9	32	126
10	44	108
11	71	91
12	104	92
13	171	76
14	323	63
15	480	52
16	834	43
17	1433	34
18	2146	30
19	4070	22
20	7274	18
21	11739	17
22	15658	15
23	24102	12
24	13463	10
25	5126	6
Média	3360	84

Capítulo 5

Precondicionador híbrido para problemas de programação linear

Muitas abordagens iterativas têm sido usadas para resolver os sistemas de equações de Newton presentes em métodos de pontos interiores. Muitas delas constroem seus precondicionadores baseados em fatorações incompletas da matriz $A\Theta A^T$. Em geral, a matriz Θ se altera rapidamente durante as iterações de pontos interiores e torna-se altamente malcondicionada nas iterações finais. Por esta razão é difícil encontrar uma estratégia de precondicionamento para os métodos iterativos que produza bom desempenho durante todas as iterações. Precondicionadores baseados em fatorações incompletas mostram-se eficientes nas primeiras iterações e se deterioram nas últimas. Por outro lado, o precondicionador separador apresenta melhores resultados nas iterações finais.

Neste capítulo, estamos propondo aplicar o método do gradiente conjugado para resolver o sistema de equações normais (3.3) utilizando um precondicionador híbrido M [16],

$$M^{-1}(A\Theta A^T)M^{-T}\bar{y} = M^{-1}(A\Theta(r_d - X^{-1}r_a) + r_p), \tag{5.1}$$

sendo $\bar{y} = M^T \triangle y$. Nossa abordagem assume a existência de duas fases durante as iterações de pontos interiores, que são definidas como fase I e fase II. Na primeira delas o precondicionador M é construído usando a fatoração controlada de Cholesky, que será descrita na Seção 5.1. Após a mudança de fases ser identificada, M será contruída pelo precondicionador separador, o qual será apresentado na Seção 5.2.

5.1 Fatoração controlada de Cholesky

A fatoração controlada de Cholesky (FCC) foi proposta, em 1995, por Campos [19]. Ela é uma variação da fatoração incompleta, cujo objetivo principal está em construir uma matriz precondicionadora que possua os autovalores agrupados e próximos a unidade, de forma a acelerar a convergência do método do gradiente conjugado. A FCC foi usada com sucesso na solução de sistemas lineares originados em problemas de equações diferenciais parciais com dependência do tempo [20]. Apesar de ter sido desenvolvida para aplicações gerais, este tipo de fatoração tem algumas propriedades desejáveis do ponto de vista de otimização, tais como, armazenamento previsível e controle do preenchimento de acordo com a memória disponível. No entanto, ainda não tinha sido usada nesse contexto.

Considere a matriz $A\Theta A^T \in \mathbb{R}^{n\times n}$ fatorada em:

$$A\Theta A^T = LL^T = \tilde{L}\tilde{L}^T + R \tag{5.2}$$

sendo:
L = fator obtido pela fatoração completa de Cholesky;
$\tilde{L}$ = fator obtido pela fatoração incompleta de Cholesky;
R = matriz resto.

Usando $\tilde{L}$ como uma matriz precondicionadora para $A\Theta A^T$, obtém-se

$$\tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} = (\tilde{L}^{-1}L)(L^T\tilde{L}^{-T}) = (\tilde{L}^{-1}L)(\tilde{L}^{-1}L)^T \ .$$

Definindo, $E = L - \tilde{L}$ e substituindo na igualdade acima, tem-se:

$$\tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} = (I + \tilde{L}^{-1}E)(I + \tilde{L}^{-1}E)^T \ .$$

Observe que, quando $\tilde{L} \to L$ então $E \to 0$ e $\tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} \to I$.

Duff e Meurant [29] mostraram que o número de iterações necessárias para convergência pelo método do gradiente conjugado está diretamente relacionado com a norma de R, sendo R expresso por:

$$R = LE^T + E\tilde{L}^T \ .$$

A FCC é construída baseada na minimização da norma de Frobenius de E, visto que, quando $\|E\| \to 0 \Rightarrow \|R\| \to 0$. Para tal, considere o seguinte problema

$$\min \|E\|_F^2 = \sum_{j=1}^{n} c_j, \ \text{ sendo } \ c_j = \sum_{i=1}^{n} |l_{ij} - \tilde{l_{ij}}|^2 \ . \tag{5.3}$$

Dividindo c_j em dois somatórios, tem-se

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

sendo

m_j: número de elementos não nulos abaixo da diagonal na j-ésima coluna da matriz $A\Theta A^T$;

η: número extra de elementos não nulos permitido por coluna;

n: ordem da matriz.

Note que, no primeiro somatório estão os $m_j+\eta$ elementos não nulos da j-ésima coluna de $\tilde{L}$. O segundo, contém apenas os elementos do fator completo L que não estão em $\tilde{L}$. Baseado nesses fatos, o problema (5.3) pode ser resolvido, utilizando a seguinte heurística

- Aumentando o fator η, permitindo maior preenchimento. Desta forma, c_j decresce, pois o primeiro somatório contém mais elementos. Isto é semelhante a minimização não seletiva que ocorre quando níveis de preenchimento são usados nas fatorações incompletas.

- Escolhendo os maiores $m_j + \eta$ elementos de $\tilde{L}$, em valor absoluto, para η fixo. Desta forma os maiores elementos estão no primeiro somatório e os menores no segundo, produzindo um fator $\tilde{L}$ ótimo, para uma determinada quantidade de armazenamento. Este é um esquema de minimização seletiva, semelhante ao de retirada por tolerância.

Uma coluna do precondicionador é calculada por vez, sendo armazenados os maiores elementos em valor absoluto. Dessa forma, uma melhor aproximação para o fator completo é obtida com a quantidade de memória disponível. As principais características desse precondicionador são descritas a seguir.

Escolha de elementos por valor - A FCC nunca considera o padrão de esparsidade da matriz original. Os elementos armazenados no precondicionador podem ou não estar nas posições correspondentes aos da matriz original, já que a escolha é feita por valor e não por posição. Este fato, produz melhores precondicionadores comparados aos que conservam o padrão de esparsidade.

Generalização do ICD (*Improved Incomplete Cholesky Decomposition*)- Jones e Plassmann [53] desenvolveram uma classe de precondicionadores na qual o padrão de esparsidade da matriz

original é ignorado. Naquela abordagem é escolhido um número fixo de elementos não nulos em cada linha ou coluna do precondicionador. Este número é o mesmo da matriz original, mas os elementos armazenados são os maiores em valor absoluto. Dessa forma, a FCC pode ser vista como uma generalização do IICD, uma vez que, o número de elementos não nulos em cada linha ou coluna do precondicionador pode variar.

Incremento exponencial para evitar falhas na fatoração incompleta - Manteufell [64] provou que se a matriz dos coeficientes V for simétrica e definida positiva então existe uma constante $\alpha > 0$, tal que a fatoração incompleta de $V + \alpha I$ existe. Uma das dificuldades ao utilizar esse critério é determinar o menor valor de α que garanta a existência da fatoração incompleta. Jones e Plassmann [53] utilizam um incremento linear. Para que se tenha uma menor perturbação na diagonal a FCC utiliza um incremento exponencial dado por $\alpha_i = 5.10^{-4}2^{i-1}$.

Precondicionador versátil - Esta classe de precondicionadores pode ser construída levando em consideração a disponibilidade de memória. O parâmetro η pode variar de $-n$ a n, sendo n a ordem da matriz. Se $\eta = -n$ será feito apenas o escalonamento diagonal na matriz, para que ela possua diagonal unitária. Para $\eta = 0$, o fator L requer o mesmo espaço de armazenamento da matriz A, mas o padrão de esparsidade normalmente não será o mesmo, já que o critério de seleção é baseado nos valores numéricos dos elementos e não em suas posições. Quando $\eta = n$, ter-se-á a fatoração de Cholesky completa. Conforme η aumenta a matriz precondicionada tende para a identidade, conseqüentemente, o número de iterações necessárias para determinar a solução pelo método do gradiente conjugado é reduzido, mas o tempo de precondicionamento aumenta. Uma boa escolha de η garante um menor tempo para a solução do sistema.

Armazenamento previsível - Em fatorações incompletas de Cholesky o preenchimento é permitido pela adição de níveis. ICD(0), denota a fatoração incompleta de Cholesky com nível zero de preenchimento, ou seja, nenhum preenchimento é permitido. Quando níveis de preenchimento são usados, não é possível, sem uma fatoração simbólica, prever a quantidade de memória que será necessária para armazenar o precondicionador. Entretanto, como mostra a Tabela 5.1 na FCC o armazenamento é previsível, uma vez que são armazenadas apenas os $m_j + \eta$ maiores elementos no precondicionador.

Embora a FCC(η) necessite maior quantidade de armazenamento quando comparada a ICD(0), ela apresenta uma estrutura de dados mais versátil que permite que o número

Tabela 5.1: Quantidade de memória usada na ICD(0) e na FCC(η).
n: ordem de matriz $A\Theta A^T$, m: número de elementos não nulos em $A\Theta A^T$
η: fator que controla o preenchimento,
i, r: número de bytes para variáveis inteiras e reais.

matrizes	número de bytes	para i=4 and r=8
$A\Theta A^T$	$(i+r)m+in+i$	$12m+4n+4$
$A\Theta A^T+\tilde{L}_{ICD(0)}$	$(i+2r)m+in+i$	$20m+4n+4$
$A\Theta A^T+\tilde{L}_{CCF(\eta)}$	$(3i+2r)m+((2i+r)\eta+2i)n+2i$	$28m+(16\eta+8)n+8$

de elementos em $\tilde{L}$ seja variável.

A Figura 5.1 apresenta uma versão do algoritmo FCC dado em [20]. Inicialmente, a coluna j de $\tilde{L}$ é calculada, depois selecionam-se os $m_j+\eta$ maiores elementos em valor absoluto e descarta-se o restante, sendo m_j o número de elementos não nulos da coluna j de $A\Theta A^T$. Os parâmetros de entrada para o algoritmo são a matriz $\mathsf{A}=A\Theta A^T$, a ordem n da matriz e o fator η para controle de preenchimento. Tem-se como parâmetro de saída, a matriz precondicionadora $\mathsf{L}=\tilde{L}$.

Matrizes precondicionadas por fatorações incompletas podem perder a propriedade de serem definidas positivas, ocorrendo presença de elementos muito pequenos ou imaginários na diagonal da matriz $\tilde{L}$. Entretanto, muitas estratégias para evitar este problema foram desenvolvidas. Na FCC, esse processo é realizado utilizando o incremento exponencial, para tal, admite-se que a matriz tenha diagonal unitária. Note que, antes da fatoração ser feita é aplicado um escalonamento diagonal na matriz.

Algoritmo FCC

{ Objetivo: Construir uma matriz precondicionadora para A utilizando a FCC }

parâmetros de entrada $n,\ A,\ \eta$

{ ordem, matriz a ser decomposta, controle de preenchimento }

parâmetros de saída L { matriz precondicionadora }

$D \leftarrow$ diagonal (A)

$A \leftarrow D^{-1/2} A D^{-1/2}$

$\sigma \leftarrow 0\ ;\ l \leftarrow 0$

repita

falha $\leftarrow$ falso

para $j \leftarrow 1$ até n faça

$t_k \leftarrow l_{j,k} d_k\ ,\ 1 \leq k \leq j-1$

$w_i \leftarrow l_{j,k} d_j \leftarrow a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} t_k\ ,\ j+1 \leq i \leq n$

$m_j \leftarrow$ número de elementos não nulos nesta coluna de A

$w \leftarrow$ seleção dos $m_j + \eta$ elementos de w

$d_j \leftarrow 1 - \sum_{k=1}^{j-1} l_{j,k} t_k + \sigma$

se $d_j \leq \epsilon_m$ então

$l \leftarrow l + 1\ ;\ \sigma \leftarrow 5.10^{-4}.2^l$

falha $\leftarrow$ verdadeiro

interrompa

fim se

para $k \leftarrow 1$ até $m_j + \eta$ faça

$i_k \leftarrow$ índice da linha selecionada

$l_{i_k} j \leftarrow w_{i_k} / d_j$

fim para

fim para

se (não falha) ou $l > 25$

então interrompa

fim se

fim repita

fim algoritmo

Figura 5.1: Fatoração controlada de Cholesky.

5.2 Precondicionador separador

Com o objetivo de evitar o uso de $A\Theta A^T$ na construção do precondicionador, Oliveira desenvolveu uma nova classe de precondicionadores para os sistemas aumentados [73]. Esse precondicionador pode ser visto como a generalização do precondicionador proposto por Resende e Veiga no contexto de problemas de custo mínimo em redes [80]. A principal característica dessa classe de precondicionadores é que ela apresenta melhores resultados nas proximidades da solução ótima, quando os sistemas lineares já são muito malcondicionados. Entretanto, esse precondicionador falha em obter convergência nas primeiras iterações de algumas classes de problemas. Foram obtidos bons resultados usando um precondicionador diagonal nas primeiras iterações [74].

Considere o seguinte precondicionador em blocos para o sistema indefinido (3.2),

$$M^{-1} = \begin{pmatrix} F & G \\ H & J \end{pmatrix} .$$

A matriz de (3.2) precondicionada por M^{-1}, tem a seguinte forma,

$$\begin{pmatrix} -F\Theta^{-1}F^T + FA^TG^T + GAF^T & -F\Theta^{-1}H^T + FA^TJ^T + GAH^T \\ -H\Theta^{-1}F^T + HA^TG^T + JAF^T & -H\Theta^{-1}H^T + HA^TJ^T + JAH^T \end{pmatrix} .$$

Os blocos F, G, H e J foram construídos com o objetivo principal de evitar a presença de $A\Theta A^T$ na matriz precondicionada. Primeiro, é feito $J = 0$ e a matriz se reduz a,

$$\begin{pmatrix} -F\Theta^{-1}F^T + FA^TG^T + GAF^T & -F\Theta^{-1}H^T + GAH^T \\ -H\Theta^{-1}F^T + HA^TG^T & -H\Theta^{-1}H^T \end{pmatrix} .$$

Para que os blocos fora da diagonal sejam nulos, é atribuido $G^T = (HA^T)^{-1}H\Theta^{-1}F^T$,

$$\begin{pmatrix} -F\Theta^{-1}F^T + FA^TG^T + GAF^T & 0 \\ 0 & -H\Theta^{-1}H^T \end{pmatrix} .$$

Fazendo $H = [I\ 0]P$, onde P é uma matriz de permutação tal que HA^T é não singular, o bloco $H\Theta^{-1}H^T$ é equivalente a uma matriz diagonal, denotada por Θ_B^{-1}, sendo,

$$P\Theta^{-1}P^T = \begin{pmatrix} \Theta_N^{-1} & 0 \\ 0 & \Theta_B^{-1} \end{pmatrix} .$$

Por último, atribui-se $F = \Theta^{\frac{1}{2}}$.

A matriz precondicionada por M^{-1} é dada por,

$$M^{-1}\begin{pmatrix} \Theta^{-1} & A^T \\ A & 0 \end{pmatrix} M^{-T} = \begin{pmatrix} -I + \Theta^{\frac{1}{2}} A^T G^T + GA\Theta^{\frac{1}{2}} & 0 \\ 0 & -\Theta_B^{-1} \end{pmatrix}, \tag{5.4}$$

sendo,

$$M^{-1} = \begin{pmatrix} \Theta^{\frac{1}{2}} & G \\ H & 0 \end{pmatrix}, \;\; G = H^T \Theta_B^{-\frac{1}{2}} B^{-1}, \;\; HP = [I\ 0] \;\text{ e }\; AP^T = [B\ N].$$

O sistema aumentado possui dimensão $n+m$, sendo n o número de variáveis e m o número de restrições do problema. Além do uso das equações normais ser evitado, pode-se considerar que o precondicionador reduz a dimensão do sistema aumentado para n, uma vez que a parte inferior é formada apenas pela matriz diagonal Θ_B^{-1} e, portanto facilmente resolvida.

O sistema de dimensão n envolve a matriz indefinida,

$$-I + \Theta^{\frac{1}{2}} A^T G^T + GA\Theta^{\frac{1}{2}}. \tag{5.5}$$

Pelo Lema 5.1 ([73, Lema 5.1]), esta matriz pode ser escrita na forma,

$$K = -I_n + U^T V^T + VU, \tag{5.6}$$

sendo $UV = V^T U^T = I_m$ e $U, V^T \in \mathbb{R}^{m\times n}$.

Matrizes deste tipo apresentam propriedades teóricas interessantes.

Lema 5.1. *Seja $A = [B; N]$, com B não singular e $G = H^T \Theta_B^{-\frac{1}{2}} B^{-1}$, sendo $H = [I; 0]$. Então $G^T \Theta^{\frac{1}{2}} A^T = A\Theta^{\frac{1}{2}} G = I$.*

As principais características do precondicionador separador são descritas a seguir:

Redução a um sistema definido positivo: a matriz (5.5) pode ser transformada em

$$I_m + \Theta_B^{-\frac{1}{2}} B^{-1} N \Theta_N N^T B^{-T} \Theta_B^{-\frac{1}{2}} \tag{5.7}$$

ou

$$I_{n-m} + \Theta_N^{\frac{1}{2}} N^T B^{-T} \Theta_B^{-1} B^{-1} N \Theta_N^{\frac{1}{2}}\ . \tag{5.8}$$

Essas matrizes tem dimensões m e $n-m$, respectivamente, são simétricas, definidas positivas e apresentam propriedades importantes relacionadas com a matriz indefinida K. Uma delas

é que todos os autovalores são maiores ou iguais a um. Essa característica é interessante, já que autovalores muito próximos de zero, geralmente, prejudicam o desempenho de métodos iterativos na solução dos sistemas lineares. Resultados numéricos mostram que o gradiente conjugado apresenta desempenho superior na solução do sistema envolvendo a matriz definida positiva (5.7), quando comparado ao SYMMLQ, para resolver (5.5).

Equivalência às equações normais: a matriz (5.7) pode ser obtida utilizando o sistema de equações normais. Considere $A = [B\ N]P$ sendo P uma matriz de permutação tal que B é não singular, então,

$$A\Theta A^T = B\Theta_B B^T + N\Theta_N N^T\ . \tag{5.9}$$

Multiplicando a equação (5.9) por $\Theta_B^{-\frac{1}{2}} B^{-1}$ e pós multiplicando por $(\Theta_B^{-\frac{1}{2}} B^{-1})^T$ tem-se:

$$T = \Theta_B^{-\frac{1}{2}} B^{-1}(A\Theta A^t)B^{-T}\Theta_B^{-\frac{1}{2}} = I + WW^T, \tag{5.10}$$

sendo $W^T = \Theta_B^{-\frac{1}{2}} B^{-1} N \Theta_N^{\frac{1}{2}}$.

Note que, perto da solução ao menos $n - m$ elementos em Θ^{-1} são grandes. Dessa forma, uma escolha adequada das colunas de B, faz com que os elementos em Θ_B^{-1} e Θ_N sejam muito pequenos perto da solução. Neste caso, W aproxima-se da matriz nula, T se aproxima da matriz identidade e os maiores autovalores de T e $\kappa_2(T)$ se aproximam da unidade.

Estratégia para determinar B: o preço pago por evitar o uso do sistema de equações normais é determinar quais colunas de A devem ser escolhidas para formar B e resolver um sistema com esta matriz. Entretanto, a fatoração $QB = LU$ é tipicamente mais fácil de calcular do que a fatoração de Cholesky de $A\Theta A^T$. A forma como as colunas de A devem ser escolhidas para formar B, tem influência no precondicionador. A estratégia para formar B é minimizar $\|W\|$. Perto da solução a matriz precondicionada se aproxima da identidade, já que os elementos em Θ_B^{-1} e Θ_N são muito pequenos. Para isso, foi utilizada uma heurística que seleciona as m colunas linearmente independentes em $A\Theta$ com menores normas.

Fatoração LU: como o conjunto de colunas linearmente independentes não é conhecido no início do processo, a estratégia mais econômica para esta aplicação é trabalhar com uma coluna por vez. Neste caso, quando uma coluna linearmente dependente aparece ela é eliminada da fatoração e o método continua com a próxima coluna dada pela heurística. Outro fator importante é o controle do preenchimento excessivo que pode ocorrer durante a fatoração. O precondicionador separador utiliza uma técnica que consiste em interromper a

fatoração quando o número de elementos não nulos for superior ao de $A\Theta A^T$, e reordenar as colunas encontradas para obter uma fatoração mais esparsa. Em seguida, a fatoração é reiniciada e o processo se repete até que as m colunas linearmente independentes sejam encontradas. Outra fatoração LU é calculada, sendo geralmente muito mais esparsa que a anterior, principalmente, em problemas de maior dimensão. Assim, o trabalho de calcular uma nova decomposição é compensado na solução do sistema linear pelo método iterativo.

Conjunto de colunas: o mesmo conjunto de colunas linearmente independentes pode ser usado durante algumas iterações para construir o precondicionador. Desse modo, essas iterações são mais rápidas, já que a maior parte do tempo está em escolher as colunas e calcular a fatoração LU. Como a matriz Θ^{-1} é alterada durante todas as iterações o precondicionador não será o mesmo.

Uma cuidadosa implementação do precondicionador separador apresentou desempenho superior, quando comparada ao método direto na solução de algumas classes de problemas de programação linear de grande porte. Nessa implementação é utilizado um precondicionador diagonal nas primeiras iterações. Todavia, em alguns problemas, mesmo de pequeno porte, o precondicionador diagonal perde eficiência muito antes que o precondicionador separador esteja em condições de apresentar bom desempenho. Nestes casos, a convergência pode ser muito lenta ou até mesmo não ser obtida.

A fatoração controlada de Cholesky é utilizada para melhorar o desempenho e a robustez deste precondicionador.

Capítulo 6

Experimentos numéricos

Neste capítulo são apresentados os experimentos numéricos realizados com a fatoração controlada de Cholesky e o precondicionador híbrido na solução de problemas de programação linear. Como o objetivo principal do trabalho está focado na solução dos sistemas lineares originados em problemas de programação linear, utilizamos o código PCx, descrito a seguir.

6.1 O código PCx

O código PCx [25] foi desenvolvido no *Optimization Technology Center at Argonne National Laboratory and Northwestern University* e implementa o algoritmo primal-dual preditor-corretor [67] com múltiplas correções [46]. Esta abordagem é considerada ser a mais eficiente das variantes de pontos interiores. As rotinas estão implementadas na linguagem C, exceto a parte responsável pela solução dos sistemas lineares, que é feita utilizando uma biblioteca para fatoração esparsa de Cholesky, desenvolvida por Ng e Peyton [72] em FORTRAN77.

O PCx trabalha com problemas de programação linear no formato MPS. Os problemas podem conter restrições de igualdade e desigualdade, variáveis livres e limitadas. Uma rotina converte os modelos em um formato padrão e após a solução ser encontrada é transformada em termos da formulação original.

Para simplificar a notação, optamos por apresentar o algoritmo primal-dual no Capítulo 2 sem considerar variáveis com limite superior. No entanto, para descrever o método imple-

mentado no código PCx foi utilizado o seguinte modelo primal,

$$\begin{aligned} \min \quad & c^T x \\ \text{s.a.} \quad & Ax = b \\ & 0 \le x_i \le u_i, \quad i \in U \\ & x_i \ge 0 \qquad i \in \bar{U} \end{aligned} \tag{6.1}$$

sendo $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ e $\bar{U}$ o conjunto de índices das variáveis não limitadas superiormente. Associado ao modelo primal, tem-se o dual,

$$\begin{aligned} \max \quad & b^T y - \sum_{i \in U} u_i r_i \\ \text{s.a.} \quad & A_i^T y + z_i - r_i = c_i \quad i \in U \\ & A_i^T y + z_i = c_i \qquad i \in \bar{U} \\ & (r, z) \ge 0, \end{aligned} \tag{6.2}$$

em que, y, $z \in \mathbb{R}^n$ e r é um vetor de dimensão $|U|$.

Critério de Parada - O algoritmo termina em um dos quatro estágios descritos a seguir. A notação $\|(x, y)\|$ denota a norma-2 do vetor $[x\ y]$.

- Estágio ótimo: a solução é considerada ótima quando satisfizer

$$\frac{\|(r_b, r_u)\|}{1 + \|(b^T, u^T)\|} \le \text{prifeastol} \tag{6.3}$$

$$\frac{\|r_c\|}{1 + \|c\|} \le \text{dualfeastol} \tag{6.4}$$

$$\frac{|c^T x - (b^T y - \sum u_i r_i)|}{1 + |c^T x|} \le \text{optol} \tag{6.5}$$

sendo,

prifeastol=10^{-8}, dualfeastol=10^{-8} e optol=10^{-8} as tolerâncias;

$r_b = Ax - b$ e $r_c = A^T y + z - r - c$ os resíduos para inviabilidade primal e dual;

$r_u = x + w - u$ o resíduo para as restrições de limite superior.

No código PCx, que nos foi disponibilizado, o critério (6.5) está substituído por,

$$\frac{\mu}{1 + |c^T x|} \le \text{optol}, \ \text{ sendo } \ \mu = \frac{x^T z + (u - x)^T r}{n}, \tag{6.6}$$

mas em nossos experimentos decidimos utilizar o critério original (6.5), por ser mais rigoroso.

- Estágio inviável: pode ser detectado utilizando a função de mérito ϕ, definida por

$$\phi(y,x,s,w,s) = \frac{\|(r_b, r_u)\|}{\max(1, \|(b,u)\|)} + \frac{\|r_c\|}{\max(1,\|c\|)} + \frac{\left|c^T x - (b^T y - \sum u_i r_i)\right|}{\max(1, \|(b,u)\|, \|c\|)}.$$

  Em geral, para problemas em que uma solução primal-dual existe ϕ decresce para zero após oscilar nas primeiras iterações. Por outro lado, em problemas inviáveis ϕ pode sofrer um acréscimo considerável. A inviabilidade do problema é detectada quando,

$$\phi(y_k, x_k, s_k, w_k, s_k) \geq \max(10^{-8}, 10^5 \phi_{\min}[k]),$$

  sendo $\phi_{\min}[k]$ o menor valor de ϕ encontrado até a k-ésima iteração.

- Estágio desconhecido: pode ocorrer em duas situações. A primeira delas identifica convergência lenta, com o teste

$$\phi_{\min}[k-30] \geq \frac{1}{2}\phi_{\min}[k] \;\text{ e }\; k \geq 30.$$

  A outra condição ocorre quando μ é reduzido muito mais rápido que $\|(r_b, r_u)\|$ e $\|r_c\|$, identificando a perda da relação do verdadeiro *gap* entre os valores da função objetivo primal e dual,

$$\left(\frac{\|(r_b,r_u)\|}{1+\|(b^T,u^T)\|} > 10^{-8} \;\text{ ou }\; \frac{\|r_c\|}{1+\|c\|} > 10^{-8}\right) \;\text{ e }\; \left(\frac{\max(\|(r_b^k, r_u^k)\|, \|r_c^k\|)/\mu_k}{\max(\|r_b^0, r_u^0)\|, \|r_c^0\|)/\mu_0} \geq 10^6\right).$$

- Estágio sub-ótimo: a solução é considerada ser sub-ótima quando nenhum dos itens anteriores são satisfeitos e o número de iterações atinge um limite pré-estabelecido.

6.1.1 Solução dos sistemas lineares: versão original

Na versão original do código PCx é implementada a abordagem direta para resolver o sistema de equações normais, que utiliza a fatoração esparsa de Cholesky concebida por Ng e Peyton. Um dos problemas encontrados ao utilizar este tipo de fatoração em métodos de pontos interiores é a ocorrência de pivôs muito próximos de zero, que geralmente aparecem durante as iterações. Para contornar essa situação, pivôs menores que 10^{-30} são substituídos por 10^{138}. As principais estapas da solução dos sistemas são:

- Reordenamento: utiliza o método de mínimo grau para determinar uma matriz de permutação P que reduza o preenchimento e o trabalho requerido durante a fatoração. Esta estratégia é idêntica a usada no SPARSPAK.

- Fatoração simbólica: gera uma estrutura de dados compacta na qual o fator de Cholesky será calculado.

- Fatoração numérica: calcula o fator de Cholesky na estrutura de dados criada na fatoração simbólica. Esta rotina explora a hierarquia de memória; trabalha com blocos de colunas que não ultrapassam a memória *cache* e utiliza técnicas de *loop unrolling* para diminuir o tempo das multiplicações matriz-vetor.

- Solução numérica: resolve os sistemas triangulares para determinar a solução.

- Tratamento de colunas densas: colunas densas são identificadas no início do processo e excluídas do cálculo de $A\Theta A^T$. A matriz A é dividida em uma parte esparsa A_s e outra densa A_d e a matriz Θ é particionada em Θ_s e Θ_d. Dessa forma,

  $$A\Theta A^T = A_s\Theta_s A_s^T + A_d\Theta_d A_d^T = M + A_d\Theta A_d^T.$$

  A matriz M é esparsa, e portanto pode ser fatorada em $PMP^T = LL^T$. Aplicando a fórmula de Sherman-Morrison-Woodbury tem-se,

  $$(M + A_d\Theta A_d^T)^{-1} = M^{-1} - (M^{-1}A_d)(\Theta_d + A_d^T M^{-1} A_d)^{-1} A_d^T M^{-1}.$$

- Refinamento da solução: quando o resíduo da solução do sistema for maior que um valor pré-estabelecido é aplicado um refinamento iterativo para melhorar a precisão. Esse processo é feito utilizando o método do gradiente conjugado precondicionado pelo fator de Cholesky.

Como a estrutura da matriz $A\Theta A^T$ não é alterada durante as iterações, a fatoração simbólica e o reordenamento são feitos uma única vez, no cálculo da solução inicial.

6.1.2 Solução dos sistemas lineares: versão modificada

O método do gradiente conjugado precondicionado pelo precondicionador separador foi agregado ao código PCx por Oliveira e Sorensen [74]. Nas iterações iniciais foi utilizado um precondicionador diagonal. Entretanto, em algumas classes de problemas o precondicionador

diagonal perde eficiência muito antes que o precondicionador separador esteja em condições de obter bom desempenho.

A fatoração controlada de Cholesky se adequa muito bem a esta situação, visto que é possível controlar o preenchimento e com isso a qualidade do precondicionador. Para adicionar a FCC ao PCx, foram implementadas as seguintes rotinas:

1. NulosAThetaAt : calcula o número de elementos não nulos em $A\Theta A^T$.
2. EstruDadosAThetaAt: calcula a estrutura de dados para armazenar $A\Theta A^T$.
3. PrecondAThetaAt: calcula o precondicionador LDL^T para a matriz $A\Theta A^T$. Esta rotina tem como subrotinas os precondicionadores escala diagonal e FCC.
4. SolveLDLt: resolve o sistema $LDL^T x = y$, para um dado vetor y.

Não utilizamos as rotinas disponíveis no código PCx para calcular $A\Theta A^T$, pois a estrutura de dados não era compatível com a da FCC. Todas as rotinas foram implementadas na linguagem C, exceto a FCC e o escala diagonal, que estão implementadas em FORTRAN77 e foram fornecidas pelo autor.

A matriz $A\Theta A^T$ é usada apenas para construir o precondicionador. Como a FCC é feita por colunas, não é necessário construir fisicamente $A\Theta A^T$. Esta é uma característica desejável, uma vez que $A\Theta A^T$ pode ser densa e, conseqüentemente, requerer grande quantidade de memória para armazenamento. Implementamos uma versão do código que trabalha neste cenário. A FCC é construída em m passos, sendo m a ordem da matriz $A\Theta A^T$. A cada passo é calculada uma coluna de $A\Theta A^T$ e a coluna referente do precondiconador. Dessa forma, é utilizado apenas um vetor de dimensão m para armazenar os valores numéricos das colunas de $A\Theta A^T$. Porém, quando ocorrem falhas na diagonal ao construir o precondicionador (ver Seção 6.3.3) é necessário recalcular a matriz $A\Theta A^T$ para que uma nova fatoração seja feita. Em nossos experimentos optamos por utilizar a versão em que a matriz $A\Theta A^T$ é construída fisicamente, pois na maioria dos casos ela apresenta melhor desempenho.

6.2 Problemas testes

Os problemas usados como testes são apresentados na Tabela 6.1. Esses problemas foram selecionados como uma forma de identificar classes de problemas onde a abordagem híbrida tem bom desempenho. Alguns desses testes são problemas de programação linear disponíveis em domínio público na NETLIB [3], STOCHLP [5], MISC [1, 2]. Os modelos QAP são da coleção QAPLIB [18] com a modificação descrita em [75]. O problema KBAPAH2 [12] pertence a uma coleção de problemas altamente malcondicionados nos quais implementações robustas tais como, PCx e HOPDM não atingem a solução ótima. A Tabela 6.1 descreve as características dos problemas testes. O número de linhas, colunas e elementos não nulos são referentes aos problemas após a fase de pré-processamento.

6.3 FCC em problemas de programação linear

Foram adotados os parâmetros originais do PCx, exceto as múltiplas correções de centralidade [46] que implicam na solução de vários sistemas lineares a cada iteração. Essa estratégia é mais adequada para tratar com abordagens diretas, já que a fatoração é feita uma única vez. Eliminando as múltiplas correções, tem-se o método primal-dual preditor-corretor e são resolvidos dois sistemas lineares a cada iteração. Em todos os experimentos foi utilizado como critério de parada para o gradiente conjugado a norma Euclidiana do resíduo $||r_k|| < 10^{-4}$ para resolver o primeiro sistema *(direção afim* $= \Delta_a$*)*, $||r_k|| < 10^{-8}$ para resolver o segundo *(direção final* $= \Delta_a + \Delta_c$*)*, e o número máximo de iterações é definido como m, sendo m a ordem do sistema.

Os experimentos numéricos foram realizados em três máquinas: tigre (processador Intel Xeon 2.8GHz com 1Gbyte de RAM), brontes (processador PENTIUM4 3.0GHz com 1Gybte de RAM) e anubis (processador PENTIUM4 2.8GHz com 1Gbyte de RAM). Em cada tabela de nossos experimentos é apresentada a máquina na qual os resultados foram obtidos. O tempo é medido em segundos, e o número de iterações é referente a solução do segundo sistema linear.

A fatoração controlada de Cholesky, apesar de não ter sido projetada especialmente para trabalhar com problemas de programação linear apresenta bom desempenho na solução de

Tabela 6.1: Problemas testes.

Problema	Linhas	Colunas	Nnulos	Coleção
KBAPAH2	15	28	299	KBAPAH
ELS-19	4350	13186	50882	QAP
CHR25A	8149	15325	53725	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
STE36C	27683	131076	512640	QAP
QAP12	2794	8856	33528	NETLIB
QAP15	5698	22275	85470	NETLIB
SCSD8-2B-64	5130	35910	112770	STOCHLP
SCSD8-2C-64	5130	35910	112770	STOCHLP
SCSD8-2R-432	8650	60550	190210	STOCHLP
GEN4	1537	4298	107103	MISC
NUG08	742	1632	5936	MISC
NUG12	2794	8856	33528	MISC
NUG15	5698	22275	85470	MISC
PDS-20	32276	106180	226494	MISC
PDS-40	64265	214385	457538	MISC
PDS-60	96503	332862	709178	MISC
PDS-80	126109	430800	916852	MISC
PDS-100	156243	514577	1096002	MISC

algumas classes de problemas ao longo de todas as iterações de pontos interiores. A seguir, são descritos os principais fatores que influenciam diretamente o desempenho desse precondicionador.

6.3.1 Escolha do η inicial

Um fator importante para trabalhar com a fatoração controlada é determinar um valor inicial para o parâmetro η. Para tal, procuramos uma estratégia baseada na densidade da matriz. Considere a média de elementos por linha (Mel) da matriz $A\Theta A^T$ calculada por,

$$Mel = \frac{|A\Theta A^T|}{m},$$

sendo m a dimensão de $A\Theta A^T$ e $|\ .\ |$ o número de elementos não nulos da matriz. A escolha de $\eta \leq -Mel$ produz o precondicionador escala diagonal. Foram realizados experimentos com diferentes valores de η para determinar o que melhor se enquadra na solução dos problemas. Na Tabela 6.2 é apresentado o desempenho do precondicionador na iteração inicial para três valores de η_0. Note que, na maioria dos problemas o número de iterações do gradiente conjugado diminui com o aumento do η, entretanto, o tempo de precondicionamento é maior. O aumento no número de iterações em alguns casos é devido ao incremento exponencial do precondicionador para evitar falhas na diagonal. O menor tempo de solução é obtido quando o precondicionador é construído com $\eta_0 = -Mel$. Esta parece ser uma escolha razoável, já que na primeira iteração a matriz é, em geral, bem condicionada. Dessa forma, o processo pode ser iniciado com o precondicionador escala diagonal implícito na FCC.

6.3.2 Influência do η

O valor do parâmetro η a ser usado durante as iterações é crucial para o bom desempenho da abordagem. Tipicamente, quando η aumenta, o número de iterações do gradiente conjugado é reduzido, uma vez que o precondicionamento é melhorado. Entretanto, o tempo necessário para construir o precondicionador é maior. Isso, sugere a existência de um η ótimo, no qual, o tempo total é mínimo. A Tabela 6.3 ilustra esse comportamento na solução do problema PDS40. Quando $\eta = -5$, é utilizado um precondicionador diagonal e o tempo para construí-lo é constante ao longo das iterações. Por outro lado, o tempo total para solução do problema é o maior dentre os três. As variações no tempo de precondicionamento para $\eta = 5$ e $\eta = 15$ ocorrem devido ao esquema utilizado para tratar com falhas na diagonal. Em alguns casos foi necessário recalcular o precondicionador oito vezes para evitar a falha. No entanto, ainda foi possível construir um bom precondicionador. Neste problema, foi observada uma pequena redução no número de iterações de pontos interiores com o aumento do η, 80 iterações para

Tabela 6.2: Escolha do η inicial.
(máquina tigre)
ITGC: iterações do gradiente conjugado; TT: tempo de solução do sistema (s).
* : falhas na diagonal.

Problema	$\eta = -Mel$		$\eta = -Mel + 20$		$\eta = -Mel + 40$	
	ITGC	TT	ITGC	TT	ITGC	TT
ELS-19	83	0.28	58	3.92	40	4.63
CHR25A*	50	0.40	113	4.21	53	15.58
SCR20	78	0.33	47	5.0	23	9.81
ROU20*	38	0.45	49	7.67	35	15.62
STE36C	93	3.15	75	61.78	46	515.16
QAP15	1217	5.41	630	7.37	451	28.13
SCSD8-2R-432*	32	0.98	92	15.88	97	19.39
NUG15	1507	10.28	873	15.04	685	27.71
PDS-40	577	14.7	293	35.91	152	29.48

$\eta = -5$, 79 para $\eta = 5$ e 78 para $\eta = 15$. Dentre os três valores de η testados, $\eta = 5$ apresentou o melhor resultado.

Os problemas da classe PDS apresentam comportamento semelhante ao longo das iterações, facilitando dessa forma o uso de uma estratégia para determinar um bom incremento para o parâmetro η. Na Tabela 6.4 é apresentada a influência do η na solução de alguns problemas desta classe. Note que, nos problemas PDS-20 e PDS-30 o menor tempo de solução é obtido para $\eta = 5$, já no PDS-60 o melhor parâmetro a ser usado é $\eta = 15$.

Tabela 6.3: Influência do η na solução do problema PDS-40.
(máquina brontes)
IT: iterações de pontos interiores; ITGC: iterações do gradiente conjugado;
TP: tempo de precondicionamento (s); TT: tempo total para resolver o problema (s).

	$\eta = -5$			$\eta = 5$			$\eta = 15$		
IT	ITGC	TP	TT	ITGC	TP	TT	ITGC	TP	TT
0	577	0.07	14.9	267	0.64	11.7	293	17.7	35.5
5	832	0.07	22.1	247	0.64	11.4	177	2.43	13.6
10	805	0.07	21.3	244	0.65	11.2	192	2.48	14.9
15	699	0.07	17.4	186	0.66	8.4	139	2.53	10.6
20	487	0.07	13.3	123	3.95	9.7	99	2.51	9.0
25	519	0.07	14.9	95	4.31	8.7	72	11.34	16.6
30	1228	0.07	14.7	184	3.83	12.2	116	10.6	18.4
35	1554	0.07	30.7	254	3.66	14.4	151	11.88	21.3
40	1456	0.07	39.6	246	3.59	14.8	150	17.78	21.9
45	1430	0.07	41.7	265	4.10	16.2	177	12.03	23.3
50	1698	0.07	47.3	305	4.15	17.8	208	14.10	27.9
55	2039	0.07	56.1	412	3.73	22.1	252	12.62	28.6
60	2852	0.07	75.7	510	4.19	26.0	295	13.06	31.1
65	4066	0.07	110.0	746	4.39	36.2	447	11.76	39.2
70	5638	0.07	151.1	1084	4.56	50.0	655	14.10	53.5
75	10517	0.07	251.4	1880	4.62	77.9	1071	13.72	70.9
78							3044	11.85	152.7
79				1226	4.45	59.5			
80	15563	0.07	346.4						
TT	5746.8			2114.6			2465.3		

Tabela 6.4: Influência do η na solução de problemas PDS.
(máquina brontes)
IT: iterações de pontos interiores; TT: tempo total de solução do problema (s).

	PDS-20		PDS-30		PDS-60	
η	IT	TT	IT	TT	IT	TT
-5	61	3945.3	73	2842.5	81	17481.5
5	61	1223.4	74	1152.3	81	6183.9
15	61	1284.7	74	1388.1	80	6141.6
25	61	1489.6	74	1744.7	80	6476.3

Foi desenvolvida uma estratégia para determinar o incremento do parâmetro η baseado no tempo necessário para convergência do gradiente conjugado. O processo é iniciado com o precondicionador escala diagonal, ou seja, $\eta_0 = -Mel$. A partir da segunda iteração η é incrementado por dez. Se o tempo de solução diminuir em relação à iteração anterior, é um bom indicativo que o precondicionamento foi melhorado e novamente este parâmetro é incrementado em dez. Por outro lado, se o tempo de solução aumentar, o precondicionador volta a ser construído com o η da iteração anterior e é mantido, até que o número de iterações do gradiente conjugado atinja um valor pré-estabelecido, o qual será discutido na seção 6.4.1. Após este ponto, o parâmetro η é incremenado em dez a cada iteração. Na Figura 6.1 é apresentado um algoritmo para o incremento do η. A Tabela 6.5 exibe alguns experimentos nos quais a solução ótima do problema foi obtida utilizando essa estratégia.

Em muitos experimentos foi observado a existência de um valor η para o qual a solução ótima é atingida. Todavia, esses valores variam muito com o problema, dificultando a escolha de um critério que seja ideal para todas as situações. Quando o parâmetro η atinge a dimensão do problema, é construído o fator completo de Cholesky e teoricamente, a convergência do gradiente conjugado ocorre em apenas uma iteração. Entretanto, a quantidade de tempo e memória requerida pode ser muito alta. Essa situação simula o comportamento do método direto, porém, é menos eficiente.

```
Algoritmo Incremento
{ Objetivo: Incrementar o parâmetro η. }
  η_0 = −Mel,  ITGC : iterações do GC,  m :  dimensão do sistema,
  i :  iteração de pontos interiores,
  t_i : tempo para resolver os sistemas da iteração i.
  se  i = 2  então
    η ← η + 10
    Flag ← 0
  fim se
  enquanto  ITGC ∗ 1.5 < m  e  Flag = 0  faça
    se i > 2  e  Flag = 0 então
      se  t_i ≤ t_{i−1}  então
        η ← η + 10
      senão
        η ← η − 10
        Flag ← 1
      fim se
    fim se
  fim enquanto
  se  ITGC ∗ 1.5 ≥ m  então
    η ← η + 10
  fim se
fim algoritmo
```

Figura 6.1: Algoritmo para incremento do parâmetro η.

6.3.3 Correção na diagonal

Um sério problema ao trabalhar com fatorações incompletas é a existência de pivôs não positivos. Muitas estratégias foram propostas para contornar esse problema, as quais podem ser classificadas em três categorias [14] :

- Aumentar a diagonal da matriz acrescentando um incremento diagonal. Isso pode ser feito localmente ou globalmente, ou seja, um valor positivo pode ser acrescentado

Tabela 6.5: Resultados do algoritmo para incremento de η.
(máquina tigre)
η_0: η na iteração inicial; η_F: η na iteração final;
IT: iterações de pontos interiores; TT: tempo total de solução do problema (s).

Problema	η_0	η_F	IT	TT
KBAPAH2	-8	15	9	0.01
CHR25A	-30	140	28	464.0
GEN4	-476	-406	18	107.4
SCSD8-2B-64	-138	-138	7	5.46
SCSD8-2B-64	-138	-138	7	5.36
SCSD8-2r-432	-226	-226	18	96.7
NUG08	-12	48	10	6.54
NUG15	-26	204	26	7026.3
PDS-20	-5	5	61	1470.6
PDS-40	-5	5	77	2778.6
PDS-60	-5	5	81	8871.0
PDS-80	-5	5	83	11203.1
PDS-100	-5	15	87	14927.4

apenas a posição da diagonal de $A\Theta A^T$ onde ocorreu a falha, ou pode ser acrescentado a todos os elementos da diagonal.

- Substituir $A\Theta A^T$ por uma matriz M na qual é garantida a existência da fatoração incompleta $M = \tilde{L}\tilde{L}^T$ e usar $\tilde{L}$ como um precondicionador para $A\Theta A^T$.

- Não modificar a matriz A, mas encontrar uma outra forma para fatoração que evite pivôs nulos.

Estamos utilizando um método pertencente a primeira categoria. Ao ocorrer a falha um incremento exponencial é aplicado globalmente a $A\Theta A^T$, e a fatoração incompleta de $A\Theta A^T + \alpha I$ é novamente calculada. Esse processo se repete até que a fatoração incompleta seja aplicada com sucesso. Em alguns problemas, o processo foi repetido muitas vezes até determinar um α adequado, aumentando o tempo de precondicionamento e prejudicando o desempenho do precondicionador.

Campos [19], inicialmente, utilizou na FCC um esquema proposto por Gill *et al.* [42], desenvolvido com o objetivo de encontrar uma fatoração definida positiva de uma matriz Hessiana originada em problemas de programação não linear. Contudo, na solução de problemas de cálculo estrutural, o incremento exponencial apresentou melhores resultados e por isso foi adotado na FCC.

6.3.4 Reordenamento

Para analisar a influência do reordenamento na solução dos sistemas lineares oriundos de programação linear, foram resolvidos oito problemas pelo programa PCx, sendo o sistema da segunda iteração gravado em arquivo. Os arquivos foram lidos por um programa descrito por Carmo [21] e os resultados estão apresentados nas Tabelas 6.6, 6.7 e 6.8.

A Tabela 6.6 exibe o tempo de reordenamento e precondicionamento usando quatro estratégias: sem ordenamento, mínimo grau aproximado, Cuthill-McKee reverso e contagem de colunas. Foi utilizado $\eta = 0$ para garantir que as quatro estratégias tenham o mesmo número de elementos não nulos. Os números pequenos à esquerda do tempo de precondicionamento indicam quantas vezes o precondicionador foi reconstruído de modo a evitar valor muito pequeno ou negativo na diagonal. Como era esperado, o menor tempo de reordenamento é obtido com a contagem de colunas, já que é a estratégia mais simples dentre as três. O método de mínimo grau aproximado, por ser mais elaborado, consome o maior tempo. Como o reordenamento é aplicado uma única vez, é preferível que seja utilizada a melhor estratégia, independente do tempo de executá-la. Não ficou evidente nesta classe de problemas a influência da técnica de reordenamento no tempo de precondicionamento. O problema PDS-100, apresenta o menor tempo de precondicionamento quando nenhuma das estratégias é aplicada. Para os problemas NUG15, QAP15 e ROU20, o menor tempo é obtido com a contagem de colunas. O método de mínimo grau apresenta melhores resultados nos problemas SCSD8-2C-64 e STE36C, os quais possuem matrizes mais densas. Outro fato observado foi que as estratégias de reordenamento não influenciam na estabilidade da diagonal. Nos problemas em que ocorrem falhas, o número de correções é aproximadamente o mesmo.

A Tabela 6.7 apresenta o número de elementos não nulos obtidos utilizando diferentes valores de η para as quatro estratégias. Para valores de η negativos não foi possível obter uma

Tabela 6.6: Tempos de reordenamento (ord) e precondicionamento (pre) para $\eta = 0$.
(máquina anubis)
SEM: sem reordenamento; AMD: mínimo grau aproximado;
RCM: Cuthill-McKee reverso; COL: contagem de colunas;
Mel: média de elementos por linha da matriz $A\Theta A^T$;
Os números pequenos à esquerda do tempo indicam o número de vezes que o precondicionador foi recalculado para evitar falhas na diagonal.

Problema	elementos não nulos	Mel	tempo(s)						
			SEM	AMD		RCM		COL	
			pre	ord	pre	ord	pre	ord	pre
ELS19	137825	31.7	2.68	1.01	3.48	0.07	1.07	0.05	1.15
SCR20	166709	32.8	9.16	1.06	4.47	0.22	3.14	0.17	3.83
ROU20	356689	48.5	25.49	1.00	19.43	0.47	16.53	0.37	15.87
STE36C	1564487	56.5	288.95	24.59	103.84	2.28	148.04	1.78	114.23
QAP15	155986	27.4	3.70	1.23	3.38	0.18	3.11	0.16	2.80
NUG15	150470	26.4	2.34	1.31	2.94	0.14	2.28	0.06	1.03
SCSD8-2C64	709145	138.2	$_{17}$78.72	2.71	$_{17}$59.40	0.90	$_{17}$77.55	0.78	$_{17}$324.84
PDS100	787355	5.2	$_{7}$2.11	24.76	$_{8}$19.85	1.55	$_{9}$3.24	0.59	3.22

evidência de qual método produz o menor número de elementos no precondicionador. Nos problemas PDS-100, ROU20 e SCR20 os melhores resultados são obtidos sem o uso de reordenamento. A heurística de mínimo grau é melhor nos problemas ELS-19, QAP15, SCSD8-2C-64 e STE36C. Cuthill-McKee reverso não foi melhor em nenhuma das situações e o método de contagem de colunas apresentou melhores resultados para o problema NUG15. Quando $\eta = 0$ o número de elementos não nulos é sempre o mesmo. Para valores positivos de η a melhor estratégia, em termos do número de elementos não nulos do precondicionador, em todos os problemas é o método do mínimo grau aproximado.

A Tabela 6.8 mostra o tempo total de precondicionamento e das iterações, em segundos, usando as quatro estratégias. Pelos experimentos não está clara qual estratégia de reordenamento produz o menor tempo de solução. Em alguns casos, foi observado que a melhor estratégia se mantém para diferentes valores de η. Os problemas NUG15, PDS-100, ROU20 e SCR20 ilustram esse comportamento. No problema ELS-19 o método de contagem de colunas torna-se o mais eficiente com o aumento do η.

Tabela 6.7: Número de elementos não nulos para diferentes valores de η.
(máquina anubis)
SEM: sem reordenamento; AMD: mínimo grau aproximado;
RCM: Cuthill-McKee reverso; COL: contagem de colunas;
Mel: média de elementos por linha da matriz $A\Theta A^T$.

	$\eta = -\text{Mel}/2$				$\eta = 0$			
Problema	SEM	AMD	RCM	COL	SEM	AMD	RCM	COL
ELS19	66117	66057	66101	66103	137825	137825	137825	137825
SCR20	77770	77818	77777	77795	166709	166709	166709	166709
ROU20	168952	169062	168984	168985	356689	356689	356689	356689
STE36C	747997	747942	747966	747975	1564487	1564487	1564487	1564487
QAP15	73378	73243	73320	73292	155986	155986	155986	155986
SCSD8-2C64	347903	346709	347803	347933	709145	709145	709145	709145
NUG15	67901	68013	67949	67881	150470	150470	150470	150470
PDS100	251603	262577	261955	282212	787355	787355	787355	787355
	$\eta = \text{Mel}/2$				$\eta = \text{Mel}$			
Problema	SEM	AMD	RCM	COL	SEM	AMD	RCM	COL
ELS19	206883	204477	206772	207070	275590	270729	275375	275926
SCR20	252446	249495	252025	252813	332785	327082	331969	333522
ROU20	539634	535301	539308	540211	714503	706054	714025	715770
STE36C	2365604	2349140	2365723	2366600	3138122	3105828	3138470	3140099
QAP15	235101	232143	234894	233554	314373	305875	313518	310561
SCSD8-2C64	1062200	884543	1063100	1060367	1399412	1041242	1405571	1401779
NUG15	229846	226808	229338	227873	303421	295650	302374	299345
PDS100	1210869	1060457	1178611	1206556	1661648	1300063	1572474	1627295

Tabela 6.8: Tempo de precondicionamento + iterações (s).
(máquina anubis)
SEM: sem reordenamento; AMD: mínimo grau aproximado;
RCM: Cuthill-McKee reverso; COL: contagem de colunas;
Mel: média de elementos por linha da matriz $A\Theta A^T$;
Os números pequenos à esquerda do tempo indicam o número de vezes que o precondicionador foi recalculado para evitar falhas na diagonal.

	$\eta = -\text{Mel}/2$				$\eta = 0$			
Problema	SEM	AMD	RCM	COL	SEM	AMD	RCM	COL
ELS19	0.35	0.74	0.19	0.22	2.70	3.58	1.11	1.17
SCR20	1.17	0.90	0.61	0.73	9.25	4.61	3.23	3.93
ROU20	3.83	3.37	2.57	2.19	25.69	19.75	16.73	16.08
STE36C	30.25	17.31	22.79	17.14	289.65	105.02	148.83	114.95
QAP15	${}_{9}4.83$	1.44	${}_{9}4.11$	${}_{7}3.51$	3.97	3.67	3.37	3.05
SCSD8-2C64	${}_{17}102.64$	${}_{17}127.13$	${}_{17}105.08$	${}_{17}169.44$	${}_{17}145.83$	${}_{17}170.37$	${}_{17}163.90$	${}_{17}412.92$
NUG15	${}_{8}2.15$	${}_{8}3.18$	${}_{1}1.01$	0.26	2.43	3.14	2.38	1.07
PDS100	624.13	1489.58	937.48	1231.14	${}_{7}887.31$	${}_{8}2228.99$	${}_{9}1478.79$	1559.74

	$\eta = \text{Mel}/2$				$\eta = \text{Mel}$			
Problema	SEM	AMD	RCM	COL	SEM	AMD	RCM	COL
ELS19	4.70	6.10	2.21	2.06	9.58	9.38	5.55	3.15
SCR20	13.96	8.20	6.18	7.09	19.27	12.61	10.01	10.75
ROU20	42.44	38.62	33.55	31.11	62.79	63.86	56.17	50.00
STE36C	424.94	199.29	242.16	204.66	580.30	322.91	358.06	309.28
QAP15	6.54	6.83	6.24	5.82	9.58	11.08	9.99	9.73
SCSD8-2C-64	${}_{17}201.89$	${}_{17}191.83$	${}_{17}204.15$	${}_{17}476.93$	${}_{17}248.17$	${}_{17}229.41$	${}_{17}250.23$	${}_{17}568.57$
NUG15	5.24	6.52	5.97	2.37	9.12	10.93	11.79	5.04
PDS100	${}_{10}1110.51$	${}_{10}2131.89$	${}_{10}1213.78$	${}_{7}1900.81$	${}_{10}1548.55$	${}_{10}2688.29$	${}_{10}1753.76$	${}_{7}5122.08$

6.4 Precondicionador híbrido

Antes de apresentar os resultados numéricos com o precondicionador híbrido para os problemas testes, serão mostrados alguns experimentos preliminares para motivar o uso dessa abordagem na solução de problemas de programação linear. Foram escolhidos dois problemas testes para ilustrar o comportamento dos precondicionadores. Na versão do código que implementa o precondicionador separador é utilizado o precondicionador escala diagonal nas primeiras iterações. A troca entre eles ocorre quando o número de iterações do gradiente conjugado (ITGC) atinge $m/4$ ou o *gap* da solução inicial for reduzido em 10^6. A FCC é iniciada com um precondicionador escala diagonal e o incremento no parâmetro η ocorre de acordo com o Algoritmo 6.1. Para comparar as duas abordagens nas iterações que o gradiente conjugado necessita muitas iterações para convergir alteramos o critério de troca entre os precondicionadores escala diagonal e separador. No momento em que ITGC$>(2/3)m$ o precondicionador separador é ativado e na FCC o parâmetro η é incrementado por 10. O número de iterações do gradiente conjugado apresentado nas tabelas a seguir é referente a solução do segundo sistema linear *(direção final* $= \Delta_a + \Delta_c$*)*.

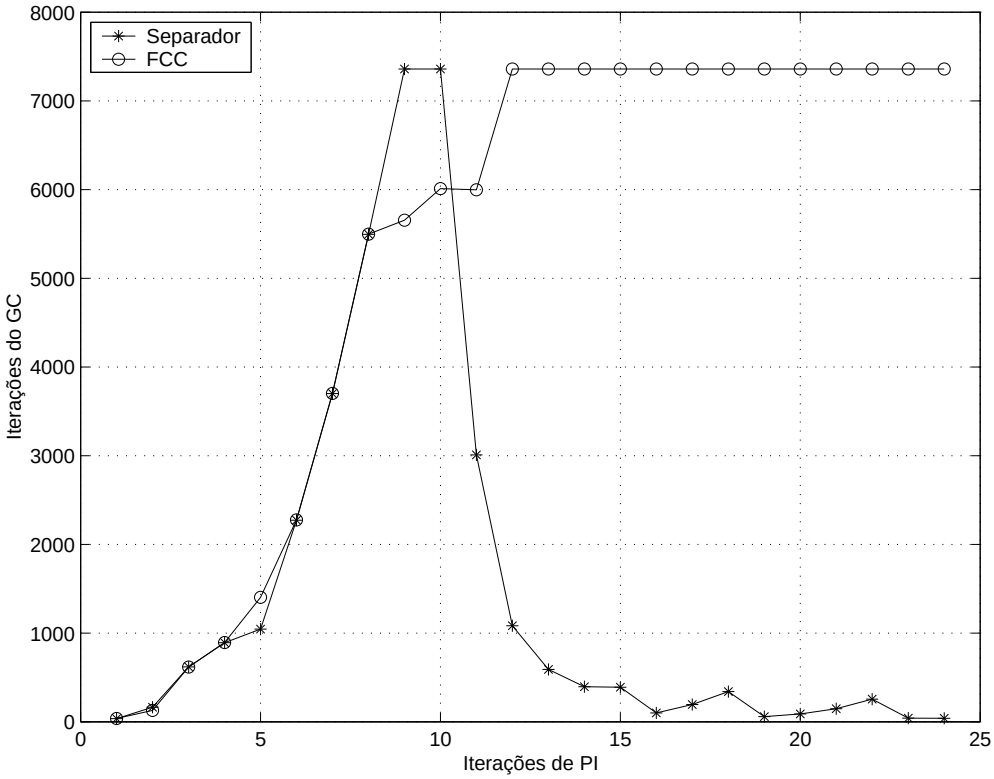
A Tabela 6.9 ilustra o desempenho do gradiente conjugado ao longo das iterações de pontos interiores na solução do problema ROU20 utilizando as duas abordagens separadamente. São apresentados o número de iterações do gradiente conjugado, o tempo necessário para a solução do sistema e o número de elementos não nulos no precondicionador. Note que, na iteração 9, acontece a troca do precondicionador escala diagonal para o separador e na FCC o parâmetro η é incrementado. Nas primeiras iterações após a troca, o tempo da solução do sistema pela FCC é muito menor. Este fato, induz que seria possível melhorar o desempenho do precondicionador separador usando a FCC neste estágio. Após a iteração 16, ocorre exatamente o oposto, o menor tempo de solução é obtido com o precondicionador separador. Outro fato a ser observado é que o número de elementos não nulos na FCC é menor quando comparada ao separador. A Figura 6.2 ilustra o comportamento das duas abordagens, são comparados o número de iterações do gradiente conjugado e o tempo necessário para solução dos sistemas ao longo das iterações de pontos interiores. Note que, após a iteração 16 ocorrem oscilações no tempo de solução dos sistemas pelo precondicionador separador. Este fato se deve ao uso da mesma matriz B durante algumas iterações. Na versão original do precondicionador separador, a solução ótima é obtida com o tempo 5671.15s. Com o uso da FCC em todas as iterações não foi possível alcançar a solução ótima, pois o número de

iterações do gradiente conjugado atinge a dimensão da matriz e as direções podem não ser precisas. Todavia, um aumento mais significativo no parâmetro η poderia apresentar melhor desempenho.

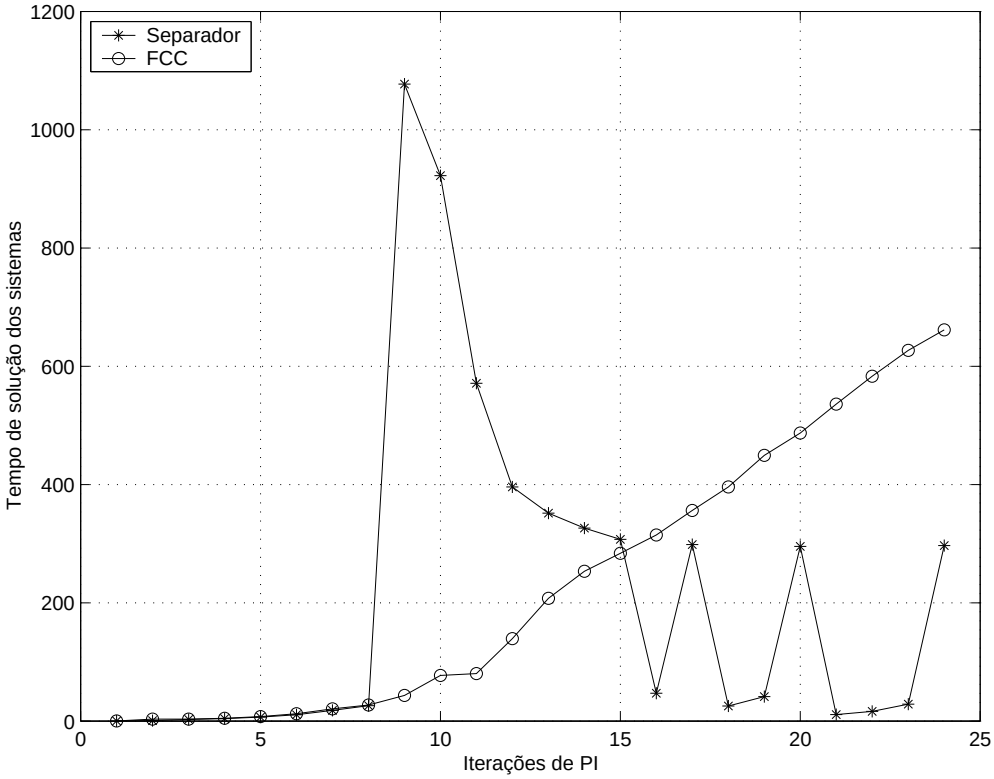
A Tabela 6.10 mostra o desempenho do precondicionador separador e da fatoração controlada de Cholesky ao longo das iterações de pontos interiores na solução do problema QAP15. Para as duas abordagens a solução ótima não é alcançada. Na iteração 4 ocorre a troca do precondicionador escala diagonal para o separador e na FCC o parâmetro η é incrementado. Durante as iterações 4 a 17 o tempo de solução dos sistemas é menor utilizando a FCC, porém a partir da iteração 9 as soluções podem não ser precisas, já que o número de iterações do gradiente conjugado atinge o limite. No separador a fase crítica para convergência do gradiente conjugado está entre as iterações 3 e 9. Este problema ilustra a situação em que o precondiciondor escala diagonal não consegue chegar ao ponto onde o precondicionador separador esteja em condições de obter bom desempenho. A Figura 6.3 ilustra o comportamento das duas abordagens; são comparados o número de iterações do gradiente conjugado e o tempo necessário para solução dos sistemas ao longo das iterações de pontos interiores.

Tabela 6.9: Desempenho dos precondicionadores no problema ROU20.
máquina brontes
PS: precondicionador separador; FCC: fatoração controlada de Cholesky;
ITGC: iterações do gradiente conjugado ; IT: iterações de pontos interiores;
TEMPO: tempo de solução dos sistemas (s);
NNULOS: elementos não nulos no precondicionador.

	ITGC		TEMPO		NNULOS	
IT	PS	FCC	PS	FCC	PS	FCC
1	36	38	0.3	0.3	7359	7359
2	164	128	0.8	3.4	7359	146940
3	619	617	2.9	3.3	7359	7359
4	892	893	4.3	4.9	7359	7359
5	1405	1403	6.8	7.8	7359	7359
6	2273	2276	10.9	12.5	7359	7359
7	3700	3702	18.1	20.7	7359	7359
8	5496	5498	26.2	27.2	7359	7359
9	7359	5655	1077.2	43.5	3313524	146940
10	7359	6011	922.4	77.4	3709036	220648
11	3010	5999	571.3	80.5	4217393	294140
12	1084	7359	396.1	139.4	4406922	371360
13	590	7359	351.6	207.7	4496392	444622
14	395	7359	326.5	253.5	4683950	517745
15	391	7359	307.5	283.8	4682429	590749
16	100	7359	47.1	314.9	4682429	663640
17	196	7359	298.6	356.2	4778235	736404
18	341	7359	25.6	396.1	4778235	809053
19	59	7359	41.5	449.4	4778235	881589
20	89	7359	295.4	487.2	4955125	954101
21	147	7359	11.1	536.1	4955125	1026335
22	255	7359	16.6	583.2	4955125	1098519
23	41	7359	28.7	627.1	4955125	1177059
24	40	7359	297.0	661.7	4931119	1242586
Tempo total:			5161.8	6291.9		



(a)



(b)

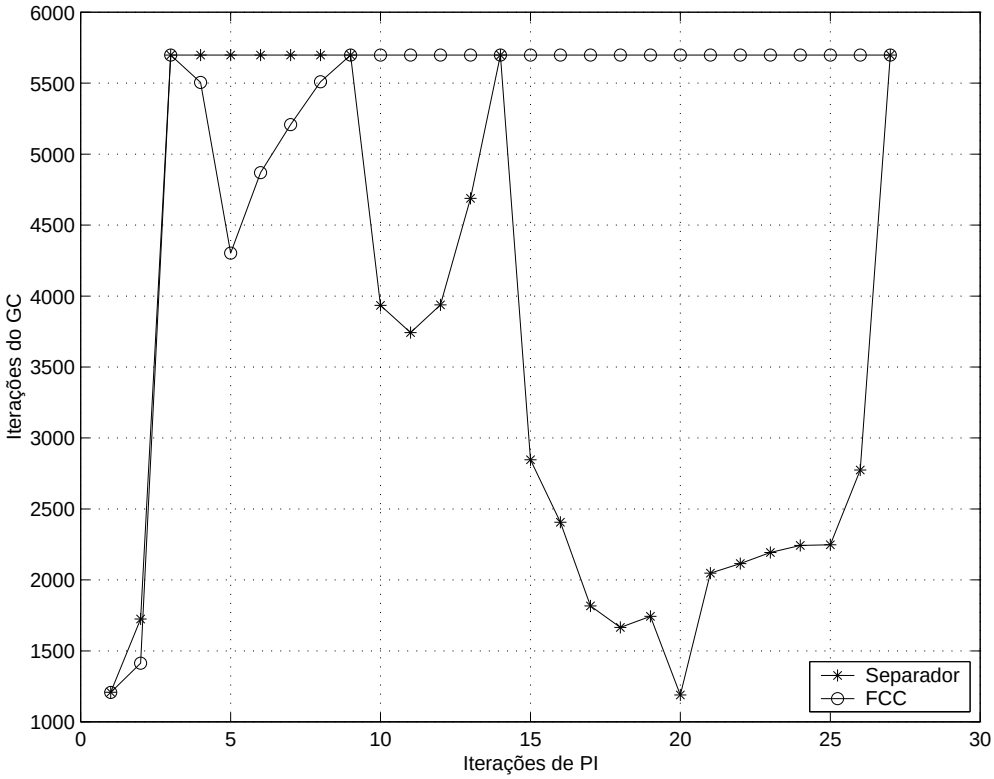
Figura 6.2: Problema ROU20.

(a) Número de iterações do GC ao longo das iterações de pontos interiores.

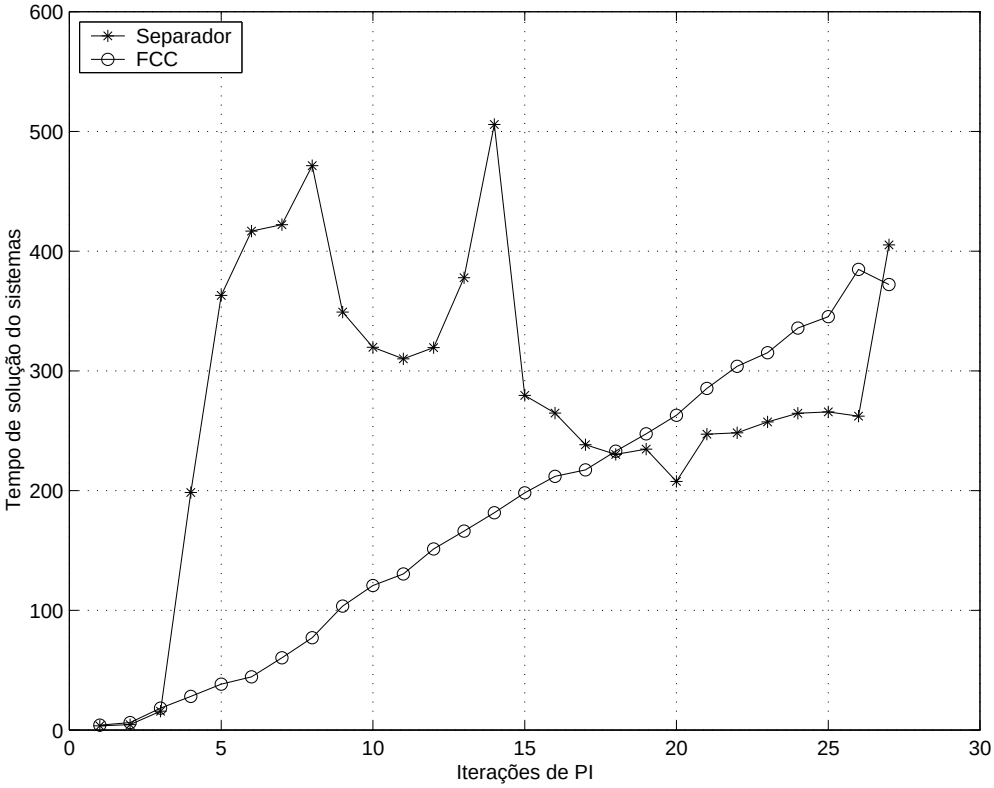
(b) Tempo de solução dos sistemas ao longo das iterações de pontos interiores.

Tabela 6.10: Desempenho dos precondicionadores no problema QAP15.
máquina brontes
PS: precondicionador separador; FCC: fatoração controlada de Cholesky;
ITGC: iterações do gradiente conjugado; IT: iterações de pontos interiores;
TEMPO: tempo de solução dos sistemas (s);
NNULOS: elementos não nulos no precondicionador.

	ITGC		TEMPO		NNULOS	
IT	PS	FCC	PS	FCC	PS	FCC
1	1208	1207	3.6	4.1	5698	5698
2	1725	1413	4.5	6.2	5698	113347
3	5698	5698	15.8	18.3	5698	5698
4	5698	5505	198.4	28.1	582189	113347
5	5698	4301	363.1	38.4	1143419	172987
6	5698	4869	416.7	44.5	1347783	229698
7	5698	5208	422.2	60.4	1434829	286112
8	5698	5510	471.5	77.2	1384443	399189
9	5698	5698	349.1	103.5	1435643	455669
10	3935	5698	319.7	120.7	1535275	511990
11	3743	5698	310.1	130.4	1523134	579775
12	3938	5698	319.5	151.3	1499981	623864
13	4688	5698	377.9	166.2	1653396	679705
14	5698	5698	505.8	181.6	1597839	735822
15	2847	5698	279.6	198.1	1548111	791681
16	2406	5698	264.8	212.0	1550296	847335
17	1817	5698	238.4	217.3	1726610	902004
18	1665	5698	230.3	232.9	1507559	958430
19	1743	5698	234.6	247.5	1627545	1013787
20	1189	5698	207.6	262.9	1486592	1068938
21	2047	5698	247.1	285.3	1543396	1124150
22	2115	5698	248.3	303.8	1480600	1179152
23	2193	5698	257.4	315.2	1534454	1234061
24	2243	5698	264.7	335.8	1507710	1288908
25	2248	5698	265.8	345.3	1492130	1343609
26	2774	5698	262.2	384.8	1499149	1398234
27	5698	5698	405.3	372.2	1621529	1452728
30		5698		441.2		1615380
Tempo total:			7953.3	7954.1		



(a)



(b)

Figura 6.3: Problema QAP15.

(a) Número de iterações do GC ao longo das iterações de pontos interiores.

(b) Tempo de solução dos sistemas ao longo das iterações de pontos interiores.

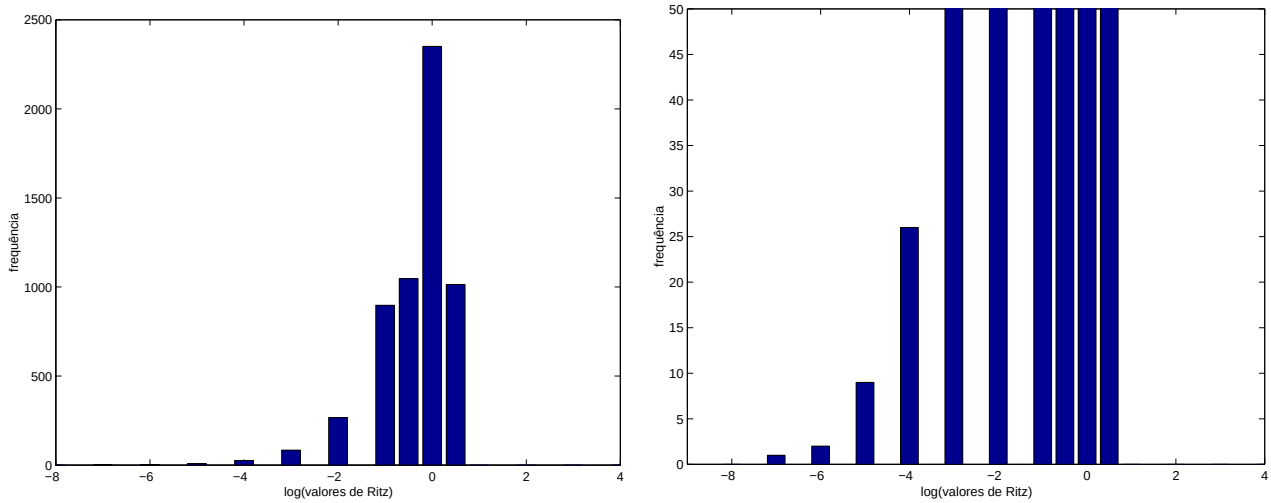
6.4.1 Mudança de fases

Como foi mostrado por Resende e Veiga [80] e é evidente pelas Figuras 6.2 e 6.3, durante as iterações de pontos interiores pode ser identificada a existência de duas fases. Na primeira delas, métodos baseados em fatorações incompletas da matriz apresentam melhores resultados, por outro lado, o precondicionador separador é mais eficiente na segunda fase. As Tabelas 6.9 e 6.10 ilustram esse comportamento. Estamos propondo o uso de uma abordagem híbrida na solução de problemas de programação linear. O processo é iniciado com a fatoração controlada de Cholesky e após a mudança de fases ser identificada o precondicionador separador continua o percurso.

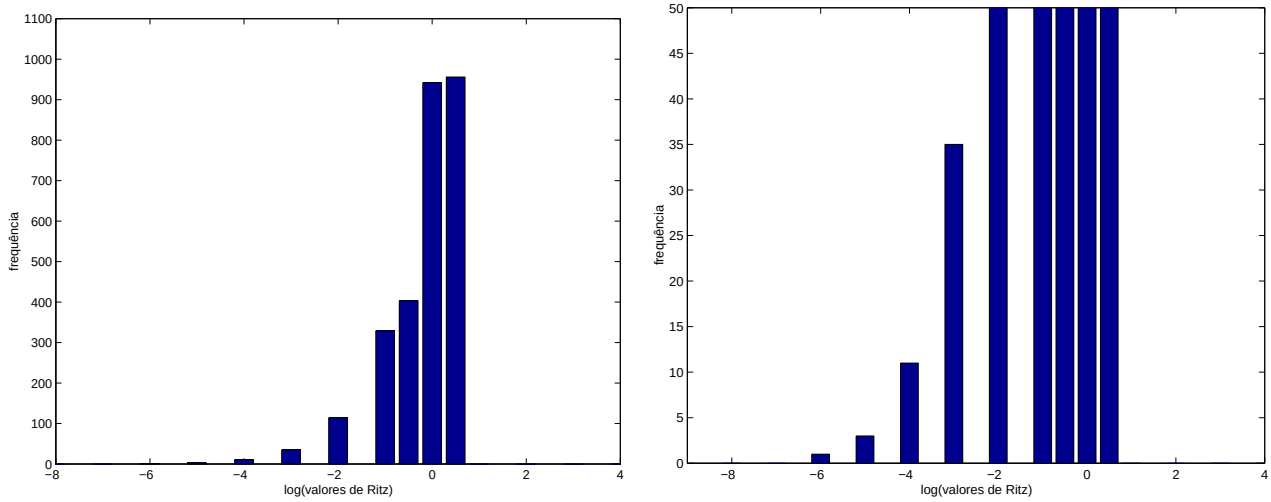
A escolha de um critério para determinar a mudança de fases é crucial para o bom desempenho da abordagem híbrida. Inicialmente, foi analisada uma estratégia baseada no número de condição da matriz, que pode ser estimado pelos valores de Ritz,

$$\kappa_2(M^{-1}A\Theta A^T M^{-T}) \approx \frac{\tau_{\max}}{\tau_{\min}},$$

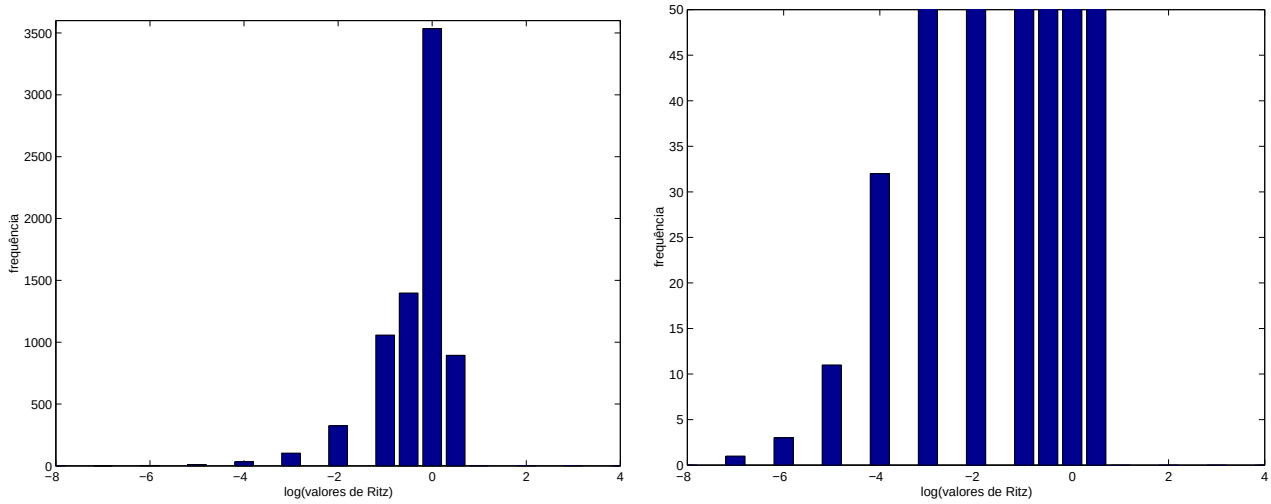
sendo, $\tau_{\min}$ e $\tau_{\max}$ estimativas para o menor e maior autovalores da matriz $A\Theta A^T$ precondicionada, respectivamente. Os valores de Ritz fornecem uma boa aproximação para o maior autovalor, contudo, a estimativa do menor autovalor não é precisa, fazendo com que o número de condição κ_2 seja mal estimado. Infelizmente, os valores de κ_2 variam muito com o problema que está sendo resolvido, dificultando a escolha de um critério que apresenta bom desempenho para todos os problemas. O uso de um método eficiente para calcular κ_2 pode apresentar custo computacional elevado. A Figura 6.4 apresenta a distribuição dos valores de Ritz da matriz precondicionada dos problemas QAP15, NUG12 e ROU20. Para cada problema foi escolhida a matriz de uma iteração considerada crítica para a convergência do GC. O espectro de autovalores é apresentado em duas escalas distintas de freqüência.



(a) Problema QAP15



(b) Problema NUG12



(c) Problema ROU20

Figura 6.4: Distribuição dos valores de Ritz.

Neste trabalho é utilizada uma heurística que funciona bem para a maioria dos problemas testes . A troca dos precondicionadores é identificada se uma das seguintes condições ocorrer:

1. o *gap* inicial é reduzido por 10^6 e o número de iterações do gradiente conjugado é maior que $m/4$, sendo m a ordem do sistema;
2. já foram feitas mais de 10 correções na diagonal para evitar falhas na FCC e o número de colunas do problema é inferior a 16000.
3. o número de iterações do gradiente conjugado é igual a m e o parâmetro η já foi incrementado três ou mais vezes.

Geralmente, quando a primeira condição é atendida, é um indicativo que a FCC já não está apresentando bom desempenho e o precondicionador separador poderá obter melhores resultados, já que uma solução ótima pode estar próxima. A segunda condição prevê situações em que a FCC apresenta dificuldades para construir o precondicionador. Como, neste caso o número de colunas do problema pode ser considerado pequeno (< 16000) o precondicionador separador poderá apresentar melhor desempenho, ainda que seja nas iterações iniciais. A última condição, sugere que a troca dos precondicionadores deve ocorrer, pois não foi possível determinar um bom parâmetro η, e a convergência do gradiente conjugado não está sendo atingida. O uso de direções imprecisas pode comprometer a solução do problema. Por outro lado, o fato de nenhuma das condições ser atendida é um indicativo que a fatoração controlada está apresentando bom desempenho ao longo de todas as iterações.

Baseado em experimentos foi observado que na maioria dos casos o menor tempo de solução é obtido, incrementando η em dez, a partir do momento em que o número de iterações do gradiente conjugado atingir $\frac{2}{3}m$. As Tabelas 6.11, 6.12 e 6.13 são exemplos dessa situação.

O precondicionador híbrido engloba dois precondicionadores. A fatoração controlada e o precondicionador separador. A mudança de fases entre eles é o principal fator para o bom desempenho da abordagem. No entanto, está é uma tarefa longe de ser trivial, e o critério proposto ainda pode ser melhorado.

Tabela 6.11: Variação do tempo com incremento do η no problema ROU20.
(máquina tigre)
im: indica que o incremento é iniciado quando o número de iterações do GC atinge im
Tempo: tempo total de solução do problema (s).

	Tempo		
	$\Delta\eta = 5$	$\Delta\eta = 10$	$\Delta\eta = 15$
m	2351.4	2361.5	2290.1
$\frac{2}{3}m$	2671.6	2281.4	2431.3
$\frac{1}{2}m$	2678.1	2465.8	2523.2

Tabela 6.12: Variação do tempo com incremento do η no problema QAP15.
(máquina tigre)
im: indica que o incremento é iniciado quando o número de iterações do GC atinge im.
Tempo: tempo total de solução do problema (s).

	Tempo		
	$\Delta\eta = 5$	$\Delta\eta = 10$	$\Delta\eta = 15$
m	4758.1	4688.8	4446.1
$\frac{2}{3}m$	4364.6	4012.2	4273.0
$\frac{1}{2}m$	4375.1	4905.4	4354.2

Tabela 6.13: Incremento do η nos problemas NUG12.
(máquina tigre)
im: indica que o incremento é iniciado quando o número de iterações do GC atinge im.
Tempo: tempo total de solução do problema (s).

	Tempo		
	$\Delta\eta = 5$	$\Delta\eta = 10$	$\Delta\eta = 15$
m	447.3	312.5	326.5
$\frac{2}{3}m$	439.1	308.7	325.2
$\frac{1}{2}m$	490.1	338.8	350.2

6.4.2 Precondicionador híbrido versus abordagem direta

Nesta seção, o comportamento da abordagem híbrida é comparado com o método direto e com a fatoração incompleta de Cholesky (FIC) [4]. Os resultados são apresentados na Tabela 6.14. Para cada problema, é indicado o número de iterações de pontos interiores e o tempo total de CPU. O número que aparece entre parênteses nas iterações da abordagem híbrida representa a iteração em que houve a troca entre os precondicionadores. A ausência desse número indica que a FCC foi utilizada em todas iterações, pois o critério para troca não foi atingido.

O problema KBAPAH2 pertence a uma coleção de problemas degenerados que foram construídos para implementações robustas falharem para atingir a otimalidade. A abordagem direta termina em estágio desconhecido e a abordagem híbrida encontra a solução ótima após nove iterações.

Para os problemas em que a solução ótima é alcançada, o número de iterações de pontos interiores é aproximadamente o mesmo para as duas abordagens. Nos problemas NUG12, NUG15, QAP12 e QAP15 é interessante notar que a solução ótima não é obtida usando a abordagem direta. Esse fato ocorre devido à estratégia utilizada para tratar com elementos próximos de zero que aparecem durante a fatoração; eles são substituídos por valores muito grandes (10^{138}) e nesses problemas ocasionam grandes erros residuais. A fatoração controlada de Cholesky utiliza uma pequena perturbação na diagonal para evitar este tipo de falha, dessa forma é possível obter uma solução aproximada mais precisa na fase I; o precondicionador separador não é afetado por este problema.

A FIC falha para atingir a convergência em quase todos os problemas testes devido ao critério utilizado para tratar com falhas na diagonal. Nesta implementação da FIC, quando surgem elementos muito pequenos ou negativos na diagonal eles são substituídos por 1.

Os problemas SCSD8-2B-64, SCSD8-2C-64 e SCSD8-2R-432 têm muitas colunas densas na matriz A. Para evitar a densidade na matriz $A\Theta A^T$, a abordagem direta trata essas colunas separadamente. Nos dois primeiros problemas, a abordagem direta não atinge a solução ótima devido à grandes erros residuais e no último ocorre uma falha na execução. A abordagem híbrida obtém a solução ótima para os dois primeiros problemas usando apenas um precondicionador diagonal, já que ele apresenta bom desempenho ao longo de todas as

iterações. No problema SCSD8-2R-432 ocorre a troca para o precondicionador separador e o tempo de solução diminui (de 96.7 para 53.5), quando comparado ao tempo obtido usando a FCC em todas as iterações (ver Tabela 6.5).

O problema GEN4 representa o caso onde a troca dos precondicionadores não ocorreu no momento adequado, e por isso a abordagem híbrida não atingiu a solução ótima. A troca ocorreu muito cedo; se o parâmetro η tivesse sido incrementado por mais duas iterações a solução ótima seria obtida após dezoito iterações. A abordagem direta termina em estágio desconhecido.

Os problemas STE36 tem centenas de milhares de elementos não nulos no fator completo de Cholesky. Após dois dias de processamento não havia sido calculado nem mesmo o fator de Cholesky. A abordagem híbrida não atinge a solução ótima utilizando os parâmetros originais do PCx. Contudo, é possível obter uma solução aproximada fazendo prifeastol=1.0^{-7} e substituindo (6.5) por (6.6).

Com a abordagem direta não foi possível obter nenhum resultado para os problemas PDS-100, STE36A, STE36B e STE36C devido à grande quantidade de tempo e memória requerida.

A Tabela 6.15 mostra a influência do preenchimento ocorrido na fatoração no desempenho da abordagem híbrida. A coluna M indica o número de elementos não nulos da matriz precondicionadora na última iteração. A coluna $A\Theta A^T$ mostra o número de elementos não nulos da parte triangular inferior da matriz. O número de elementos não nulos do fator completo de Cholesky é apresentado na coluna Matriz L.

Os problemas QAP também apresentam a matriz $A\Theta A^T$ não muito esparsa. A fatoração de Cholesky dessas matrizes tem um número muito elevado de elementos não nulos, tornando a abordagem direta menos efetiva. Com o uso do parâmetro η negativo torna possível construir um precondicionador eficiente na fase I com baixo custo. Na fase II é possível trabalhar com a mesma matriz B, durante algumas iterações, reduzindo o número de fatorações LU e conseqüentemente, o tempo de solução.

Tabela 6.14: Comparação entre as abordagens híbrida, FIC e direta (Cholesky).
(máquina tigre)
*: termina em estágio desconhecido.
F: representa falhas durante a execução; T: tempo excedido ($> 48h$).
$^{+}$: critério alterado.

	Iterações			Tempo		
Problema	Híbrida	Cholesky	FIC	Híbrida	Cholesky	FIC
KBAPAH2	9	13	F	0.0	*0.0	F
ELS-19	32 (15)	30	F	252.5	497.1	F
CHR25A	32 (9)	31	F	46.8	91.9	F
SCR15	24 (12)	22	F	51.4	78.6	F
SCR20	21 (13)	21	F	260.1	910.6	F
ROU20	24 (13)	21	F	2285.2	5807.8	F
STE36A	24	T	F	$^{+}$7834.6	T	F
STE36B	22	T	F	$^{+}$6537.2	T	F
STE36C	22	T	F	$^{+}$8362.7	T	F
QAP12	22 (8)	46	F	379.9	*340.6	F
QAP15	23 (10)	54	F	4015.7	*2977.4	F
SCSD8-2B-64	8	8	F	6.7	*1.7	F
SCSD8-2C-64	8	5	F	6.7	*1.6	F
SCSD8-2R-432	18 (11)	F	F	53.5	F	F
GEN4	34 (7)	36	F	*438.5	*136.7	F
NUG08	11 (7)	12	F	5.1	3.15	F
NUG12	22 (11)	51	F	309.2	*609.1	F
NUG15	24 (15)	44	F	3291.5	*3885.7	F
PDS-20	61	61	61	1470.5	1197.8	2579.8
PDS-40	77	74	77	2778.2	13246.1	6642.1
PDS-60	81	75	83	8871.5	41460.3	19335.8
PDS-80	83	81	83	11203.2	94756.1	31397.9
PDS-100	87	T	87	14927.9	T	48736.3

Tabela 6.15: A influência do preenchimento no desempenho da abordagem híbrida.

η_o: valor de η na iteração inicial;
$|M|$: elementos não nulos no precondicionador da iteração final.
$A\Theta A^T$: elementos não nulos em $A\Theta A^T$;
Matriz L: elementos não nulos no fator completo de Cholesky.

Problema	η_0	$\|M\|$	$A\Theta A^T$	Matriz L
KBAPAH2	-8	120	120	120
ELS-19	-31	63705	137825	3849458
CHR25A	-30	28821	249324	2653126
SCR15	-26	27060	59009	1330759
SCR20	-32	1010537	166709	6672137
ROU20	-48	4995689	356689	20818131
STE36A	-56	1246527	1564487	176625274
STE36B	-56	1107933	1564487	176625274
STE36C	-56	1284858	1564487	176625274
QAP12	-31	468892	60174	2138580
QAP15	-27	1493922	155986	8197968
SCSD8-2B-64	-138	5130	709145	27026045
SCSD8-2C-64	-138	5130	709145	27026045
SCSD8-2R-432	-226	8650	1956985	76779485
GEN4	-476	186415	731971	1033917
NUG08	-12	33849	9274	233032
NUG12	-20	658470	56405	2793152
NUG15	-26	1762211	150470	11053969
PDS-20	-5	307339	169915	7089645
PDS-40	-5	631026	341419	28195225
PDS-60	-5	981789	523496	58118583
PDS-80	-5	1271825	676093	94270275
PDS-100	-5	1512937	2061607	120240013

Os problemas PDS não geram matrizes $A\Theta A^T$ densas. Por outro lado, a fatoração de Cholesky produz um grande número de elementos não nulos. Conforme a dimensão do problema aumenta, a abordagem híbrida apresenta melhor desempenho quando comparada a abordagem direta. A Figura 6.5 ilustra este fato. Para estes problemas o critério de troca de fases não é atingido, uma vez que a FCC apresenta bom desempenho durante todas as iterações de pontos interiores. Além disso, após algumas iterações o número de elementos não nulos no precondicionador é mantido constante.

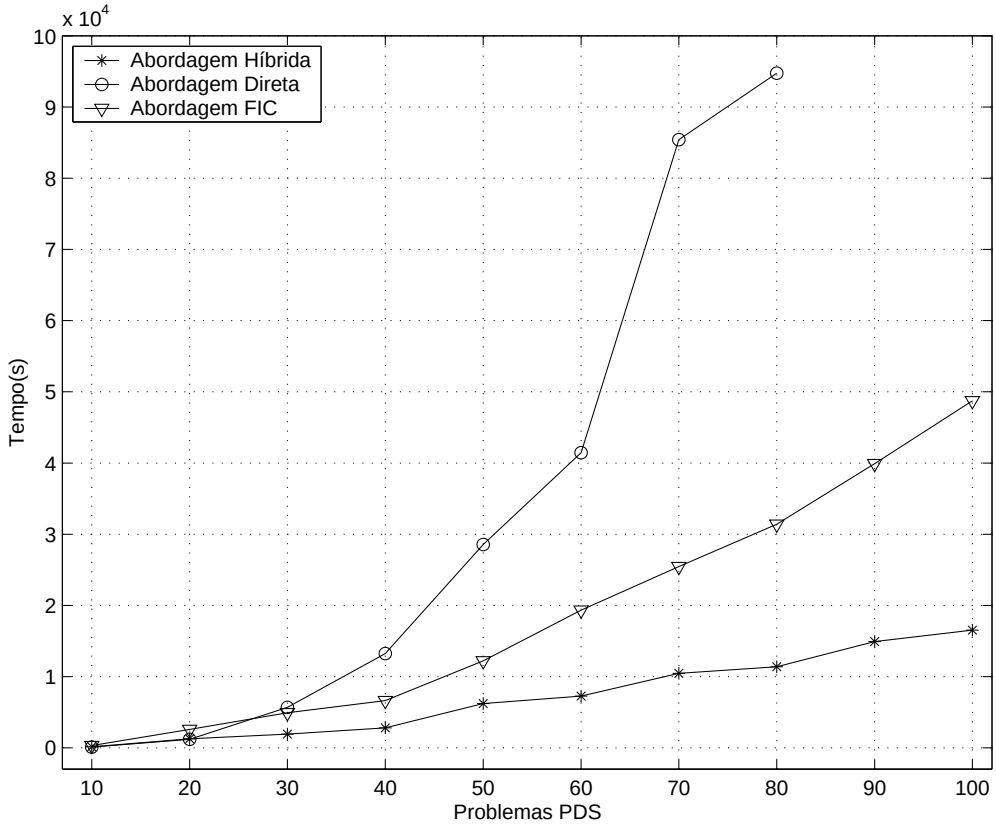


Figura 6.5: Problemas PDS - abordagem híbrida versus abordagem direta e FIC.

6.4.3 Abordagem híbrida - versão modificada

Com o objetivo de avaliar a influência do critério de parada do GC no desempenho da abordagem híbrida foram realizados alguns experimentos adicionais. Como a solução obtida nas iterações iniciais, tipicamente, ainda está longe da otimalidade, o critério de parada do GC pode ser relaxado . Dessa forma, são requeridas menos iterações para convergência.

Foi desenvolvida uma versão do código utilizando como critério de parada para o GC a norma Euclidiana do resíduo $\|r_k\| < 10^{-4}$ para resolver os dois sistemas (direção afim e direcão final). Quando a viabilidade primal (6.3) for menor que 10^{-5} ou a mudança de fases identificada é atribuido $\|r_k\| < 10^{-8}$. A partir desse momento, se o resíduo verdadeiro for superior a 10^{-7} aplica-se um refinamento sucessivo na solução. O número máximo de iterações é definido como m, sendo m a ordem do sistema.

Os critérios utilizados para a troca de fases e incremento do parâmetro η também foram alterados, já que estes dependem do número de iterações do GC. Na versão modificada, o parâmetro η é incrementado por 10 quando o número de iterações do GC for superior a $m/4$. A troca entre os precondicionadores ocorre se alguma das condições descritas a seguir acontecer:

1. o gap inicial é reduzido por 10^6 e o número de iterações do GC é superior a $m/4$.
2. já foram feitas mais de 10 correções na diagonal para evitar falhas na FCC e o número de colunas do problema é inferior a 16000.
3. o número de iterações do GC é superior a $m/2$ e o parâmetro η já foi incrementado três ou mais vezes.

Na Tabela 6.16 é apresentada a influência desses parâmetros no desempenho da abordagem híbrida. A coluna Híbrida-O mostra os resultados originais (Tabela 6.14) e a coluna Hibrida-M apresenta o os resultados obtidos com os parâmetros modificados. Como pode ser observado não houve alterações significativas nos resultados. A pequena redução no número de iterações (IT) ocorre nos problemas em que é aplicado o refinamento sucessivo. Com a versão modificada foi possível resolver até a otimalidade os problemas STE36A e STE36C.

Tabela 6.16: Influência dos parâmetros usados na abordagem híbrida (máquina tigre)

*: termina em estágio desconhecido, $^{+}$: critério alterado.

Problema	Iterações		Tempo	
	Híbrida-O	Híbrida-M	Híbrida-O	Híbrida-M
KBAPAH2	9	8	0.0	0.0
ELS-19	32	31	252.5	290.1
CHR25A	32	32	46.8	87.4
SCR15	24	24	51.4	44.1
SCR20	21	21	260.1	234.0
ROU20	24	24	2285.2	2288.4
STE36A	24	37	$^{+}$7834.6	36545.7
STE36B	22	27	$^{+}$6537.2	$^{+}$24327.1
STE36C	22	41	$^{+}$8362.7	80142.3
QAP12	22	21	379.9	449.1
QAP15	23	23	4015.7	3247.1
SCSD8-2B-64	8	7	6.7	6.6
SCSD8-2C-64	8	7	6.7	6.4
SCSD8-2R-432	18	18	53.5	65.0
GEN4	34	12	*438.5	*315.8
NUG08	11	10	5.1	3.8
NUG12	22	21	309.2	349.1
NUG15	24	24	3291.5	2971.1
PDS-20	61	61	1470.5	1258.3
PDS-40	77	78	2778.2	2811.8
PDS-60	81	80	8871.5	7254.2
PDS-80	83	83	11203.2	11337.1
PDS-100	87	87	14927.9	16540.4

Capítulo 7

Conclusões e trabalhos futuros

Uma das principais etapas em implementações de pontos interiores é a solução dos sistemas de equações de Newton, presentes a cada iteração desses métodos. Visto que esta é a tarefa que consome a maior parte do tempo de processamento, muitas estratégias de solução foram propostas. Em geral, são utilizadas duas formulações: os sistemas aumentados, que são simétricos e indefinidos, e as equações normais, que envolvem matrizes simétricas e definidas positivas. A abordagem mais comumente usada em códigos de pontos interiores, implementa a fatoração de Cholesky das matrizes definidas positivas. Esta fatoração pode ser densa para algumas classes de problemas, tornando o uso de métodos iterativos mais adequado, desde que sejam convenientemente precondicionados. Como as matrizes variam muito durante as iterações de pontos interiores é difícil encontrar uma estratégia de precondicionamento que apresenta bom desempenho ao longo de todas iterações. Baseado neste fato, propomos um precondicionador híbrido para resolver os sistemas de equações normais pelo método do gradiente conjugado. Assumimos a existência de duas fases durante as iterações. Na fase I, o precondicionador é construído pela fatoração controlada de Cholesky, e na fase II pelo precondicionador separador.

A fatoração controlada de Cholesky ainda não tinha sido usada em problemas de otimização e foi mostrado que esta estratégia apresenta bom desempenho na classe de problemas analisados. A facilidade de controlar o preenchimento no precondicionador pelo ajuste do parâmetro η é uma característica desejável neste contexto. À medida que os sistemas tornam-se mal-condicionados um maior preenchimento é permitido, melhorando a qualidade do precondicionamento. Esta técnica possibilita simular até mesmo o comportamento do método direto,

escolhendo η que produza o fator completo como precondicionador. A forma para o incremento de η influencia diretamente o desempenho do precondicionador. Desenvolvemos uma estratégia na qual o processo é iniciado com o precondicionador escala diagonal, presente na FCC, e o parâmetro η é incrementado à medida que a FCC perde eficiência. Nossa estratégia apresentou bons resultados.

O desempenho do precondicionador usado na fase I, pode ser melhorado determinando o ajuste ótimo para o parâmetro η. Campos [20] sugeriu determinar o parâmetro ótimo η_{opt} quando uma série de sistemas lineares com a mesma matriz de coeficientes necessita ser resolvida. Problemas desse tipo estão em inúmeras aplicações, como cálculo de inversas, soluções de sistemas de equações diferenciais com diferentes termos independentes, entre outras. Como nestes problemas a matriz dos coeficientes permanece inalterada durante as iterações, foi determinada uma estratégia baseada no tempo para solução do sistema. Em métodos de pontos interiores a matriz dos coeficientes é alterada durante as iterações, dificultando o desenvolvimento de um critério baseado apenas no tempo de solução que seja eficiente para todos os problemas. Estimativas do número de condição podem auxiliar no ajuste do parâmetro η.

O tempo de solução na fase I também pode ser reduzido com o uso de estratégias de reordenamento para minimizar o preenchimento no fator. Foram realizados experimentos com o método de mínimo grau aproximado, Cuthill-Mckee reverso e contagem de colunas. Como foi observado, não ficou claro para qual heurística de reordenamento a FCC apresenta o melhor desempenho. Na maioria das implementações de pontos interiores é adotada a heurística de mínimo grau, porém quando fatorações incompletas estão sendo usadas, essa pode não ser a melhor alternativa. Em alguns de nossos experimentos o menor tempo de precondicionamento foi obtido aplicando a heurística de contagem de colunas. Como trabalho futuro, poderiam ser propostas estratégias de reordenamento adicionais e desenvolvido um estudo para determinar a melhor estratégia a ser utilizada.

Outro fator para melhorar o desempenho da FCC é utilizar novas estratégias para evitar falhas que ocorrem pela presença de elementos muito próximos de zero na diagonal. Em alguns experimentos o incremento exponencial foi repetido muitas vezes até que o problema fosse evitado, prejudicando o desempenho do precondicionador. Uma estratégia a ser analisada é o esquema proposto por Gill *et al* [42] para encontrar uma fatoração definida positiva de uma matriz Hessiana originada em otimização não linear. Além disto, para melhorar a

estabilidade numérica na fatoração, poderiam ser aplicados termos de regularização primal e dual, similares aos utilizados no código HOPDM [45].

O precondicionador separador, utilizado na fase II, foi projetado para trabalhar nas proximidades da solução ótima e por isso apresenta melhor desempenho nas iterações finais, quando os sistemas já estão malcondicionados. Uma característica interessante desse precondicionador é a possibilidade de trabalhar com a mesma matriz B durante algumas iterações. Dessa forma, o tempo de solução é reduzido, já que o maior trabalho está em escolher as colunas para formar B e calcular a fatoração LU. Como o separador foi desenvolvido para as iterações finais, Oliveira e Sorensen [74] utilizaram o precondicionador escala diagonal nas iterações iniciais, porém em algumas classes de problemas este precondicionador não atinge o ponto onde o separador estivesse em condições de obter bom desempenho. Com o uso da fatoração controlada de Cholesky na fase I, foi possível melhorar o desempenho e a robustez do precondicionador separador.

Determinar o momento ideal para a troca das fases, é crucial para o bom desempenho da abordagem híbrida. A princípio, analisamos uma estratégia baseada no número de condição da matriz, estimado pelos valores de Ritz. A estimativa para o maior autovalor é precisa, contudo, a presença de valores de Ritz muito próximos de zero, não fornecem uma boa aproximação para o menor autovalor, fazendo com que o número de condição seja mal estimado. Infelizmente, estes valores variam muito com o problema, e o processo não foi adotado. Desenvolvemos uma heurística que apresentou bons resultados para maioria dos problemas testes. Entretanto, devido à grande variedade entre as classes de problemas, a estratégia proposta pode ser melhorada. Resultados mais promissores podem ser obtidos com o desenvolvimento de um critério específico para cada classe de problemas. Como pesquisa futura, podem ser aplicados outros critérios para estimar o número de condição da matriz [30, 49]. Ou ainda, pode ser elaborada uma estratégia intercalando os dois precondicionadores durante a fase I.

A abordagem híbrida, FCC na fase I e precondicionador separador na fase II, apresentou desempenho superior na solução de algumas classes de problemas quando comparada a abordagem direta. Tipicamente, esse fato ocorre quando o fator de Cholesky tem um grande número de elementos não nulos. Foram apresentados alguns casos onde foi possível determinar a solução ótima de problemas nos quais a abordagem direta não atinge esse estágio. Alguns problemas por consumo excessivo de tempo e memória e outros por grandes erros

residuais acumulados durante as iterações devido ao mecanismo utilizado para tratar com pivôs muito próximos de zero na fatoração.

Finalizando, outro campo para pesquisa futura é utilizar o precondicionador híbrido na solução de problemas de programação quadrática. Neste caso, a solução poderia ser obtida via sistemas aumentados. Estes sistemas possuem maior estabilidade numérica e o precondicionador híbrido pode ser uma importante ferramenta nesta classe de problemas.

Referências Bibliográficas

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

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

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

[4] SLATEC collection. The dsics.f subroutine in NETLIB repository. Online at http://www.netlib.org/slatec/lin/dsics.f.

[5] Stochastic LP test sets. Hungarian Academy of Sciences OR Lab. Online at http://www.sztaki.hu/~meszaros/public_ftp/lptestset/stochlp.

[6] I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA Journal on Computing*, 1(2):84–106, 1989.

[7] I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga. An implementation of Karmarkar's algorithms for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[8] A. Altman and J. Gondzio. Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. *Optimization Methods and Software*, 11:275–302, 1999.

[9] E. Andersen, J.Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for linear programming. In T. Terlaky, editor, *Interior Point Methods of Mathematical Programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[10] K. M. Anstreicher. Potential reduction methods. In T. Terlaky, editor, *Interior point methods in mathematical programming*. Kluwer Academic Publishers, 1996.

[11] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[12] 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.

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

[14] M. Benzi, G. H. Golub, and M. Tuma. A robust incomplete factorization preconditioner for positive definite matrices. *Numerical Linear Algebra with Applications*, 10:385–400, 2003.

[15] L. Bergamaschi, J. Gondzio, and G. Zilli. Preconditioning indefinite systems in interior point methods for optimization. *Computational Optimization and Applications*, 28(2):149–171, 2004.

[16] 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. Indicado por J. Gondzio para ser publicado em uma edição especial de *Computational Optimization and Applications on LINEAR ALGEBRA ISSUES ARISING IN INTERIOR POINT METHODS*, 2006.

[17] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal Numerical Analysis*, 8:639–655, 1971.

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

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

[20] 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.

[21] F. C. Carmo. *Análise da influência de algoritmos de reordenação de matrizes esparsas no desempenho do método $CCCG(\eta)$*. Dissertação de mestrado, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, 2005.

[22] T. J. Carpenter and D. F. Shano. An interior point method for quadratic programs based on conjugate projected gradients. *Computational Optimization and Applications*, 2:5–28, 1993.

[23] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. H. Tong. A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems. *SIAM Journal on Scientific Computing*, 15(2):338–347, 1994.

[24] V. Chvátal. *Linear Programming*. W. H. Freeman, New York, 1983.

[25] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright. PCx an interior point code for linear programming. *Optimization Methods and Software*, 11(2):397–430, 1999.

[26] I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Soviet Mathematics Doklady*, 29(8):674–675, 1967.

[27] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. Oxford University Press, New York, 1989.

[28] I. S. Duff, R. G. Grimes, and J. G. Lewis. User's guide for the Harwell-Boeing sparse matrix collection (release I). Technical Report PA-92-86, CERFACS, Toulouse, France, 1992.

[29] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29(4):635–657, 1989.

[30] I. S. Duff and C. Vomel. Incremental norm estimation for dense and sparse matrices. *BIT*, 42(2):300–322, 2002.

[31] C. Durazzi and V. Ruggiero. Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems. *Numerical Linear Algebra with Applications*, 10:673–688, 2003.

[32] R. Fletcher. Conjugate gradient methods for indefinite systems. In *Lecture Notes in Mathematics*, volume 506, pages 73–89. Proceedings of Dundee Conference on Numerical Analysis - 1975, Springer-Verlag, 1976.

[33] G. E. Forsythe and E. G. Straus. On best conditioned matrices. In *Proceedings of the Amer. Math. Soc.*, volume 6, pages 340–345, 1955.

[34] R. Fourer and S. Mehrotra. Solving symmetric indefinite systems in an interior point method for linear programming. *Mathematical Programming*, 62:15–40, 1993.

[35] A. Frangioni and G. Gentile. New preconditioners for KKT systems of network flow problems. *SIAM Journal on Optimization*, 14(3):894–913, 2004.

[36] R. W. Freund. A transpose-free quasi-minimum residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.*, 14:470–482, 1993.

[37] R. W. Freund and N. M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[38] R. W. Freund and N. M. Nachtigal. Software for simplified Lanczos and QMR algorithms. *Applied Numerical Mathematics*, 19(3):319–341, 1995.

[39] A. George. Computer implementation of the finite-element method. Technical Report CS-208, Department of Computer Science, Stanford University, 1971.

[40] A. George and J. W. H. Liu. *Computer solution of large sparse definite systems*. Computational Mathematics, 1981.

[41] A. George and J. W. H. Liu. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19, 1989.

[42] P. E. Gill, W. Murray, D. B. Ponceléon, and M. A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix Anal. and Applications*, 13:292–311, 1992.

[43] G. H. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3^a edition, 1996.

[44] J. Gondzio. Implementing Cholesky factorization for interior point methods of linear programming. *Optimization*, 27:121–140, 1993.

[45] 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.

[46] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[47] J. Gondzio and T. Terlaky. A computational view of interior point methods for linear programming. In J. Beasly, editor, *Advances in Linear and Integer Programming*, pages 393–481. Oxford University Press, 1995.

[48] C. C. Gonzaga. Path following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992.

[49] R. G. Grimes and J. G. Lewis. Condition number estimation for sparse matrices. *SIAM J. Sci. Statist. Comput.*, 2(4):384–388, 1981.

[50] J. C. Haws and C. D. Meyer. Preconditioning KKT systems. Online at http://meyer.math.ncsu.edu/Meyer/PS_Files/KKT.pdf.

[51] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[52] P. Huard. Resolution of mathematical programming with nonlinear constraints by the method of centres. In J. Abadie, editor, *Nonlinear Programming*, pages 209–219. John Wiley & Sons, Inc., 1967.

[53] M. T. Jones and P. E. Plassmann. An improved incomplete Cholesky factorization. *ACM Trans. Math. Software*, 21:5–17, 1995.

[54] J. J. Júdice, J. Patricio, L. F. Portugal, M. G. C. Resende, and G. Veiga. A study of preconditioners for network interior point methods. *Comp. Optim. Appl.*, 24(1):5–35, 2003.

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

[56] N. Karmarkar and K. G. Ramakrishnan. Computational results of an interior point algorithm for large scale linear programming. *Mathematical Programming*, 52:555–586, 1991.

[57] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[58] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point method for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 29–47. Springer Verlag, 1989.

[59] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bureau Standards*, 45(4):255–282, 1950.

[60] J. W-H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Trans. Math. Software*, 11:141–153, 1985.

[61] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications*, 152:191–222, 1991.

[62] 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.

[63] 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):1–15, 1994.

[64] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 34(150):473–497, 1980.

[65] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 131–158. Springer Verlag, 1989.

[66] S. Mehrotra. Implementation of affine scaling methods: Aproximate solutions of systems of linear equations using preconditioned conjugate gradient methods. *ORSA journal on Computing*, 4:103–118, 1992.

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

[68] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear system of which the coefficient matrix is a symetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977.

[69] R. D. C. Monteiro and I. Adler. Interior path-following primal-dual algorithms, Part1: Linear Programming. *Mathematical Programming*, 44:27–41, 1989.

[70] N. Munksgaard. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients. *ACM Trans. Math. Software*, 6(2):206–219, 1980.

[71] C. Mészáros. The inexact minimum local fill-in ordering algorithm. Technical Report WP-95-7, Computer and Automation Research Institute, Hungarian, 1995.

[72] E. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Computing*, 14:1034–1056, 1993.

[73] A. R. L. Oliveira. *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1997.

[74] A. 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.

[75] M. Padberg and M. P. Rijal. *Location, Scheduling, Design and Integer Programming*. Kluwer Academic, Boston, 1996.

[76] L. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[77] L. Paige and M. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.

[78] B. N. Parlett and D. S. Scott. The Lanczos algorithm with selective orthogonalization. *Math. Comp.*, 33(145):217–238, 1979.

[79] 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.

[80] M.G. C. Resende and G. Veiga. An implementation of the dual affine scaling algorithm for minimum cost flow on bipartite uncapaciated networks. *SIAM Journal on Optimization*, 3:516–537, 1993.

[81] C. Ross, T. Terlaky, and J.-Ph. Vial. *Theory and algorithms for linear optimization: An interior point approach*. John Wiley, Chichester, 1997.

[82] Y. Saad. The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. *SIAM J. Numer. Anal.*, 19:470–484, 1982.

[83] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing Company, 1996.

[84] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[85] H. D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra Appl.*, 61:101–131, 1984.

[86] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10:36–52, 1989.

[87] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. on Sci. and Stat. Comput.*, 13(2):631–644, 1992.

[88] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1996.

[89] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior point methods. *Mathematical Programming*, 58:1–32, 1993.

[90] 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.

[91] S. Wright. *Primal-dual interior point methods*. SIAM, Philadelphia, 1996.

[92] M. Yannakakis. Computing the minimal fill-in is NP-complete. *SIAM Journal Algebraic Discrete Methods*, 2:77–79, 1981.

[93] Y. Zhang and R. A. Tapia. Superlinear and quadratic convergence of primal-dual interior point methods for linear programming revisited. *Journal of Optimization Theory and Applications*, 73:229–242, 1992.