UMA ABORDAGEM DE PRÉ-CONDICIONAMENTO HÍBRIDA PARA RESOLVER OS SISTEMAS LINEARES DE MÉTODOS DE PONTOS INTERIORES ITERATIVAMENTE

Carla T. L. S. Ghidini
Depto de Matemática Aplicada, IMECC, UNICAMP
13083-859, Campinas, SP
E-mail: carla@ime.unicamp.br

Aurelio Ribeiro L. Oliveira
Depto de Matemática Aplicada, IMECC, UNICAMP
13083-859, Campinas, SP
E-mail: aurelio@ime.unicamp.br

RESUMO

Neste trabalho, métodos iterativos são usados para resolver os sistemas lineares dos métodos de pontos interiores. Tais sistemas são mal-condicionados próximos de uma solução. Criar pré-condicionadores especialmente adaptados é fundamental. Por outro lado, os sistemas lineares iniciais não apresentam essa característica. Assim, torna-se conveniente utilizar pré-condicionadores híbridos. Nas primeiras iterações, a fatoração controlada de Cholesky é utilizada e nas iterações finais, o pré-condicionador separador, o qual se comporta muito bem próximo da solução. Devido a sua boa característica, este pré-condicionador pode ser caro de calcular. Assim, uma implementação cuidadosa deve ser realizada para obter resultados competitivos em relação a velocidade e robustez. O principal desafio ao implementar esse pré-condicionador é selecionar um conjunto adequado de colunas linearmente independentes para criar uma matriz não singular. Várias estratégias para encontrar tal conjunto são apresentadas. Experimentos numéricos são realizados para mostrar o desempenho das estratégias.

PALAVRAS-CHAVE: programação linear, métodos de pontos interiores, pré-condicionamento.

Programação Matemática

ABSTRACT

In this work iterative methods are used to solve the linear systems arising from interior point methods. These systems are ill-conditioned near a solution. Create preconditioners specially adapted is fundamental. The early linear systems do not present the same features. Then, it is advisable to adopt hybrid preconditioners. Concerning the first iterations, the controlled Cholesky factorization is used. After the splitting preconditioner is adopted. Its major advantage is the good behavior near a solution. Due to its good feature, this preconditioner can be expensive to compute. A careful implementation must be performed to achieve competitive results regarding both: speed and robustness. The main challenge to implement this preconditioner relies upon finding a suitable set of linearly independent columns to form a nonsingular matrix. Several strategies to help finding such set of columns are presented. Numerical experiments are performed in order to illustrate the performance of the given stategies.

KEYWORDS: linear programming, interior point methods, preconditioning.

Mathematical Programming

1 Introdução

O desenvolvimento de códigos sofisticados para resolver problemas de programação linear usando métodos de pontos interiores começaram desde o início dos trabalhos sobre este assunto. Existem três principais linhas de pesquisa voltadas para melhorar a eficiência de tais métodos ao resolver problemas de grande porte: redução do número total de iterações; técnicas para realizar rapidamente uma iteração e métodos específicos para determinadas classes de problemas.

Este trabalho lida com a segunda linha. Métodos iterativos são usados para resolver os sistemas lineares, que é o passo mais caro a cada iteração do método de pontos interiores. Uma vez que tais sistemas são muito mal-condicionados próximo da solução, desenvolver pré-condicionadores especialmente adaptados é muito importante. Por outro lado, como os sistemas lineares iniciais não apresentam a mesma característica, é aconselhável adotar pré-condicionadores híbridos, começando com um genérico (Bocanegra *et al.*, 2007).

Nas primeiras iterações, a fatoração controlada de Cholesky é adotada (Campos e Birkett, 1998). A sua principal vantagem é o parâmetro de controle que permite variar o pré-condicionador de todas as formas, desde um pré-condicionador diagonal até a fatoração de Cholesky completa. Depois de um certo número de iterações, o pré-condicionador separador é usado (Oliveira e Sorensen, 2005). Sua principal vantagem é um comportamento muito bom próximo da solução do problema de programação linear. No entanto, essa característica tem um preço. O pré-condicionador pode ser muito caro de calcular. Assim, uma implementação cuidadosa deve ser feita a fim de obter resultados numéricos competitivos tanto em relação a velocidade quanto a robustez. O principal desafio para implementar o pré-condicionador separador consiste em encontrar um conjunto adequado de colunas linearmente independentes para criar uma matriz não singular a ser fatorada, a partir da matriz de restrições.

O pré-condicionador separador evita calcular o sistema de equações normais. Existem diversas técnicas para ajudar a encontrar tal conjunto de colunas. Algumas são bem conhecidas, já aplicadas em outros contextos, enquanto outras foram desenvolvidas para calcular o pré-condicionador separador. Entre as técnicas utilizadas está o estudo da estrutura dos elementos não nulos da matriz de restrições a fim de acelerar a fatoração numérica. Outras questões de implementação, incluindo formas de fazer a troca de pré-condicionadores são discutidas em Ghidini *et al.* (2012) e Velazco *et al.* (2010).

A escolha da fatoração controlada de Cholesky se justifica devido à possibilidade de calcular um pré-condicionador de baixo custo nas primeiras iterações do método de pontos interiores e quando os sistemas lineares começam a se tornar mal condicionados, o pré-condicionador pode ser melhorado apenas alterando o valor de um parâmetro. Experimentos numéricos ilustram a necessidade de tais estratégias, a fim de resolver problemas de programação linear de grande porte.

O texto está organizado da seguinte forma: Na Seção 2, falamos sobre o método de pontos interiores Preditor-Corretor, como calcular as direções de busca e formas de resolver os sistemas lineares. Ainda nessa seção, descrevemos sobre os pré-condicionadores fatoração controlada de Cholesky e separador, que são usados numa abordagem híbrida de pré-condiconamento para resolução dos sistemas. Na Seção 3, apresentamos diversas técnicas para se obter uma implementação eficiente do pré-condicionador separador. A Seção 4 traz uma descrição dos experimentos computacionais realizados e os resultados obtidos e na Seção 5, estão as conclusões.

2 Método de Pontos Interiores

Entre os vários tipos de métodos de pontos interiores, um dos mais bem sucedidos é o Preditor-Corretor (Mehrotra, 1992). Neste método, as direções de busca são obtidas resolvendo dois sistemas lineares

resultantes da aplicação do método de Newton às condições KKT do problema de programação linear na forma padrão. Primeiro a direção afim é calculada resolvendo o sistema:

$$\begin{bmatrix} 0 & I & A^t \\ Z & X & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta\tilde{z} \\ \Delta\tilde{y} \end{bmatrix} = \begin{bmatrix} r_d \\ r_a \\ r_p \end{bmatrix}, \quad (1)$$

em que A é uma matriz de posto completo e dimensão $m \times n$, x é o vetor das variáveis primais de dimensão n, y é o vetor das variáveis livres de dimensão m e z é o vetor de variáveis de folga duais de dimensão n. Além disso, $X = diag(x)$, $Z = diag(z)$ e os resíduos são dados por: $r_p = b - Ax$, $r_d = c - A^t y - z$ e $r_a = -XZe$ e e é um vetor unitário. Finalmente as direções de busca $(\Delta x, \Delta y, \Delta z)$ são obtidas resolvendo (1) com r_a substituído por: $r_c = \mu e - XZe - \Delta\tilde{X}\Delta\tilde{Z}e$, em que μ é o parâmetro de centralidade.

Múltiplas correções podem ser calculadas de modo a melhorar o método preditor-corretor (Gondzio, 1996). Para cada direção adicional é preciso resolver um sistema linear, com a mesma matriz.

2.1 Calculando as direções de busca

O passo principal de cada iteração consiste na resolução de dois sistemas lineares, como (1). Uma vez que eles possuem a mesma matriz, restringiremos a discussão a um dos sistemas.

Eliminando a variável Δz reduzimos o sistema:

$$\begin{bmatrix} -D & A^t \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1} r_a \\ r_p \end{bmatrix}, \quad (2)$$

em que $D = X^{-1}Z$ é uma matriz diagonal. O sistema (2) é referido como sistema aumentado.

Eliminando Δx de (2) obtemos:

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

o qual é chamado de sistema de equações normais.

OBS: Os sistemas de equações normais e aumentado para problemas com variáveis canalizadas têm a mesma estrutura dos sistemas apresentados. Portanto, as ideias desenvolvidas aqui podem ser aplicadas a esses problemas.

2.2 Abordagens para resolver os sistemas lineares

Usar a fatoração de Cholesky no sistema de equações normais para calcular as direções de busca no método de pontos interiores é a abordagem mais utilizada (ver, por exemplo, Adler *et al.* (1989), Czyzyk *et al.* (1999), Gondzio (2012)). No entanto, a matriz fatorada pode ter muito menos esparsidade e é, geralmente, mais mal-condicionada do que a matriz do sistema (2). Resolver o sistema aumentado por métodos diretos é outra opção. Porém, a sequência de pivôs na decomposição depende dos valores numéricos e esta abordagem, embora robusta, é em geral mais cara (Gondzio, 2012).

Além disso, a abordagem direta não pode ser aplicada para algumas classes problemas de grande porte devido à limitações de memória e/ou tempo. Para estes problemas, um método iterativo pré-condicionado para a solução do sistema linear deve ser escolhido (Bergamaschi *et al.* (2007), Bocanegra *et al.* (2007), Oliveira e Sorensen (2005)).

A maioria dos pesquisadores trabalha com o sistema aumentado, uma vez que ele é menos mal-condicionado. Aqui trabalhamos com o sistema de equações normais, pois a matriz é positiva definida, permitindo a utilização do método do gradiente conjugado e o pré-condicionador separador funciona muito bem nas iterações finais onde os sistemas linear são altamente mal-condicionados.

2.3 Pré-condicionador Separador

O pré-condicionador separador (Oliveira e Sorensen, 2005) é calculado como segue: Seja $A = [B\ N]P$ em que $P \in \Re^{n\times n}$ é uma matriz de permutação tal que $B \in \Re^{m\times m}$ é não singular, então

$$ADA^t = [B\ N]PDP^t \begin{bmatrix} B^t \\ N^t \end{bmatrix} = [B\ N] \begin{bmatrix} D_B & 0 \\ 0 & D_N \end{bmatrix} \begin{bmatrix} B^t \\ N^t \end{bmatrix} = BD_BB^t + ND_NN^t\ .$$

O pré-condicionador é dado por $D_B^{-\frac{1}{2}}B^{-1}$ e a matriz pré-condicionada M é como segue:

$$M = D_B^{-\frac{1}{2}}B^{-1}(ADA^t)B^{-t}D_B^{-\frac{1}{2}} = I_m + GG^t,$$

em que $G = D_B^{-\frac{1}{2}}B^{-1}ND_N^{\frac{1}{2}}$.

Próximo a uma solução, pelo menos $n - m$ elementos de D são pequenos. Uma estratégia para determinar B é minimizar $\|D_B^{\frac{1}{2}}B^{-1}ND_N^{-\frac{1}{2}}\|$. Este problema é difícil de resolver, mas pode ser resolvido aproximadamente com uma heurística simples: selecione as primeiras m colunas linearmente independente de AD^{-1} com a menor norma 2.

Esta escolha de colunas tende a produzir matrizes melhores condicionadas quando o método de pontos interiores se aproxima de uma solução, onde os sistemas lineares são altamente mal-condicionado.

2.4 Pré-condicionador Fatoração Controlada de Cholesky

O pré-condicionador Fatoração Controlada de Cholesky (FCC), construído para resolver sistemas gerais definidos positivos (Campos e Birkett, 1998), pode ser visto como uma variação da Fatoração Incompleta de Cholesky.

A fatoração de Cholesky é como segue: $ADA^t = LL^t = \bar{L}\bar{L}^t + R$, em que L representa o fator obtido quando a fatoração é completa, $\bar{L}$ representa o fator obtido quando a fatoração é incompleta e R é uma matriz resto da fatoração incompleta de Cholesky.

A matriz $\bar{L}$ é usada como um pré-condicionador para ADA^t,

$$\bar{L}^{-1}(ADA^t)\bar{L}^{-t} = (\bar{L}^{-1}L)(L^t\bar{L}^{-t}) = (\bar{L}^{-1}L)(\bar{L}^{-1}L)^t.$$

Seja $F = L - \bar{L}$. Substituindo L na última igualdade $\bar{L}^{-1}(ADA^t)\bar{L}^{-t} = (I_m + \bar{L}^{-1}F)(I_m + \bar{L}^{-1}F)^t$.

Note que quando $\bar{L} \approx L$ então $F \to 0$ e, portanto, $\bar{L}^{-1}(ADA^t)\bar{L}^{-t} \to I_m$.

A fatoração controlada de Cholesky é baseada na minimização da norma de Frobenius de F. Logo, quando $||F|| \to 0 \Longrightarrow ||R|| \to 0$. Considere o seguinte problema:

$$\text{Min } \|F\|_F^2 = \sum_{j=1}^{m} c_j \text{ com } c_j = \sum_{i=1}^{m} |l_{ij} - \bar{l}_{ij}|^2,$$

em que l_{ij} são os elementos de L.

Dividindo o segundo somátorio em dois temos:

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

em que t_j é o número de não nulos abaixo da diagonal na j-ésima coluna da matriz ADA^t e η é o número de elementos extras permitidos por coluna na fatoração incompleta.

O primeiro somatório contém todos $t_j + \eta$ elementos diferentes de zero da j-ésima coluna de $\bar{L}$. O segundo tem apenas os elementos restantes do fator completo L que não tem os elementos correspondentes em $\bar{L}$.

Assim, o pré-condicionador pode ser construído usando a seguinte heurística:

- Aumentar η (permitindo mais preenchimento). O termo c_j deve diminuir, porque o primeiro somatório contém mais elementos.
- Escolhendo os $t_j + \eta$ maiores elementos de $\bar{L}$ em valor absoluto para η fixo. Neste caso, os maiores elementos estão no primeiro somatório deixando apenas os menores l_{ij} no segundo somatório.

O pré-condicionador $\bar{L}$ é construído por colunas. Consequentemente, ele necessita somente da j-ésima coluna de ADA^t por vez, evitando os cálculo do sistema de equações normais.

2.5 Pré-condicionador híbrido

A matriz D muda significativamente de uma iteração para outra no método de pontos interiores e torna-se bastante mal-condicionada nas iterações finais. Por esta razão, é difícil encontrar uma estratégia de pré-condicionamento que tenha um bom desempenho ao longo de todas as iterações de pontos interiores.

Em Bocanegra *et al.* (2007) foi proposto aplicar o método do gradiente conjugado para resolver o sistema de equações normais pré-condicionado por uma matriz pré-condicionadora híbrida M.

Esta abordagem assume a existência de duas fases durante as iterações do método de pontos interiores. Na primeira, o pré-condicionador fatoração controlada de Cholesky é usado para construir a matriz M. Após a mudança de fases, a matriz M é construída utilizando o pré-condicionador separador. Em Velazsco *et al.* (2010), foi proposta uma heurística para a mudança de pré-condicionadores. Se o número de iterações necessárias para o método do gradiente conjugado convergir for superior a $\frac{m}{6}$, o parâmetro η na fatoração controlada de Cholesky é aumentado, isto é, $\eta = \eta + 10$. A mudança ocorre quando η excede um valor máximo pré-fixado.

No entanto, esta abordagem pode falhar e não convergir para algumas classes de problemas de programação linear quando a fatoração controlada de Cholesky não é mais eficaz e o pré-condicionador separador ainda não está pronto para ser usado. A fim de melhorar esta abordagem, algumas iterações do algoritmo de ajustamento ótimo para p coordenadas são realizadas um pouco antes do pré-condicionador separador ser aplicado, obtendo um ponto mais próximo da otimalidade. Esta abordagem diminui o *gap* na troca de pré-condicionadores. Para mais detalhes veja Ghidini *et al.* (2012).

3 Implementando o pré-condicionador separador

Esta classe de pré-condicionadores não será competitiva com a abordagem de método direto sem uma implementação cuidadosa. Isto porque uma fatoração LU deve ser calculada sem o conhecimento do

conjunto de colunas independentes antes de seu início. Esta fatoração pode ser muito cara, por dois motivos. Primeiro, porque ela pode gerar muito preenchimento. Segundo, porque pode ser necessário fatorar muitas colunas antes da completar a fatoração, para que as colunas dependentes sejam descartadas. Várias técnicas para a implementação de um código eficiente são discutidas a seguir.

3.1 Soluções inexatas

Uma ideia que vem em mente quando métodos iterativos são usados para resolver sistemas lineares é relaxar a tolerância. Assim, começamos o método de pontos interiores com uma tolerância relaxada de (10^{-4}) e, quando uma iteração não reduz (pelo menos) pela metade o *gap*, $(x^t z)$, a tolerância é alterada para a raiz quadrada do epsilon da máquina. Existe um outro lugar onde podemos aplicar esta ideia. Vimos que para o cálculo das direções de busca, dois sistemas lineares devem ser resolvidos. O segundo sistema pode ser escrito de tal forma que as direções de busca finais já são dadas. Assim, o primeiro sistema linear pode ser resolvido com uma tolerância mais relaxada do que o segundo.

3.2 Mantendo um conjunto de colunas

Uma boa propriedade do pré-condicionador separador é que nós podemos trabalhar com um conjunto selecionado de colunas por algumas iterações. Como consequência, o pré-condicionador fica muito mais barato de calcular nessas iterações. É importante notar que manter a matriz B de iterações anteriores não significa manter o mesmo pré-condicionador, uma vez que D muda a cada iteração. Assim, esta estratégia apresenta diferentes pré-condicionadores a cada iteração, que são muito fáceis de calcular. No entanto, tais pré-condicionadores não têm o melhor conjunto de colunas depois da primeira fatoração de acordo com a heurística. Nos experimentos, mudamos o conjunto sempre que o método iterativo fizer mais que $(\sqrt{m})$ iterações ou quando a solução dada por ele não for precisa.

3.3 Fatoração LU Incompleta

Foi observado na prática que a fatoração LU normalmente gera muito preenchimento. A razão disto é que nenhum procedimento de reordenamento para reduzir o preenchimento pode ser usado, pois as colunas da matriz não são conhecidas até que elas sejam aceitas como linearmente independentes. Aqui discutiremos uma outra possibilidade que não é adotada como uma opção padrão. Ela consiste em calcular uma fatoração LU incompleta. A fatoração incompleta padrão, em que a estrutura dos elementos não nulos da matriz original coincide com estrutura dos elementos não nulos das matrizes triangulares L e U, não fuunciona bem para este problema. Por outro lado, o uso de tolerância para determinar se um elemento deve ser eliminado da matriz parece ser uma abordagem viável. A ideia é eliminar qualquer elemento não nulo menor do que um valor pré-definido. Para uma tolerância cuidadosamente escolhida, esta técnica pode ser muito útil e apresentar melhores resultados. Esta linha de pesquisa merece mais investigação.

3.4 Fatoração LU

Para esta aplicação, a maneira mais econômica para calcular a fatoração LU é trabalhar com a versão de atualização atrasada. Quando uma coluna linearmente dependente aparece, ela é eliminada da fatoração e o método prossegue com a próxima coluna na ordenação. Uma das principais desvantagens de uma implementação não sofisticada do pré-condicionador separador é o excesso de preenchimento

na fatoração LU. Uma boa técnica consiste em interromper a fatoração quando preenchimento excessivo ocorre e reordenar as colunas independentes encontradas até o momento de acordo com o número de elementos não nulos. A fatoração é então iniciada do começo e o processo é repetido até que m colunas independentes sejam encontradas. Na implementação, nós consideramos como preenchimento excessivo, a fatoração que produz mais não nulos que o sistema de equações normais.

3.5 Calcular uma segunda fatoração LU

Uma segunda fatoração é aplicada sobre o conjunto escolhido de colunas independentes utilizando as técnicas convencionais para o cálculo de uma fatoração LU eficiente. Esta abordagem melhora significativamente os resultados de alguns problemas e, por isso, a segunda fatoração é sempre realizada. Como consequência, não é necessário armazenar a U na fatoração que determina B. As seguintes técnicas são as opções padrão para a segunda fatoração LU no nosso código:
(i) As colunas são permutadas de acordo com a ordenação de B^tB, obtida usando grau mínimo.
(ii) Um parâmetro que determina o intervalo de tolerância para escolha do pivô foi incluído.
(iii) Em cada passo da fatoração, a permutação de linha é feita escolhendo como pivô aquele com menos elementos não nulos em sua linha para as colunas restantes da matriz original.
(iv) Encontramos também componentes fortemente conexas para reescrever a matriz na forma bloco triangular.

3.6 Detecção antecipada de colunas dependentes

Uma das dificuldades em determinar o subconjunto de colunas independentes é o número de colunas dependentes visitadas no processo. Uma ideia consiste em verificar se uma coluna é dependente ou não na versão de atualização atrasada da fatoração LU. Se verificarmos que uma coluna candidata já é dependente nas primeiras k colunas, é desnecessário continuar atualizando a coluna candidata.

3.7 Colunas Simbolicamente Dependentes

Colunas simbolicamente dependentes são colunas que são linearmente dependentes quaisquer que sejam os valores numéricos dos elementos não nulos. A ideia é encontrar um conjunto de, digamos, k colunas com elementos não nulos em no máximo $k-1$ linhas. Este conjunto de colunas é simbolicamente dependente. Vamos primeiro considerar uma matriz quadrada por simplicidade. Nesta situação, o problema é equivalente a permutar elementos não nulos na diagonal. Este problema é equivalente a encontrar um *matching* de um grafo bipartido, em que as linhas e colunas formam o conjunto de vértices e as arestas são representadas pelos elementos não nulos. Essa ideia foi usada pela primeira vez por Duff (1981). Em Coleman e Pothen (1987) esta ideia é estendida para matrizes retangulares. As colunas são reordenadas pelo grau e o algoritmo do *matching* aplicado para determinar um conjunto de colunas candidatas, denominadas *colunas chave*, que não são simbolicamente dependentes. Note que o número de colunas independentes antes da k-ésima coluna chave na matriz é, no máximo, $k-1$. Portanto, é possível acelerar a fatoração LU sempre que encontramos $k-1$ colunas numericamente independentes localizadas antes da k-ésima coluna chave. A aceleração é alcançada descartando todas as colunas da atual para a k-ésima coluna chave.

3.8 *Matching* durante a fatoração

Outra forma de evitar operações de ponto flutuante é fazer um *matching* durante a fatoração. Assim, antes de atualizar a coluna verificamos se ela é simbolicamente dependente ou não. Se for, a coluna é descartada e a fatoração continua com a próxima coluna. Esta técnica pode evitar esforço computacional porque um *matching* pode ser feito na matriz original ao invés da fatorada. Além disso, nenhuma operação de ponto flutuante é realizada. Se várias colunas são dependentes, o esforço causado pelas que não são dependentes é compensado reduzindo o tempo total para calcular a fatoração.

3.9 Colunas Simbolicamente Independentes

Colunas simbolicamente independentes são colunas que são linearmente independentes para todos os valores numéricos de seus elementos não nulos. Uma estratégia eficaz consiste em mover as colunas simbolicamente independentes para o início da lista ordenada uma vez que essas colunas necessariamente estarão na fatoração. Então estas colunas podem ser reordenadas a fim de reduzir o número de preenchimento na fatoração LU. Note que as colunas simbolicamente dependentes podem ser ignoradas nesta etapa. Nós usamos abordagens heurísticas para identificar algumas das colunas simbolicamente independentes. Na heurística descrita abaixo, dizemos que a coluna j é a primeira coluna com elemento não nulo na linha i se j contém o primeiro elemento não nulo na linha i no conjunto ordenado. Nós consideramos uma coluna j simbolicamente independente, dado um conjunto ordenado, se pelo menos uma das seguintes regras se aplica:
1. A coluna j é a primeira coluna com elemento não nulo em pelo menos uma linha;
2. A coluna j é a segunda coluna com elemento não nulo na linha i e a primeira coluna com elemento não nulo na linha i é também a primeira coluna com elemento não nulo em pelo menos uma outra linha não presente na coluna j.

3.10 Colunas Chave e Colunas Independentes

Outro uso para as colunas chave é aproximar como seria a estrutura esparsa da matriz B a ser fatorada. Tal informação pode ser usada para reduzir o número de preenchimento na fatoração. Essa ideia funciona bem para alguns problemas, mas piora o desempenho ao calcular o pré-condicionador em outros. Um dos critérios para decidir sobre o uso dessa abordagem é o número de colunas simbolicamente independentes encontrado. Se esse número for próximo ao número total de colunas, as colunas chave darão uma melhor aproximação do padrão de esparsidade de B uma vez que a maioria das colunas na fatoração sejam conhecidas.

3.11 *Merge* de Colunas Simbolicamente Independentes e Dependentes

Depois de reordenar as colunas simbolicamente independentes, é possível reduzir ainda mais o número de preenchimento na fatoração, fazendo um *merge* na lista de colunas simbolicamente independentes e dependentes usando como critério o grau. Isto é permitido desde que as colunas permaneçam simbolicamente independentes. É muito caro verificar se as colunas permanecem simbolicamente independente em cada passo do processo de *merge*. Por isso, usamos a primeira ordenação das colunas como uma heurística barata. Assim, nós movemos para frente na lista uma coluna simbolicamente dependente com grau menor desde que ela permaneça atrás de colunas simbolicamente independentes com índice menor na primeira ordenação.

3.12 Componentes Fortemente Conexas

Esta estratégia é aplicada as colunas chave. Uma vez que as colunas chave são determinadas por um procedimento apropriado (*matching*), uma permutação para calcular as componentes fortemente conexas já está disponível. Dadas as componentes fortemente conexas, suas colunas são reordenadas de acordo com o pré-condicionador separador. Em Oliveira e Sorensen (2005) foi provado que podemos considerar a primeira coluna simbolicamente dependente em sua própria componente conexa estudando somente as linhas desta componente. Todas as colunas com índice menor na ordenação são simbolicamente independentes. Observe que, com esta estratégia, as heurísticas para encontrar colunas simbolicamente independentes podem ser aplicadas em cada bloco diagonal.

4 Experimentos Computacionais

Nós apresentamos vários experimentos numéricos com a abordagem de pré-condicionador híbrida. Estes experimentos foram feitos para ilustrar como as técnicas para calcular o pré-condicionador separador trabalham em conjunto e para determinar quais devem ser adotadas como padrão.

Os procedimentos para resolver os sistemas lineares com o pré-condicionador separador foram implementados em linguagem C e aplicados dentro do código PCx (Czyzyk *et al.*, 1999). Os parâmetros padrão do PCx são usados, exceto as múltiplas correções que foram desligadas. Todas as tolerâncias foram definidas como a raiz quadrada do epsilon da máquina. Os experimentos foram realizados em um processador Intel Core 2 Duo de 64 bits, 2GB e 2.2GHz e o sistema operacional Linux.

Para resolver os sistemas lineares, o critério de parada foi definido como $||r_k|| < 10^{-4}$.

Quando o *gap* de otimalidade é menor que 10^{-5} ou a mudança de fase é detectada, o critério muda para $||r_k|| < 10^{-8}$.

O número máximo de iterações do método dos gradientes conjugados é o tamanho sistema. O conjunto de problemas testes são das coleções NETLIB, QAPLIB e Kennington.

Chamamos de padrão a versão que utiliza a abordagem híbrida de precondicionamento, refatora a matriz B, não cria a matriz B em toda iteração, faz a procura de linha e/ou colunas com um único elemento, utiliza colunas chaves, não faz o *matching*, reordena as colunas linearmente independentes, não considera a fatoração LU incompleta e reordena as colunas durante a decomposição LU.

A Tabela 1 apresenta uma comparação do tempo total de resolução da versão padrão com as versões que consideram uma das técnicas presentes na padrão, descritas na Seção 3, para calcular o pré-condicionador separador. As colunas da tabela têm os seguintes significados:
Coluna **Separ**: somente o pré-condicionador separador é utilizado.
Coluna **NRefat**: a matriz B não é refatorada.
Coluna **B**: a matriz B é criada em toda iteração.
Coluna **Unit**: Linhas e/ou colunas com um único elemento não são procuradas.
Coluna **NChave**: não utiliza as colunas chave.
Coluna **Match**: faz o *matching* durante a LU.
Coluna **LUInc**: a fatoração LU incompleta é considerada.
Coluna **NLI**: as colunas linearmente independentes não são reordenadas.
Coluna **NReord**: o reordenamento das colunas não é feito durante a fatoração LU.

Analisando a Coluna **Separ** e comparando-a com a Coluna **Padrão**, podemos afirmar que ao utilizar o pré-condicionador separador desde a primeira iteração do método de pontos interiores perde-se em eficência na maioria dos problemas (aproximadamente 81% dos problemas). Outros problemas como etamacro, fit1p, pds-01, pds-02, pds-06, pds-10 and pds-20, que são resolvidos antes da troca de

pré-condicionadores na abordagem híbrida, foram testados usando somente o pré-condicionador separador. Os resultados tiveram o mesmo comportamento, sendo que para o problema pds-10 o aumento no tempo chegou a ser 9,96 vezes maior. Além disso, a abordagem padrão é mais robusta, pois ela resolve um número maior de problemas. Estes outros problemas não aparecem na Tabela 1, pois foram considerados somente para esta comparação.

A abordagem que não refatora a matriz é mais lenta que a padrão, principalmente, nos problemas de grande porte. Em torno de 63% dos problemas testados o desempenho piorou. Por exemplo, o problema ste36b teve um aumento no tempo total de resolução de aproximadamente 66%. Este resultado é confirmado pelo número médio de elementos não nulos, que reduz significativamente na matriz refatorada. Para o problema ste36b, essa redução foi em torno de 72%. Um resultado semelhante em relação ao tempo total de resolução acontece quando a matriz B é calculada a cada iteração, em particular para problemas de grande porte também.

Quando a procura por linhas e/ou colunas com um único elemento não nulo não é feita, 46% dos problemas não são resolvidos e os que são resolvidos, o aumento no tempo chegou a ser 6,2 vezes maior (não considerando o problema adlittle).

Utilizar colunas chave produz resultados inconclusos. O desempenho de aproximadamente 43% dos problemas melhora e de 34% dos problemas piora. O maior aumento foi em torno 22%, enquanto que a redução, no melhor caso foi em torno de 38%.

Fazer o *matching* melhora a eficiência do método em aproximadamente 42% dos problemas e piora em torno de 27% deles. Porém, a melhora não é tão significativa. Para a maioria dos problemas de grande porte parece ser aconselhável não realizar o *matching*, uma vez que o tempo chega a ser 22% menor (ste36b problem).

Considerar a fatoração LU Incompleta não trouxe aumento ou diminuição significativo no tempo total de resolução dos problemas. Entretanto, houve perda de robustez, pois em torno de 15% dos problemas (os de maior dimensão) deixaram de ser resolvidos.

A abordagem padrão que reordena as colunas LI é mais rápida e estável que a versão que não faz o reordenamento, uma vez que o desempenho da versão sem reordenamento é pior em aproximadamente 63% dos problemas e 15% dos problemas (os maiores) deixaram de ser resolvidos. Isto confirma a importância dessa técnica.

Quando o reordenamento não é feito durante a decomposição LU também piora o desempenho da abordagem padrão, pois o tempo torna-se maior em 66% dos problemas. No pior caso, o aumento foi em torno de 22%. Além disso, um problema não foi resolvido.

Com relação ao número total de iterações do método de pontos interiores, os resutados são praticamente os mesmos para todas as variações estudadas. Na maioria dos casos em que houve alguma alteração a diferença foi de apenas uma iteração.

5 Conclusões

Neste trabalho, consideramos uma abordagem híbrida de pré-condicionamento para resolver os sistemas lineares oriundos do método de pontos interiores, mais especificamente, o Preditor-Corretor. O pré-condicionador fatoração controlada de Cholesky é usado inicialmente e depois o pré-condicionador separador, o qual tem um bom desempenho próximo da solução. No entanto, uma implementação eficiente do pré-condicionador separador não é trivial e diversas técnicas foram consideradas para tornar a versão padrão competitiva. De acordo com os experimentos realizados, concluímos que a versão padrão é robusta, pois resolve todos os problemas. Entretanto, melhorias ainda precisam ser feitas, uma

Problema	Padrão	Separ	NRefat	B	Unit	NChave	Match	LUInc	NLI	NReord
25fv47	2,66	8,23	2,69	2,67	*	2,67	2,65	2,66	2,64	2,69
adlittle	0	0,01	0	0,01	*	0,01	0,01	0,01	0	0
agg2	0,67	0,80	0,67	0,66	1,47	0,67	0,66	0,67	0,70	0,68
agg3	0,5	0,65	0,50	0,45	1,18	0,50	0,49	0,49	0,49	0,50
bandm	0,15	0,31	0,16	0,15	*	0,16	0,16	0,16	0,16	0,16
blend	0	0,01	0,01	0,01	0	0,01	0,01	0,01	0	0
bnl2	7,32	11,2	7,35	6,13	*	7,38	7,23	7,30	7,38	7,41
boeing1	0,27	0,23	0,28	0,23	*	0,27	0,27	0,27	0,28	0,28
boeing2	0,03	0,03	0,03	0,04	*	0,03	0,03	0,03	0,03	0,03
bore3d	0,02	0,02	0,02	0,02	0,06	0,02	0,02	0,02	0,02	0,03
capri	0,08	0,11	0,09	0,09	*	0,09	0,09	0,09	0,09	0,09
chr20b	15,86	10,04	16,27	14,36	16,15	15,58	16,04	15,91	16,15	16,02
chr20c	12,19	6,78	12,28	12,26	12,74	12,18	12,22	12,24	12,70	12,31
chr22b	16,35	18,04	16,25	16,12	17,90	15,76	16,22	16,35	16,52	17,73
chr25a	43,26	44,62	43,14	43,72	46,12	41,32	43,55	43,37	43,73	48,01
d6cube	2,02	3,73	2,03	2,05	*	2,09	1,93	2,17	1,94	2,04
degen2	0,37	0,32	0,35	0,43	*	0,29	0,35	0,35	0,33	0,36
degen3	8,35	6,54	8,44	13,53	*	5,56	8,32	8,34	7,55	8,50
e226	0,19	0,36	0,18	0,18	*	0,18	0,18	0,18	0,18	0,18
els19	93,8	112,37	94,98	336,61	267,18	58,59	97,39	94,20	123,13	95,19
finnis	0,26	*	0,27	0,19	*	0,26	0,26	0,26	*	0,26
forplan	0,17	0,75	0,17	0,17	0,18	0,17	0,17	0,16	0,17	0,17
hil12	7,65	11,60	7,61	9,54	8,27	6,53	7,65	7,68	8,89	7,65
israel	0,13	0,12	0,13	0,13	*	0,14	0,12	0,13	0,13	0,13
kb2	0	0	0	0,01	0	0	0	0	0	0
maros	2,50	*	2,56	2,49	*	2,51	2,48	2,49	2,82	2,54
nug05	0,03	0,03	0,02	*	0,05	0,03	0,03	0,02	*	0,03
nug06	0,12	0,12	0,12	0,12	0,15	0,11	0,12	0,12	0,12	0,12
nug07	0,48	0,62	0,49	0,58	0,48	0,41	0,48	0,48	0,53	0,50
nug08	1,18	1,76	1,20	1,50	1,25	1,31	1,20	1,17	1,37	1,20
nug12	156,10	313,59	169,31	184,11	147,76	189,64	155,09	*	*	188,36
nug15	1592,32	5585,45	2156,89	1900,52	*	1669,45	1581,81	*	2011,06	1610,35
qap12	151,83	265,37	176,75	189,50	144,55	139,42	151,26	*	168,44	153,12
qap15	4711,82	*	6617,60	4897,61	*	3859,09	*	*	3562,19	*
rou10	1,74	2,63	1,75	2,15	1,80	1,61	1,77	1,75	2,27	1,75
rou20	1769,07	8199,02	1375,91	10557,01	1776,06	1523,41	1766,24	*	2074,52	1798,90
scr12	1,80	1,89	1,81	1,88	8,19	1,74	1,80	1,80	1,86	1,80
scr15	11,62	12,84	11,66	20,75	*	8,56	11,49	11,56	13,61	11,69
scr20	138,56	169,12	138,91	224,79	858,26	92,98	144,43	137,95	169,11	139,76
ste36b	20957,77	*	34725,27	34768,03	*	*	25477,85	*	38422,29	25633,65
stocfor2	2,56	*	2,63	2,11	*	2,57	2,53	2,55	2,59	2,60

Tabela 1: Tempo total de resolução (s)

*: significa que o método falhou

vez que ela resolveu 32% dos problemas testados em menor tempo comparando com todas as outras variantes juntas.

Agradecimentos

Ao CNPq e a FAPESP pelo apoio financeiro.

Referências

Adler, I. Resende, M.G.C., Veiga, G., Karmarkar, N. (1989), An implementation of Karmarkar's algorithm for linear programming, *Mathematical Programming* 44, 297-335.

Bergamaschi, L., Gondzio, J., Venturin, M., Zilli, G. (2007), Inexact constraint preconditioners for linear systems arising in interior point methods, *Computational Optimization and Applications*, 36, 137–147.

Bocanegra, S., Campos, F. F., Oliveira, A. R. L. (2007), Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods, *Computational Optimization and Applications*, 36, 149–164.

Campos, F. F., Birkett, N. R. C. (1988), 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, 126–138.

Coleman, T. F., Pothen, A. (1987), The null space problem II. Algorithms, *SIAM J. Alg. Disc. Meth.*, 8, 544–563.

Czyzyk, J., Mehrotra, S., Wagner, M., Wright, S.J. (1999), PCx an interior point code for linear programming, *Optimization Methods δ Software* 11-2, 397-430.

Duff, I. S. (1981), On algorithms for obtaining a maximum transversal, *ACM Trans. Math. Software*, 7, 315–330.

Ghidini, C. T. L. S., Oliveira, A. R. L., Silva, J., Velazsco, M. I. (2012), Combining a hybrid preconditioner and a optimal adjustment algorithm to accelerate the convergence of interior point methods, *Linear Algebra and its Applications*, 218, 1267–1284.

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

Gondzio, J. (2012), Interior point methods 25 years later, *European Journal of Operational Research*, 218, 587–601.

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

Oliveira, A. R. L., Sorensen, D. C. (2005), 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.

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