COPPE/UFRJ

SOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE COM MÚLTIPLOS LADOS DIREITOS

Valéria Saldanha Motta

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

Orientadores: Nelson Maculan Filho
Luiz Mariano Paes de Carvalho Filho

Rio de Janeiro
Janeiro de 2010

SOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE COM MÚLTIPLOS LADOS DIREITOS

Valéria Saldanha Motta

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

Aprovada por:

Prof. Nelson Maculan Filho, D.Sc.

Prof. Luiz Mariano Paes de Carvalho Filho, D.Sc.

Prof. Adilson Elias Xavier, D.Sc.

Prof. Carlile Campos Lavor, D.Sc.

Prof. Luiz Satoru Ochi, D.Sc.

Prof. Marcia Helena Costa Fampa, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JANEIRO DE 2010

Motta, Valéria Saldanha

Solução de Sistemas Lineares de Grande Porte com Múltiplos Lados Direitos/Valéria Saldanha Motta. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIV, 121 p.: il.; 29, 7cm.

Orientadores: Nelson Maculan Filho

Luiz Mariano Paes de Carvalho Filho

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 117 – 121.

1. Sistemas lineares. 2. Projeção. 3. Subespaço de Krylov. 4. Bl-BiCGStab. I. Maculan Filho, Nelson *et al.*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu pai, Helio Motta,
que foi o exemplo de que estudar
e aprender são os maiores bens
de uma alma.

Agradecimentos

Meu alicerce, meu amparo, minha mãe, Heloisa Ribeiro Saldanha Motta.

Meu ídolo e companheiro de longa data, meu irmão, Maurício Saldanha Motta.

Meu parceirinho, meu calmante natural, meu cãozinho, Fünf.

Minha alegria e suavidade, minhas filhas, Isabella e Ingrid Saldanha Motta Barros de Oliveira.

Meu melhor amigo, meu sagrado amor, meu marido, Gerson Bazo Costamilan.

Meus incentivadores e amigos de jornada, o casal, Kely Diana Villacorta Villacorta e Felipe Antonio Garcia Moreno.

Minha amiga e confidente, Maria de Fátima Cruz Marques.

À toda minha família e amigos que me apoiaram e encorajaram nesta caminhada, muito obrigada!

Meus orientadores, meus amigos, os professores, Nelson Maculan Filho e Luiz Mariano Paes de Carvalho Filho.

Sou muito grata pelos ensinamentos, carinho e compreensão.

Às secretárias e funcionários do Programa de Engenharia de Sistemas e Computação, meu muito obrigado.

Gostaria da agradecer à direção do Instituto Militar de Engenharia, local onde leciono, pela licença que me concederam para fazer doutorado.

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

SOLUÇÃO DE SISTEMAS LINEARES DE GRANDE PORTE COM MÚLTIPLOS LADOS DIREITOS

Valéria Saldanha Motta

Janeiro/2010

Orientadores: Nelson Maculan Filho
Luiz Mariano Paes de Carvalho Filho

Programa: Engenharia de Sistemas e Computação

Sistemas lineares de grande porte aparecem como resultado da modelagem de vários problemas nas engenharias. A busca de métodos para resolução destes sistemas é muito abordada pela Ciência da Computação.
Nosso objetivo foi o de desenvolver e implentar um código para resolver sistemas lineares de múltiplos lados direitos, tal código corresponde ao método do Gradiente Bi-Conjugado Estabilizado em Bloco (Bl-BiCGStab) com precondicionadores.
Para atingir a nossa meta estudamos dez métodos, baseados ou no método de Arnoldi para se gerar bases ortonormais, ou no método da Bi-Ortogonalização de Lanczos. O foco deste estudo foi o de utilizar projeções, ortogonais ou oblíquas, sobre subespaços de Krylov apropriados. Como resultado deste estudo obtivemos nove proposições.
Fizemos a implementação do método do Bl-BiCGStab com precondicionadores, usando o programa MATLAB. Assim, concluímos que o método proposto se apresenta como uma opção para resolução de sistemas lineares de grande porte com múltiplos lados direitos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SOLUTION OF LARGE LINEAR SYSTEMS WITH MULTIPLE HIGHT-HAND SIDES

Valéria Saldanha Motta

January/2010

Advisors: Nelson Maculan Filho

Luiz Mariano Paes de Carvalho Filho

Department: Systems Engineering and Computer Science

Large linear systems appear as a result of modelling several problems in engineering. The search for methods to solve these systems is much discussed in Computer Science.

Our goal was to develop and implement a code to solve linear systems of equations with multiple right-hand sides, this code corresponds to the Block Bi-Conjugate Gradient Stabilized (Bl-BiCGStab) with preconditioner method.

To achieve our goal we studied ten methods, based or on Arnoldi's method to generate orthonormal basis and or on Bi-Orthogonalization of Lanczos method. The focus of this study was to use orthogonal or oblique projections over suitable Krylov subspaces. As a result of this study we obtained nine propositions.

We implemented the Bl-BiCGStab with preconditioner method using the MATLAB program. Thus, we concluded that the proposed method offers as an option for solving large linear systems with multiple right-hand sides.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Sistemas lineares de grande porte aparecem como resultado da modelagem de vários problemas nas engenharias. A busca de métodos para resolução destes sistemas é muito abordada pela Ciência da Computação.

Consideremos um sistema linear,

$$AX = B, \tag{1.1}$$

onde A é uma matriz real de ordem n, não-simétrica, X e B matrizes reais de ordem $n \times s$, com $s << n$.

Uma maneira de resolver (1.1) seria solucionar s sistemas lineares, usando, por exemplo, uma decomposição LU da matriz A. O custo computacional seria o da decomposição de A, $O(n^3)$, o que poderia ser problemático do ponto de vista numérico, dependendo do valor de n.

Nosso objetivo é o estudo da resolução de sistemas lineares com múltiplos lados direitos. Para tal, iremos estudar alguns métodos de resolução de sistemas, dando maior enfoque ao método do Gradiente Bi-Conjugado Estabilizado em Bloco (Bl-BiCGStab) [7]. Faremos a implementação de tal método usando o programa MATLAB. Temos como diferencial o acréscimo de precondicionadores.

O enfoque teórico escolhido foi o de determinar quais os espaços de Krylov envolvidos, que tipo de projeção é usada, ortogonal ou oblíqua, e obter a fórmula matricial dos operadores de projeção nos diversos métodos iterativos abordados. Chamaremos de *teorema* os resultados clássicos da literatura e de *proposição* o que apresentaremos e demonstraremos neste trabalho. Neste contexto, foram desenvolvidas 19 proposições.

No Capítulo 2, faremos a análise e síntese de trabalhos publicados. Os trabalhos foram selecionados, considerando sua relevância e data de publicação, priorizando os mais recentes.

Nos Capítulos 3 e 4, vamos apresentar a parte teórica, definições e teoremas de Álgebra Linear, que serão importantes para o estudo de diversos métodos iterativos abordados. No capítulo 4, teremos como resultado a Proposição 4.1.

No Capítulo 5, apresentaremos o Algoritmo da Bi-Ortogonalização de Lanczos [13], Algoritmo 5.1, e, a partir dele, vamos obter as Proposições 5.1 e 5.2. O método da Bi-Ortogonalização de Lanczos [13] será abordado, usando-se o enfoque de projeções. A partir daí, vamos obter a Proposição 5.3, que explicita o operador de projeção usado por este método.

No Capítulo 6, apresentaremos o Algoritmo de Arnoldi [14] e teremos como resultado a Proposição 6.1, que mostra o operador de projeção ortogonal do método de Arnoldi [14].

No Capítulo 7, será abordado o método do Resíduo Mínimo Generalizado (GMRES) [15]. Antes, porém, vamos apresentar métodos precursores do GMRES [15]. São eles: o método do Resíduo Conjugado Generalizado (GCR) [16] e o método da Ortogonalização Completa (FOM) [14]. Serão apresentadas as Proposições 7.1 a 7.6, que estão relacionadas aos métodos estudados neste capítulo.

No Capítulo 8, vamos apresentar o método do Gradiente Bi-Conjugado Estabilizado [17]. Antes, porém, apresentaremos os métodos Gradiente Bi-Conjugado [18] e do Gradiente Conjugado Quadrado [19]. Como resultado, teremos as Proposições 8.1 a 8.5.

No Capítulo 9, o método a ser apresentado é o do Gradiente Bi-Conjugado Estabilizado em Bloco (Bl-BiCGStab) [7]. Faremos, previamente, o estudo do método do Gradiente Bi-Conjugado em bloco (Bl-BiCG) [20]. Como resultados decorrentes deste estudo serão apresentadas as Proposições 9.1 a 9.5.

No Capítulo 10, faremos a implementação do código do Bl-BiCGStab [7] com precondicionadores, desenvolvido em MATLAB. Também faremos a implementação do Bl-GMRES [21], acrescentando precondicionadores. Teremos quinze tabelas de dados, onde analisaremos a performance do Bl-BiCGStab [7] com precondicionadores. Vamos compará-lo com um método em bloco, o Bl-GMRES [21] com precondiciona-

dores. Mostraremos que o Bl-BiCGStab [7] com precondicionadores é mais eficiente que o Bl-GMRES [21] com precondicionadores. Vamos também comparar o Bl-BiCGStab [7] com precondicionadores com o GMRES [15] e com o BiCGStab [17] aplicados s vezes. Como resultado obteremos performances equivalentes.

Capítulo 2

Estado da Arte

2.1 Introdução

Neste capítulo, vamos expor uma visão geral sobre alguns métodos para resolução de sistemas lineares com múltiplos lados direitos. Vamos considerar a equação (1.1), com as mesmas hipóteses da Introdução desta tese.

Usaremos o produto interno de Frobenius, $(X, Y)_F = tr(X^T Y)$ e, consequentemente, a norma gerada por este produto interno, $\|X\|_F = [tr(X^T X)]^{1/2}$. A solução inicial será dada por uma matriz de ordem $n \times s$, X_0 ; o resíduo inicial será $R_0 = B - AX_0$.

Na seção (2.2), faremos a análise de alguns artigos relevantes. Nesta, ao se mencionar sistema linear em bloco, devemos nos reportar à equação (1.1), munidos das hipóteses aqui apresentadas.

Na seção (2.3), será apresentada uma síntese dos artigos abordados. Vamos agrupar alguns conceitos segundo vários critérios, para que possamos obter uma visão geral dos métodos desenvolvidos que se correlacionam com esta tese.

2.2 Análise de artigos selecionados

Apresentaremos a análise de alguns artigos selecionados. Tais artigos foram escolhidos levando-se em conta a relevância a partir do número de citações e importância para o desenvolvimento desta tese, outro fator de escolha foi o tempo de publicação, escolhemos os mais recentes.

Se forem mencionados sistemas lineares em bloco, estamos nos referindo à equação

(1.1), munida das hipóteses apresentadas na introdução deste capítulo.

2.2.1 *An Iterative Method for Nonsymmetric Systems with Multiple Right-Hand Sides* [1]

Seja o sistema linear (1.1), sob as mesmas hipóteses que estão na introdução deste capítulo.

O artigo [1], em suas primeiras seções, descreveu os métodos em bloco existentes para se resolver sistemas lineares, destacando-se o método do Resíduo Mínimo Generalizado em Bloco (Bl-GMRES) [21]. Apresentou-se, então, o algoritmo do Resíduo Mínimo Generalizado Modificado Híbrido (MHGMRES) [1].

O MHGMRES parte de uma solução inicial para o sistema (1.1), X_0, e calcula o resíduo inicial, R_0. Tendo R_0, um novo sistema (*seed system*) é gerado e resolvido empregando-se o GMRES [15]; a solução do sistema original é atualizada como um todo; gera-se um novo sistema, e assim sucessivamente. Este algoritmo apresenta uma boa performance quando todos os lados direitos do sistema não estão disponíveis ao se iniciar a sua resolução.

Alguns resultados teóricos foram obtidos; por exemplo, uma cota superior para a norma do resíduo, a cada iteração. Os autores finalizaram com os resultados numéricos obtidos com a implementação do método para algumas matrizes da coleção de matrizes esparsas Harwell-Boeing [22], aplicando o algoritmo sobre diferentes números de lados direitos, $s = 5, 10, 15, 20$.

2.2.2 *Breakdowns in the Implementation of the Lanczos Method for Solving Linear Systems* [2]

O artigo [2] descreveu o método de Lanczos [13] usando, para isto, polinômios ortogonais e funcionais lineares. Desta forma, foi obtido o polinômio ortogonal que descreve o resíduo associado a um sistema linear $Ax = b$, onde A é uma matriz real de ordem n e x e b, vetores de ordem n.

Foi exibida uma classificação quanto às possíveis falhas, *breakdowns*, do método de Lanczos [13].

O *true breakdown* ocorre quando o polinômio ortogonal, P_k, dado por $r_k = P_k(A)r_0$,

não existe. Neste caso, para se evitar este tipo de *breakdown* no algoritmo, os autores sugeriram três estratégias:

- identificar este tipo de *breakdown*;

- determinar o grau do próximo polinômio ortogonal existente;

- pular os polinômios não existentes.

O *ghost breakdown* ocorre quando nenhum dos termos usados nas fórmulas de recorrência empregadas no algoritmo existe. Para se evitar este tipo de *breakdown*, deve-se identificar e pular os polinômios durante a execução do algoritmo.

Existe, ainda, o *near breakdown*, que pode ser do tipo *true* ou *ghost*. Este ocorre quando algum dos termos usados nas recorrências não é exatamente igual a zero, mas tão próximo, que serão gerados erros de aproximação que comprometerão seriamente o algoritmo. Uma forma de se evitar esta falha é pular os polinômios que geram este tipo de *breakdown*. Dependendo do caso, uma outra estratégia também pode ser aplicada, que é a mudança do produto interno, consequentemente da norma utilizada ao longo das recorrências envolvidas.

2.2.3 *A Lanczos-Type Method for Multiple Starting Vectors* [3]

O objetivo deste artigo [3] é apresentar uma extensão do método de Lanczos [13]. O método será iniciado com dois blocos de vetores iniciais para os subespaços de Krylov, à direita e à esquerda, de uma determinada matriz real de ordem n. Este método é diferente do método de Lanczos [13] em bloco e dos métodos em bloco em geral, pois os vetores em bloco iniciais não precisam ter a mesma dimensão. Foram definidos dois processos que foram implementados no algoritmo proposto. O *look-ahead*, que consiste em continuar o algoritmo quando um *breakdown* ou um *near-breakdown* ocorre, aplicando uma relaxação em alguma condição do método, por exemplo, na biortogonalização. O outro processo é a deflação, que consiste em identificar e ignorar vetores linearmente dependentes nos subespaços de Krylov em bloco.

Os autores propuseram três desafios, no sentido de obter um algoritmo robusto. São eles:

- Incluir o processo de deflação no algoritmo, para detectar e deletar vetores linear-

mente dependentes;

- Tratar vetores em bloco de diferentes ordens pelo algoritmo;

- Incorporar técnicas de *look-ahead* para se evitar os possíveis *exact breakdowns* e *near-breakdowns*.

A partir de então, os autores apresentaram argumentos teóricos para a construção dos passos do algoritmo de Lanczos [13] com deflação e *look-ahead*.

Após apresentarem o algoritmo em si, alguns teoremas foram expostos e demonstrados, concluindo, assim, o artigo.

2.2.4 *Block Bidiagonalization Methods for Solving Nonsymetric Linear Systems with Multiple Right-Hand Sides* [4]

Neste artigo [4], foram apresentados dois algoritmos de bidiagonalização em bloco, Bidiagonalização de Lanczos em Bloco (Bl-BLanczos) e Bidiagonalização do Resíduo Mínimo em Bloco (Bl-BMinres), para resolução de sistemas lineares de grande porte, não-simétricos, com múltiplos lados direitos.

Nos algoritmos propostos pelo autor, o sistema linear (1.1) foi substituído pelo sistema linear simétrico abaixo,

$$\hat{A}\hat{X} = \hat{B}, \tag{2.1}$$

onde

$$\hat{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}, \quad \hat{X} = \begin{bmatrix} Y \\ X \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix} \tag{2.2}$$

são matrizes em bloco de ordens $2n \times 2n$, $2n \times s$ e $2n \times s$, respectivamente.

Foram apresentados teoremas e lemas que embasam teoricamente os algoritmos apresentados.

Os métodos Bl-BLanczos e Bl-BMinres [4] foram comparados não só numericamente, mas também com os métodos do Gradiente Biconjugado em Bloco (Bl-BCG) [20] e do Resíduo Mínimo Generalizado em Bloco (Bl-GMRES) [21]. Os métodos propostos também foram comparados com os métodos da Bidiagonalização de Lanczos (BLanczos) [23] e da Bidiagonalização do Resíduo Mínimo (BMinres) [23] aplicados aos s lados direitos dos sistema (1.1). Não foram usados quaisquer precondicionadores no sistema estendido (2.1) a ser resolvido.

O resultado dos experimentos numéricos mostrou que os algoritmos propostos, Bl-

BLanczos e Bl-BMinres [4], são mais eficientes para resolução de sistemas lineares com múltiplos lados direitos, quando comparados com BLanczos, BMinres, Bl-BCG [20] e Bl-GMRES [21].

2.2.5 *The breakdowns of BiCGStab* [5]

Este artigo [5] tem como propostas classificar os possíveis tipos de falhas (*breakdowns*) do algoritmo do BiCGStab [24] e, também, ver como um parâmetro externo ao algoritmo pode afetar estas falhas.

Tomando um sistema linear, $Ax = b$, onde A é uma matriz real de ordem n, e, x e b, vetores de ordem n, os três tipos de falhas que podem ocorrer com o BiCGStab [17] são *pivot breakdown*, *Lanczos breakdown* e *stabilization breakdown*, que serão definidas posteriormente. Segue o algoritmo.

Inicialização: Escolha $\mathbf{x}_0^0 \in \mathbb{C}^n$ e seja $\mathbf{w}_0^0 := \hat{\mathbf{w}}_0^0 := b - A\mathbf{x}_0^0$. Escolha $\tilde{\mathbf{y}}_0 \in \mathbb{C}^n$ tal que $\tilde{\delta}_0 := \tilde{\mathbf{y}}_0^H \hat{\mathbf{w}}_0^0 \neq 0$ e $\tilde{\delta}_0' := \tilde{\mathbf{y}}_0^H A_0^0 \neq 0$.

Recursividade: Para $k = 0, 1, 2, \ldots$

$$\omega_k := \frac{\tilde{\delta}_k}{\tilde{\delta}_k'}, \tag{2.3}$$

$$\mathbf{w}_{k+1}^k := \mathbf{w}_k^k - A\hat{\mathbf{w}}_k^k \omega_k, \tag{2.4}$$

$$\chi_{k+1} := \frac{(A\mathbf{w}_{k+1}^k)^H \mathbf{w}_{k+1}^k}{\|A\mathbf{w}_{k+1}^k\|^2}, \tag{2.5}$$

$$\mathbf{w}_{k+1}^{k+1} := \mathbf{w}_{k+1}^k - A\mathbf{w}_{k+1}^k \chi_{k+1}, \tag{2.6}$$

$$\mathbf{x}_{k+1}^{k+1} := \mathbf{x}_k^k + \hat{\mathbf{w}}_k^k \omega_k + \mathbf{w}_{k+1}^k \chi_{k+1}, \tag{2.7}$$

$$\tilde{\delta}_{k+1} := \tilde{\mathbf{y}}_0^H \mathbf{w}_{k+1}^{k+1}, \tag{2.8}$$

$$\psi_k := \frac{-\tilde{\delta}_{k+1}}{\tilde{\delta}_k' \chi_{k+1}}, \tag{2.9}$$

$$\hat{\mathbf{w}}_{k+1}^{k+1} := \mathbf{w}_{k+1}^{k+1} - (\hat{\mathbf{w}}_k^k - A\hat{\mathbf{w}}_k^k \chi_{k+1})\psi_k, \tag{2.10}$$

$$\tilde{\delta}_{k+1}' := \tilde{\mathbf{y}}_0^H A\hat{\mathbf{w}}_{k+1}^{k+1}. \tag{2.11}$$

Finalização: Se $\mathbf{w}_{k+1}^{k+1} = 0$, então $\mathbf{x}_{k+1} := \mathbf{x}_{k+1}^{k+1}$ é a solução do sistema. Se $\mathbf{w}_{k+1}^k = 0$, então $\mathbf{x}_{k+1}^k := \mathbf{x}_k^k + \hat{\mathbf{w}}_k^k \omega_k$ é a solução do sistema.

As falhas do algoritmo foram caracterizadas como:

- *Pivot breakdown*, $\tilde{\delta}_{k+1}' = 0$, como consequência ω_{k+1} pode se tornar infinito.

- *Lanczos breakdown*, $\tilde{\delta}_{k+1} = 0$, a base fica estagnada, ou seja, com a mesma dimensão.

- *Stabilization breakdown*, $\chi_k = 0$, ψ_k pode se tornar infinito.

Como vários sistemas lineares vêm da discretização de alguns PDEs, e estes envolvem um parâmetro, consideremos como γ tal parâmetro. Os autores correlacionaram o valor de γ com o do parâmetro γ_c, que é o valor do parâmetro obtido no momento do *breakdown*. Concluíram que, no caso dos *Lanczos* e *stabilization breakdowns*, a perda de acurácia numérica é descontínua quando consideram $|\gamma - \gamma_c|$ e, para o *pivot breakdown*, é aproximadamente inversamente proporcional a $|\gamma - \gamma_c|$.

2.2.6 *A priori error bounds on invariant subspace approximations by block Krylov subspaces* [6]

O principal ponto deste artigo [6] é estabelecer uma estimativa de erro a priori para métodos de subespaços de Krylov em bloco.

Considerando um subespaço de Krylov, sem ser em bloco, $\mathcal{K} = K_m(A, x)$, o Lema 1.1 [6] mostra que em $\mathcal{K}$ temos várias informações sobre um autovetor v. Este lema se utiliza do seno entre um subespaço e um vetor para concluir este fato.

Os autores estendem o Lema 1.1 [6] para o caso de um subespaço de Krylov em bloco, $\mathbf{K} = K_m(A, X) = [X, AX, \ldots, A^{m-1}X]$. Para tal, generalizam a definição da função seno, e uma série de proposições teóricas são apresentadas e demonstradas.

Obtêm, assim, um limite para o erro (*error bound*) para o seno entre o subespaço de Krylov em bloco, $\mathbf{K}$, e o subespaço invariante de A desejado, V_1.

Os autores concluem estabelecendo relações entre suas estimativas e as apresentadas por Saad [25].

2.2.7 *A Block Version of BICGSTAB for Linear Systems with Multiple Right-Hand Sides* [7]

Consideremos o sistema linear (1.1), sob as mesmas hipóteses dadas.

Os autores deste artigo [7] apresentaram várias definições e proposições, evidenciando alguns tópicos teóricos de polinômios de matrizes.

Construíram o algoritmo de uma versão em bloco do Gradiente Bi-Conjugado Estabilizado e demonstraram alguns resultados a respeito desse algoritmo, usando a teoria anteriormente apresentada.

Finalizaram o artigo com experimentos numéricos, utilizando algumas matrizes da coleção de matrizes esparsas Harwell-Boeing [22] e aplicando o Bl-BiCGStab [7] sobre os lados direitos, $s = 5, 10$. Obtiveram comparações entre o Bl-BiCGSTAB [7] e os seguintes métodos: Bl-QMR, Bl-GMRES [21] e Bl-BiCG. Concluíram que o Bl-BiCGStab [7] é vantajoso em relação ao Bl-GMRES [21] e ao BiCGStab [17] aplicado sobre os s lados direitos.

2.2.8 *Global FOM and GMRES algorithms for matrix equations* [8]

Partindo de um sistema linear em bloco (1.1), este artigo [8] apresentou duas novas alternativas de resolução deste sistema: os métodos da Ortogonalização Completa Global (Gl-FOM) e do Resíduo Mínimo Generalizado Global (Gl-GMRES). Tais métodos que utilizam, respectivamente, projeções ortogonal e oblíqua do resíduo inicial sobre subespaços de Krylov em bloco.

Os autores descreveram o algoritmo de Arnoldi Global (Gl-Arnoldi), que utiliza projeção ortogonal para gerar uma base ortonormal para um subespaço de Krylov em bloco. É importante observar que a matriz de Hessenberg, H_k, gerada pelo Gl-Arnoldi é de ordem k, enquanto que a gerada pelo método de Arnoldi em Bloco (Bl-Arnoldi) [26] é de ordem ks.

Seguem os algoritmos Gl-FOM e Gl-GMRES, juntamente com alguns teoremas e lemas a respeito deles. Foi obtida uma desigualdade ((4.14) [8]) que relaciona as normas dos resíduos gerados pelos dois métodos desenvolvidos, bem como a interpretação geométrica desta relação.

Para ambos os métodos, a necessidade de memória aumenta a cada iteração executada. Uma tentativa de solucionar este problema é reiniciar os algoritmos a cada m iterações. Pode-se, ainda, truncar o Gl-Arnoldi, usando-se uma Ortogonalização de Gram-Schmidt Global parcial.

O algoritmos do Gl-GMRES e do Bl-GMRES [21] foram comparados, levando-se em consideração o custo das principais operações envolvidas.

Ainda como uma contribuição, o Gl-GMRES foi aplicado na resolução da equação de Lyapunov, $AX + XA^T + B = 0$, gerando um novo algoritmo.

A partir de experimentos numéricos com matrizes da coleção Harwell-Boeing [22] e

s lados direitos, $s = 5, 10, 20, 30, 40$, os autores concluíram que o Gl-GMRES é mais eficiente do que aplicar s vezes o GMRES [15] para cada lado direito do sistema linear (1.1).

2.2.9 *Block Krylov Space Methods for Linear Systems With Multiple Right-hand Sides: an Introduction* [9]

Consideremos o sistema linear (1.1) e as hipóteses feitas na introdução deste capítulo.

Este artigo de Gutknecht [9] trata, principalmente, da parte teórica relacionada a subespaços de Krylov em bloco e aborda os métodos de Arnoldi em Bloco (Bl-Arnoldi) [26] e Resíduo Mínimo Generalizado em Bloco (Bl-GMRES) [21].

Os métodos de Krylov em bloco, quando comparados aos que não são em bloco, têm como vantagens obter espaços de busca maiores para as soluções aproximadas e poder computar simultaneamente vários produtos matriz-vetor. Como desvantagens, temos que os resíduos gerados pelos métodos em bloco são, possivelmente, linearmente dependentes. Há a necessidade de muita memória, e algumas constantes matriciais usadas durante os algoritmos se tornam não inversíveis, gerando problemas ao longo dos algoritmos.

Muitos dos algoritmos em bloco podem ser obtidos por generalização dos métodos existentes, porém, deve ser dada uma atenção à estabilidade dos algoritmos em bloco. Um outro ponto importante é que estes algoritmos devem ser implementados usando-se precondicionadores, caso contrário, irão convergir muito lentamente.

Há um esclarecimento importante em relação à descrição dos subespaços de Krylov em bloco, ligado à definição de grau destes subespaços. Muitos artigos fazem a generalização da combinação linear de matrizes, sem levar em consideração qual é a principal característica das constantes. Como cada matriz tem, por exemplo, s colunas, os escalares desta combinação linear serão matrizes de ordem s. Os métodos onde os escalares não são matriciais devem ser chamados, na verdade, de métodos globais e não de métodos em bloco.

Quando se trabalha com sistemas lineares com múltiplos lados direitos, a possibilidade de que alguns resíduos sejam linearmente dependentes deve ser resolvida, reduzindo-se o números de lados direitos; o nome deste processo é deflação. A defla-

ção pode ser feita ao se iniciar o algoritmo ou em outro passo qualquer. "A deflação não depende de um destes s sistemas lineares em particular, mas da dimensão do subespaço gerado pelos s resíduos"[9].

A partir da teoria desenvolvida no texto, o autor analisou o algoritmo do Bl-GMRES [21]. Considerou o algoritmo de Bl-Arnoldi [26], visto a importância da base ortogonal em bloco que é construída por este algoritmo e que posteriormente será usada pelo algoritmo Bl-GMRES [21]. A discussão a respeito do Bl-GMRES [21] abordou a necessidade de memória e o custo computacional. Algumas tabelas foram geradas, obtendo-se dados relevantes a respeito deste algoritmo.

Posteriormente, os dois algoritmos já citados foram deflacionados e analisados a partir desta mudança.

O artigo foi finalizado com a apresentação de dados de experimentos numéricos desenvolvidos pelo autor.

2.2.10 *Convergence properties of some block Krylov subspace methods for multiple linear systems* [10]

Seja o sistema linear (1.1). O artigo [10] apresentou alguns resultados sobre a convergência dos seguintes métodos de Krylov em bloco: Gl-GMRES e Gl-FOM [8]. Para a obtenção de tais resultados, o estudo foi feito em duas grandes classes de métodos: a do Resíduo Mínimo Global (Gl-MR), que é equivalente, ao Gl-GMRES e a do Resíduo Ortogonal Mínimo Global (Gl-OR), equivalente ao Gl-FOM.

Os autores trabalharam com o produto matricial definido abaixo em muitas de suas demonstrações.

Definição 2.1 *Sejam $A = [A_1, A_2, \ldots, A_p]$ e $B = [B_1, B_2, \ldots, B_l]$ matrizes de dimensões $n \times ps$ e $n \times ls$, respectivamente, onde A_i e B_j $(i = 1, \ldots, p;\ j = 1, \ldots, l)$ são matrizes de ordem $n \times s$. Então a matriz $A^T \diamond B$ de ordem $p \times l$ é definida por:*

$$A^T \diamond B = \begin{bmatrix} < A_1, B_1 >_F & < A_1, B_2 >_F & \ldots & < A_1, B_l >_F \\ < A_2, B_1 >_F & < A_2, B_2 >_F & \ldots & < A_2, B_l >_F \\ \vdots & \vdots & \vdots & \vdots \\ < A_p, B_1 >_F & < A_p, B_2 >_F & \ldots & < A_p, B_l >_F \end{bmatrix}. \tag{2.12}$$

Observações

1. Se $s = 1$, então $A^T \diamond B = A^T B$.

2. Se $s = 1$, $p = 1$ e $l = 1$, então dados $A = u \in \mathbb{R}^n$ e $B = v \in \mathbb{R}^n$, temos $A^T \diamond B = u^T v \in \mathbb{R}$.

3. A matriz $A = [A_1, A_2, \ldots, A_p]$ é F-ortonormal se e somente se $A^T \diamond A = I_p$.

4. Se $X \in \mathbb{R}^{n \times s}$, então $X^T \diamond X = \|X\|_F^2$.

Os teoremas e proposições obtidas se utilizam do produto de Kronecker e do complemento de Shur, além do produto definido (2.12). Foram obtidas aproximações e relações entre os resíduos do Gl-MR e do Gl-OR. Como um resultado bastante interessante, podemos destacar os resíduos e soluções aproximadas que foram obtidas empregando o complemento de Shur para o Gl-MR e o Gl-OR.

2.2.11 *The Block Grade of a Block Krylov Space* [11]

Os autores fizeram um estudo teórico sobre o subespaço de Krylov em bloco. A importância deste estudo se deve ao fato de que estes subespaços são fundamentais quando tratamos de alguns dos principais métodos iterativos para resolução de sistemas lineares com múltiplos lados direitos.

A caracterização do grau do subespaço foi apresentada na Proposição 1 [11]. Também foi apresentada uma série de definições equivalentes para este conceito. Teoricamente, o grau do subespaço de Krylov é o número máximo de iterações para se chegar à solução exata de um sistema linear. Os métodos iterativos geram soluções aproximadas em um número menor de iterações.

Foram descritos os subespaços de Krylov em bloco e o grau em bloco deste subespaço. Seguiram, daí, teoremas e lemas de caracterização, equivalência e relações entre os conceitos expostos. A patir de então, demonstraram que a solução exata de um sistema linear em bloco pertence a um subespaço de Krylov em bloco.

Usando a teoria desenvolvida ao longo do artigo [11], a base do subespaço de Krylov e os polinômios mínimos em bloco associados a uma matriz A foram caracterizados.

2.3 Síntese dos artigos selecionados

Ao se abordar um sistema linear em bloco (1.1), podemos analisá-lo usando diferentes enfoques e, a partir de cada um deles, obter uma classificação.

Há, no entanto, uma preocupação comum: evitar a geração de vetores linearmente dependentes pelo método escolhido. Os métodos estudados usam como espaço de busca um subespaço de Krylov. Se forem obtidas direções linearmente dependentes, o grau do subespaço fica comprometido, consequentemente, a solução aproximada também. Nos artigos selecionados, aparecem duas estratégias para se evitar este tipo de problema: a deflação e o *look-ahead*. A deflação consiste em reduzir o número de lados direitos do sistema (1.1), o que acarreta uma diminuição no número de operações matriz-vetor, levando em conta os resíduos que são linearmente dependentes. O *look-ahead* consiste em continuar o algoritmo quando ocorre um *breakdown*, por exemplo, relaxando a condição de bi-ortogonalização no caso do algoritmo de Lanczos [13]. De alguma forma, deve-se pular os vetores que causam problema no algoritmo.

Considerando a equação (1.1), uma forma de diferenciar os métodos iterativos utilizados na resolução deste sistema é pensar qual o tipo de projeção de resíduo que é feito sobre um determinado subespaço de Krylov. Usando a projeção ortogonal, temos os seguintes métodos: Gl-FOM [8] e Bl-Arnoldi [26]. Se optamos pela projeção oblíqua, temos: MHGMRES [1], Bl-BMinres [4], Bl-BiCGStab [7], Gl-GMRES [8], Bl-GMRES [21], Bl-BiCG [20], Bl-Lanczos [23] , BMinres [23], BiCGStab [24] e Lanczos [13].

Podemos diferenciar os métodos a partir da forma como o algoritmo vai gerar a solução aproximada do sistema (1.1). Os métodos em bloco vão iterar os s sistemas a serem resolvidos simultaneamente. Como exemplos, temos: Bl-BMinres [4], Bl-BiCGStab [7], Bl-GMRES [21], Bl-BiCG [20], Bl-Lanczos [23] e Bl-Arnoldi [26]. Outra característica é que a matriz de Hessenberg gerada tem ordem ns. Os métodos globais ocorrem quando projetamos globalmente, o resíduo sobre um subespaço de Krylov, de forma ortogonal ou oblíqua, . Alguns destes métodos são: Gl-FOM [8] e Gl-GMRES [8]. Nestes algoritmos, a matriz de Hessenberg tem dimensão s . Há os métodos que partem da resolução de um sistema inicial, os *seed systems*.

Os outros resíduos são projetados no subespaço de Krylov gerado pela resolução do *seed system*, por exemplo, o MHGMRES [1]. Este método também é chamado de método híbrido, pois é um método em bloco baseado no GMRES [15], que usa *seed system* e a aproximação de Richardson.

A partir da nossa pesquisa bibliográfica, podemos constatar que existem poucos artigos sobre Bl-BiCGStab [7]. Por este motivo, preferimos trabalhar com este método, aplicado a sistemas lineares com múltiplos lados direitos de grande porte.

Capítulo 3

Operadores de Projeção

3.1 Introdução

Neste capítulo serão tratadas algumas definições e teoremas de Álgebra Linear, utilizados nos capítulos seguintes. Tais conceitos serão aplicados a alguns métodos iterativos de resolução de sistemas lineares.

Foram feitas demonstrações de alguns teoremas, porque ou as provas encontradas na literatura não eram bastante claras, ou a técnica utilizada na demonstração era interessante para os tópicos a serem desenvolvidos. Os teoremas que não foram demonstrados apresentam referências relativas às demonstrações.

Os pontos principais deste capítulo são tratar as projeções, ortogonal ou oblíqua, sob o ponto de vista matricial, e, também, sua capacidade de minimização.

3.2 Imagem e Núcleo de uma Projeção

Definição 3.1 *Considere P uma transformação linear, P é dita uma projeção, $P : \mathbb{C}^n \to \mathbb{C}^n$ se $P^2 = P$, P é também chamada* ***idempotente****.*

Definição 3.2 *Seja I uma transformação linear, $I : \mathbb{C}^n \to \mathbb{C}^n$, com $I(x) = x$, I é chamada* ***transformação identidade****.*

Definição 3.3 *Considere T uma transformação linear, $T : \mathbb{C}^n \to \mathbb{C}^n$, temos:*

1. *O* ***núcleo*** *de T, $N(T)$, é dado por:*

$$N(T) = \{x \in \mathbb{C}^n | T(x) = 0\}.$$

2. *O* ***domínio*** *de* T*,* $D(T)$*, é dado por:*

$$D(T) = \{x \in \mathbb{C}^n | \exists y \in \mathbb{C}^n, T(x) = y\}.$$

3. *A* ***imagem*** *de* T*,* $Im(T)$*, é dada por:*

$$Im(T) = \{y \in \mathbb{C}^n | \exists x \in \mathbb{C}^n, T(x) = y\}.$$

Teorema 3.1 *Seja* P *uma projeção, temos então que:*

1. $N(P) = Im(I - P)$*;*
2. $N(P) \cap Im(P) = \{0\}$*;*
3. *Se* $x \in Im(P) \Rightarrow P(x) = x$*.*

Prova [26].

Teorema 3.2 *O espaço* $\mathbb{C}^n$ *pode ser decomposto na seguinte soma direta:*

$$\mathbb{C}^n = N(P) \oplus Im(P),$$

ou seja,

$$\forall x \in \mathbb{C}^n, \exists x_1 \in Im(P) \wedge \exists x_2 \in N(P) \quad \textit{tais que} \quad x = x_1 + x_2,$$

onde P *é uma projeção,* $P : \mathbb{C}^n \to \mathbb{C}^n$*.*

Prova

Considere P uma projeção, $P : \mathbb{C}^n \to \mathbb{C}^n$. Todo elemento $x \in \mathbb{C}^n$ pode ser escrito da forma,

$$x = (I - P)(x) + P(x).$$

Pelo Teorema 3.1, itens 1 e 2, podemos concluir a prova. $\diamond$

Uma recíproca interessante é dada no teorema a seguir.

Teorema 3.3 *Para todo par de subespaços* M *e* S *de* $\mathbb{C}^n$*, tais que, a soma direta desses dois subespaços resulta em* $\mathbb{C}^n$*, temos que existe uma única projeção,* P*, onde* $Im(P) = M$ *e* $N(P) = S$*.*

Prova

Seja $x \in \mathbb{C}^n$, $x = x_1 + x_2$, onde $x_1 \in M$ e $x_2 \in S$. Vamos definir a seguinte função P.

$$P(x) = \begin{cases} x, & \text{se} \quad x \in M \\ 0, & \text{se} \quad x \in S \end{cases} \tag{3.1}$$

Para $x = 0$, temos que, $P(0) = 0$, pois P é uma transformação linear, então, seja $x \in \mathbb{C}^n$,

$$\begin{aligned} P^2(x) &= P^2(x_1 + x_2) = P^2(x_1) + P^2(x_2) \\ &= P(x_1) + P(0) = x_1 = P(x). \end{aligned}$$

Logo, $P^2(x) = P(x), \forall x \in \mathbb{C}^n$, ou seja, P é uma projeção. A partir da definição da função, temos que, $Im(P) = M$ e $N(P) = S$, o que conclui a prova da existência da função P.

Suponhamos que uma projeção Q, $Q : \mathbb{C}^n \to \mathbb{C}^n$, tal que, $Im(Q) = M$ e $N(Q) = S$ e, $Q(x) \neq P(x)$, $\forall x \in \mathbb{C}^n$, temos

$$\begin{aligned} x \in \mathbb{C}^n, \exists! x_1 \in M \wedge \exists! x_2 \in S \quad &\text{com} \quad x = x_1 + x_2 \\ Q(x) \quad &= \quad Q(x_1) + Q(x_2) = Q(x_1) = x_1 \\ &= \quad P(x_1) + P(x_2) = P(x) \\ &\Rightarrow \quad P(x) = Q(x), \forall x \in \mathbb{C}^n. \quad \text{Absurdo!} \end{aligned} \qquad \diamond$$

Corolário 3.1 *Considerando o Teorema 3.3 temos*

$$\mathbb{C}^n = N(P) \oplus Im(P).$$

Prova

Por hipótese temos, $\mathbb{C}^n = M \oplus S$, e $Im(P) = M$ e $N(P) = S$. Assim $\mathbb{C}^n = N(P) \oplus Im(P)$. $\diamond$

Corolário 3.2 *Considerando o Teorema 3.3, seja $L = S^{\perp}$. Então*

$$\begin{cases} u \in M \\ x - u \perp L \end{cases} \tag{3.2}$$

Prova

Se $dim(Im(P)) = m$ então, pelo Teorema 3.1, item 1, $dim(N(P)) = dim(Im(I - P)) = n - m$. Podemos definir S a partir de seu complemento ortogonal, $L = S^{\perp}$, onde $dim(L) = m$. Façamos $u = P(x)$, assim, pelo Teorema 3.1, ítem 1 e pelo Teorema 3.3, temos que:

$$\begin{cases} P(x) \in M \\ x - P(x) \in S \end{cases} \Longrightarrow \begin{cases} u \in M \\ x - u \perp L \end{cases} \quad \diamond$$

Essa transformação linear P é dita projeção sobre M, ao longo ou paralela ao subespaço linear S, ou, uma projeção sobre M e ortogonal ao subespaço L.

Teorema 3.4 *Dados dois subespaços $M, L \in \mathbb{C}^n$, de mesma dimensão m, as duas condições abaixo são equivalentes:*

1. *Nenhum vetor de M diferente de zero é ortogonal a L.*
2. *Para todo $x \in \mathbb{C}^n$ existe um único vetor u que satisfaz às condições dadas em (3.2).*

Prova

1) Queremos provar que as condições dadas em (3.2) são satisfeitas. Temos, por hipótese, que se $x \in M$ e $x \perp L \Rightarrow x = 0$, ou seja, $M \cap L^{\perp} = \{0\}$. Como $dim(L) = m \Rightarrow dim(L^{\perp}) = n - m$, então $\mathbb{C}^n = M \oplus L^{\perp}$. Usando o Teorema 3.3, temos que existe uma projeção P, tal que $Im(P) = M$ e $N(P) = L^{\perp}$, como pelo Teorema 3.1, ítem 1, $N(P) = Im(I - P)$, segue que :

$$\begin{cases} P(x) \in M \\ x - P(x) \in L^{\perp} \end{cases} \quad \text{fazendo,} \quad u = P(x) \quad \Rightarrow \begin{cases} u \in M \\ x - u \perp L \end{cases}$$

ou seja, $1 \Rightarrow 2$.

2) Suponhamos que $\exists u \in M$, $u \neq 0$, tal que u é ortogonal a L. Seja $x \in \mathbb{C}^n$, onde $x \in L$, então:

$$(x - u, u) = (x, u) - (u, u) = 0 - \|u\| = \|u\| \neq 0$$

assim $x - u$ não é ortogonal a L, ou seja, as condições dadas em (3.2) não são satisfeitas, absurdo! Logo $u = 0$, podemos concluir que $2 \Rightarrow 1$. $\diamond$

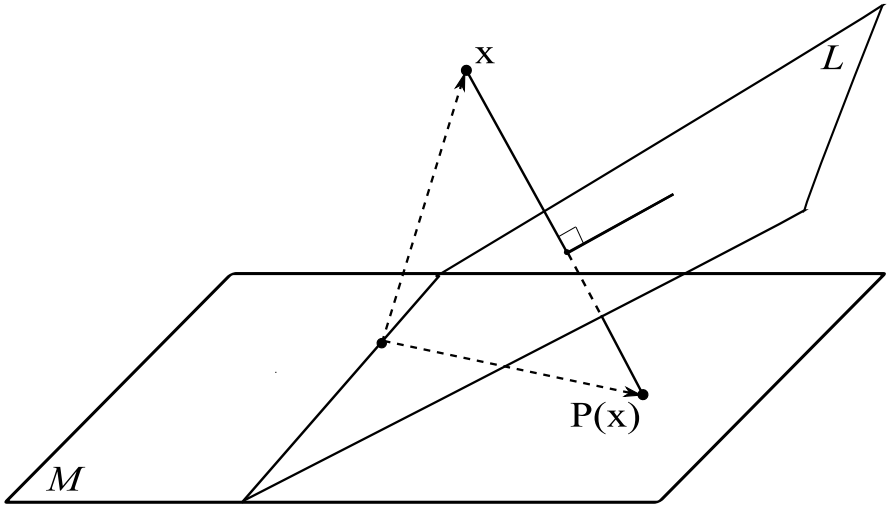


Figura 3.1: Projeção de x sobre M e ortogonal a L.

Uma interpretação geométrica desse teorema seria a de que dados $M, L \subset \mathbb{C}^n$, sob as hipóteses do Teorema 3.4, existe um único subespaço $L^\perp \subset \mathbb{C}^n$ ortogonal a M. Essa interpretação é análoga ao V postulado de Euclides.

Resumindo, dados em $\mathbb{C}^n$ os subespaços de mesma dimensão, M e L, satisfazendo à condição, $M \cap L^\perp = \{0\}$, existe uma projeção P sobre M e ortogonal a L que define o vetor projeção u para todo $x \in \mathbb{C}^n$ atendendo às condições de (3.2). Essa projeção é tal que:

$$Im(P) = M \quad \text{e} \quad N(P) = L^\perp$$

Se $P(x) = 0$, $x \in N(P)$, ou seja, $x \in L^\perp$.

3.2.1 Representação Matricial

Definição 3.4 *Seja a matriz $A \in \mathbb{C}^{n\times n}$,a matriz **adjunta**, que é a transposta conjugada de A, notada por A^H é*

$$A^H = \bar{A}^T,$$

onde barra representa a operação de conjugação.

Teorema 3.5 *Considere $M, L \in \mathbb{C}^n$, com $dim(M) = dim(L) = m$, $M \cap L^\perp = \{0\}$, V e W, bases de M e L, respectivamente. Então, $W^H V$ é não-singular.*

Prova [26].

Definição 3.5 *Dois conjuntos de vetores $A = \{a_1, \ldots, a_m\}$ e $B = \{b_1, \ldots, b_m\}$ são ditos **bi-ortogonais** se:*

$$(a_i, b_j) = \delta_{ij}, \quad onde \quad \delta_{ij} = \begin{cases} 1, se & i = j, \\ 0, se & i \neq j. \end{cases}$$

Na forma matricial temos $B^H A = I$.

Teorema 3.6 *Considere $M, L \in \mathbb{C}^n$, com $dim(M) = dim(L) = m$, $M \cap L^\perp = \{0\}$, V e W, bases de M e L, respectivamente. Então, existe uma projeção P, sobre M e com direção ortogonal a L, dada por:*

$$P = V(W^H V)^{-1} W^H. \tag{3.3}$$

Prova

Do Teorema 3.4, temos que $\mathbb{C}^n = M \oplus L^{\perp}$, pelo Teorema 3.3, podemos considerar $M = Im(P)$. Sejam, $V = [v_1, \ldots, v_m]$ e $W = [w_1, \ldots, w_m]$, bases de M e L, respectivamente. Como $P(x) \in M$, seja Vy sua representação na base V. A restrição $x - P(x) \perp L$ é equivalente à condição:

$$((x - Vy), w_j) = 0 \quad \text{para} \quad j = 1, \ldots, m.$$

Na forma matricial pode ser reescrita como:

$$W^H(x - Vy) = 0. \tag{3.4}$$

Segue que:

$$\begin{aligned} W^H(x - Vy) &= 0 \\ W^H x - W^H V y &= 0 \\ W^H V y &= W^H x \\ y &= (W^H V)^{-1} W^H x \end{aligned}$$

logo:

$$P = V(W^H V)^{-1} W^H. \qquad \diamond$$

Corolário 3.3 *Considerando as hipóteses do Teorema 3.6, e sejam V e W bi-ortogonais, então*

$$P(x) = VW^H x,$$

ou seja,

$$P = VW^H. \tag{3.5}$$

Prova

Por definição, $W^H V = I$, substituindo em (3.4), obtemos (3.5). $\diamond$

3.2.2 Projeções Ortogonais

Definição 3.6 *Seja o subespaço vetorial, $S \subset \mathbb{C}^n$. A operação ou função, **produto interno**, $(,) : S \times S \to \mathbb{C}$, está definida, se os seguintes axiomas são atendidos: Sejam $u, v, w \in S$ e $\alpha, \beta \in \mathbb{C}$*

1. $(u, v) = \overline{(v, u)}$;

2. $(u, v + w) = (u, v) + (u, w)$;

3. $(\alpha u, v) = \alpha(u, v)$ *e* $(u, \beta v) = \bar{\beta}(u, v)$;

4. $(u, u) \geq 0$, *e,* $(u, u) = 0 \Leftrightarrow u = 0$.

Definição 3.7 *Uma projeção P é dita* ***ortogonal*** *se*

$$P^H P = P P^H = I.^1 \tag{3.6}$$

Se o espaço vetorial considerado, $S \subset \mathbb{C}^n$, admite um produto interno, temos que

$$(u, v) = (Pu, Pv), \ \forall x, \forall y \in S.$$

Se a igualdade (3.6) não se verifica, P é dita ***oblíqua****.*

Teorema 3.7 *Seja P uma projeção ortogonal de $\mathbb{C}^n$, então o núcleo de P é igual ao complemento ortogonal da imagem de P, ou seja,*

$$N(P) = Im(P)^{\perp}. \tag{3.7}$$

Prova

Sejam $x \in N(P)$ e $y \in Im(P)$, segue que

$$(x, y) = (P(x), P(y)) = (0, y) = 0,$$

como $x \in N(P)$ e $\forall y \in Im(P)$, temos que $(x, y) = 0$, então

$$x \in Im(P)^{\perp} \Rightarrow N(P) \subset Im(P)^{\perp}. \tag{3.8}$$

Seja $x \in Im(P)^{\perp}$, então $(x, P(y)) = 0$, $\forall y \in S$. Mas

$$(x, P(y)) = (P(x), P((P(y))) = (P(x), P^2(y)) = (P(x), y),$$

assim $(P(x), y) = 0$, $\forall y \in S$, então

$$x \in N(P) \Rightarrow Im(P)^{\perp} \subset N(P). \tag{3.9}$$

De (3.8) e (3.9), podemos concluir a igualdade (3.7). $\diamond$

Podemos ilustrar a condição acima na Figura 1.2.

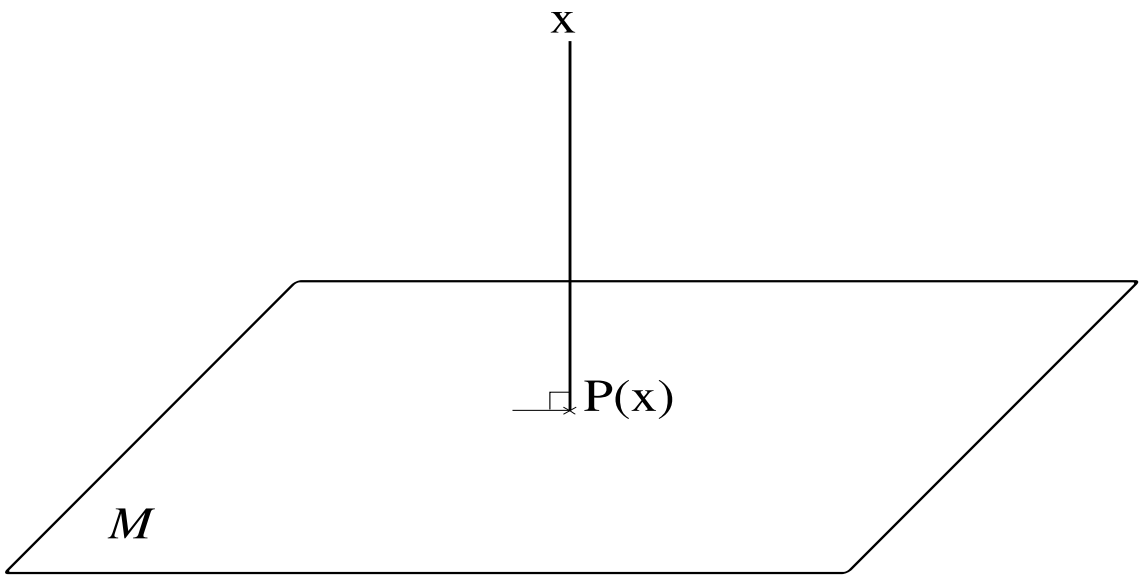


Figura 3.2: Projeção ortogonal de x sobre M.

Teorema 3.8 *Seja P uma projeção , então P^H é tal que*

$$(P^H x, y) = (x, Py), \qquad \forall x, \forall y \in \mathbb{C}^n, \tag{3.10}$$

além disso, P^H é uma projeção.

Prova

Tomando P^H e aplicando a propriedade anti-comutativa do produto interno, temos (3.10). Podemos observar que para $\forall x, \forall y \in \mathbb{C}^n$,

$$((P^H)^2 x, y) = (P^H x, Py) = (x, P^2 y) = (x, Py) = (P^H x, y).$$

Então P^H é uma projeção. $\diamond$

Corolário 3.4 *Como conseqüência do Teorema 3.8, temos:*

$$N(P^H) = Im(P)^{\perp}, \tag{3.11}$$

$$N(P) = Im(P^H)^{\perp}. \tag{3.12}$$

Prova

Vamos provar a igualdade (3.11).

[1]Matrizes, P, tais que $P^H P = PP^H$, são chamadas *matrizes normais*

Seja $x \in N(P^H)$, então, $\forall x' \in \mathbb{C}^n$, temos

$$\begin{aligned}(P^H(x), x') &= 0 \text{ de } () \Rightarrow (x, P(x')) = 0 \\ &\Rightarrow x \perp Im(P) \Rightarrow x \in Im(P)^{\perp}. \end{aligned} \tag{3.13}$$

Por outro lado, se $x \in Im(P)^{\perp}$, então, $\forall x' \in \mathbb{C}$, segue que

$$\begin{aligned}(x, P(x')) &= 0 \text{ de } () \Rightarrow (P^H x, x') = 0 \\ &\Rightarrow P^H(x) = 0 \Rightarrow x \in N(P)^H) \end{aligned} \tag{3.14}$$

De, (3.13) e (3.14), concluímos (3.11); analogamente, provaríamos (3.12). $\diamond$

Teorema 3.9 *Uma projeção é ortogonal se, e somente se, é Hermitiana.*

Prova [26].

Teorema 3.10 *Seja P uma projeção ortogonal sobre o subespaço vetorial $M \subset \mathbb{C}^n$, $M = Im(P)$. Considere uma matriz V, de ordem $n \times m$, cujas colunas formam uma base ortonormal de M. Então, podemos representar P pela matriz $P = VV^H$, além disso,*

$$VV^H x \in M \qquad e \qquad (I - VV^H)x \in M^{\perp}. \tag{3.15}$$

Prova

Esse é um caso particular da representação matricial de uma projeção dada por (3.5), fazendo $M = L$ e, tomando V segundo a hipótese, segue daí, também, (3.15). $\diamond$

Teorema 3.11 *A representação de uma projeção ortogonal, P, não é única, e, para quaisquer duas bases ortonormais, $V_1, V_2 \in M$, com $Im(P) = M$, temos que $V_1 V_1^H = V_2 V_2^H$.*

Prova

A representação de P não será única, pois M tem mais de uma base ortonormal, basta então aplicar o Teorema 3.10 às bases consideradas.

Sejam V_1, V_2 bases ortonormais de M, então $\exists C \in \mathbb{C}^{m \times m}$ tal que $V_1 = V_2 C$, com $CC^H = C^H C = I$ [27], assim

$$V_1 V_1^H = V_2 C (V_2 C)^H = V_2 C C^H V_2^H = V_2 V_2^H. \qquad \diamond$$

Teorema 3.12 *Seja P uma projeção ortogonal, então*

$$\|x\|_2^2 = \|Px\|_2^2 + \|(I-P)x\|_2^2. \tag{3.16}$$

Prova

Seja $x \in \mathbb{C}^n$, temos que $x = Px + (I-P)x$, mas $Px \perp (I-P)x$, assim (3.16) se verifica. $\diamond$

Corolário 3.5 *Uma consequência do Teorema 3.12 é que, $\forall x \in \mathbb{C}^n$,*

$$\|Px\|_2^2 \leq \|x\|_2^2. \tag{3.17}$$

Prova

Segue de (3.16). $\diamond$

Corolário 3.6 *Segue do Corolário 3.5 que*

$$\frac{\|Px\|_2}{\|x\|_2} \leq 1, \qquad \forall x \in \mathbb{C}^n.$$

Temos ainda que, se tomarmos $x \in Im(P)$,

$$P(x) = x \quad \Rightarrow \quad \|P(x)\|_2 = \|x\|_2 \quad \Rightarrow \quad \frac{\|P(x)\|_2}{\|x\|_2} = 1,$$

logo o $\sup_{\|x\|_2 \neq 0} \left\{ \frac{\|P(x)\|_2}{\|x\|_2} \right\} = 1$ é atingido. Portanto:

$$\|P\|_2 = 1,$$

para toda projeção ortogonal P.

Prova

Segue de (3.17). $\diamond$

Uma projeção ortogonal tem apenas dois autovalores distintos: zero, com multiplicidade igual à $dim(N(P))$, e um, com multiplicidade igual à $dim(Im(P))$. Qualquer vetor da imagem de P é um autovetor associado ao autovalor um. Qualquer vetor do núcleo de P é um autovetor associado ao autovalor zero.

No próximo teorema, uma importante condição de otimalidade é estabelecida.

Teorema 3.13 *Seja P uma projeção ortogonal sobre um subespaço $M \subset \mathbb{C}^n$. Então, para todo $x \in \mathbb{C}^n$, é verdade que:*

$$\min_{y \in M} \|x - y\|_2 = \|x - Px\|_2. \tag{3.18}$$

Prova [26].

Seja $\hat{y} = Px$ para a projeção ortogonal P sobre o subespaço M, temos:

Corolário 3.7 *Sejam M um subespaço de $\mathbb{C}^n$ e $x \in \mathbb{C}^n$. Então:*

$$\min_{y \in M} \|x - y\|_2 = \|x - \hat{y}\|_2$$

se satisfaz às duas condições abaixo:

$$\begin{cases} \hat{y} \in M, \\ x - \hat{y} \perp M. \end{cases}$$

Prova [26].

Capítulo 4

Método de Projeção e Subespaço de Krylov

4.1 Introdução

Neste capítulo vamos abordar o algoritmo de uma projeção oblíqua e alguns teoremas relacionados. Uma de nossas contribuições é apresentar o operador de projeção e sua representação matricial.

Em relação ao outro assunto tratado, Subespaço de Krylov, apresentaremos algumas definições e teoremas. Optamos por fazer as demonstrações de alguns deles explicitamente, como contribuição, pois na literatura consultada, ou não aparecem demonstrados, ou não são bem explicados.

Denominamos de *teorema* os resultados clássicos obtidos da literatura e *proposição*, as que foram propostas e demonstradas e que não se encontram na bibliografia pesquisada.

4.2 Método de Projeção

Seja A uma matriz de ordem n e b um vetor de $\mathbb{C}^n$, considere, então, o sistema linear

$$Ax = b. \tag{4.1}$$

Uma das formas de se obter uma aproximação para a solução desse sistema é usar uma técnica de projeção.

Sejam $\mathcal{K}$ e $\mathcal{L}$ dois subespaços de $\mathbb{C}^n$ de dimensão m. A partir desses subespaços, vamos obter um operador de projeção, P, de tal forma que $\mathbb{C}^n = N(P) \oplus Im(P)$. Vamos utillizar a projeção oblíqua sobre $\mathcal{K}$, com direção ortogonal a $\mathcal{L}$. Considerando $\tilde{x}$ uma solução aproximada do sistema e o resíduo dado por $r = b - A\tilde{x}$, teremos como resultado uma aproximação da solução do sistema dado, que atende a

$$\text{Determine} \quad \tilde{x} \in \mathcal{K} \quad \text{tal} \quad \text{que} \quad b - A\tilde{x} \perp \mathcal{L}.$$

Se desejarmos utilizar uma solução inicial, x_0, então a solução aproximada deve ser procurada no espaço afim $x_0 + \mathcal{K}$, ao invés de no espaço vetorial $\mathcal{K}$. O problema pode ser redefinido por:

$$\text{Determine} \quad \tilde{x} \in x_0 + \mathcal{K} \quad \text{tal} \quad \text{que} \quad b - A\tilde{x} \perp \mathcal{L}.$$

Podemos escrever $\tilde{x}$ na forma $\tilde{x} = x_0 + \delta$ com $\delta \in \mathcal{K}$. O vetor resíduo inicial r_0 é definido como $r_0 = b - Ax_0$ e o erro e é dado por $e = x^* - \tilde{x}$, onde x^* é a solução exata do sistema (4.1). Sustituindo $\tilde{x}$ e r_0, segue que

$$b - A(x_0 + \delta) \perp \mathcal{L} \quad \text{ou} \quad r_0 - A\delta \perp \mathcal{L}.$$

Em outras palavras, a solução aproximada pode ser definida como:

$$\tilde{x} = x_0 + \delta, \quad \delta \in \mathcal{K}; \tag{4.2}$$

$$(r_0 - A\delta, w) = 0, \quad \forall w \in \mathcal{L}. \tag{4.3}$$

Esse é o passo básico da projeção, na sua forma mais geral.

Vamos construir o algoritmo da projeção, tendo uma solução inicial x_0,

$$\begin{aligned}
\tilde{x} &= x_0 + Vy; \\
Vy &= \tilde{x} - x_0; \\
AVy &= A(\tilde{x} - x_0); \\
AVy &= b - Ax_0 - (b - A\tilde{x}) = r_0 - \tilde{r}; \\
W^H AVy &= W^H r_0 - W^H \tilde{r}; \\
W^H AVy &= W^H r_0; \\
y &= (W^H AV)^{-1} W^H r_0.
\end{aligned}$$

Onde $W^H AV$ precisa ser invertível, para que esta operação faça sentido.

Um protótipo da técnica de projeção é representado no Algoritmo 2.1.

Algoritmo 4.1. Protótipo do Método de Projeção

1. Sejam a matriz A e os vetores x_0 e b, **Do**
2. Selecione um par de subespaços $\mathcal{K}$ e $\mathcal{L}$
3. Escolha bases $V = [v_1, \ldots, v_m]$ e $W = [w_1, \ldots, w_m]$ para $\mathcal{K}$ e $\mathcal{L}$
4. $r := b - Ax$
5. $y := (W^H AV)^{-1} W^H r$
6. $x := x + Vy$
7. **EndDo**

O teorema, a seguir, mostra dois casos particulares onde sabemos, a priori, que a matriz $W^H AV$ é invertível.

Teorema 4.1 *Sejam $A \in \mathbb{C}^{n\times n}$, $\mathcal{K}$ e $\mathcal{L}$ subespaços de $\mathbb{C}^n$, satisfazendo uma das condições abaixo:*
(i) A é positiva definida e $\mathcal{K} = \mathcal{L}$ ou
(ii) A é não-singular e $\mathcal{L} = A\mathcal{K}$.
Então a matriz $W^H AV$ é não-singular para quaisquer bases V e W, de $\mathcal{K}$ e $\mathcal{L}$, respectivamente.

Prova [26].

Teorema 4.2 *Suponha que A é uma matriz simétrica positiva definida, $\mathcal{K}, \mathcal{L} \subset \mathbb{C}^n$ e $\mathcal{K}$=$\mathcal{L}$. Então um vetor $\tilde{x}$ resulta de um método de projeção ortogonal sobre $\mathcal{K}$, com vetor inicial x_0, se minimiza a A-norma do erro sobre $x_0 + \mathcal{K}$, isto é, se*

$$E(\tilde{x}) = \min_{x \in x_0 + \mathcal{K}} E(x),$$

onde

$$E(x) \equiv (A(x^* - x), x^* - x)^{1/2},$$

x^* é solução exata.

Prova [26].

Teorema 4.3 *Seja A uma matriz quadrada qualquer, $\mathcal{K}, \mathcal{L} \subset \mathbb{C}^n$ e suponha que $\mathcal{L}$=$A\mathcal{K}$. Então um vetor $\tilde{x}$ é resultado de um método de projeção (oblíqua) sobre $\mathcal{K}$ com direção ortogonal a $\mathcal{L}$ e, partindo do vetor inicial x_0, se minimiza a norma*

Euclidiana (2-norma) do resíduo $b - Ax$ sobre $x_0 + \mathcal{K}$, isto é, se

$$R(\tilde{x}) = \min_{x \in x_0 + \mathcal{K}} R(x),$$

onde $R(x) \equiv \|b - Ax\|_2$.

Prova [26].

Proposição 4.1 *Seja o Algoritmo 2.1. Então, existe um operador de projeção oblíqua, P, que projeta o erro $e = x^* - x_0$, onde x^* é a solução exata, sobre o subespaço $\mathcal{K}$, de tal forma que o resíduo r é ortogonal a $\mathcal{L}$. O subespaço $\mathcal{K}$ é o subespaço de busca da solução do sistema, a projeção tem direção orotogonal ao subespaço $\mathcal{L}$. A matriz que representa tal operador é dada por:*

$$P = V(W^H AV)^{-1} W^H A.$$

onde V e W são bases dos subespaços $\mathcal{K}$ e $\mathcal{L}$, respectivamente.

Prova

O passo 6 do algoritmo é semelhante à equação (4.2), tomemos $\delta = Vy$.
No passo 7 temos $y = (W^H AV)^{-1} W^H r$, sabendo que $r = Ae$, onde e é o erro, temos que:

$$y = (W^H AV)^{-1} W^H Ae \quad \text{e} \quad \delta = V(W^H AV)^{-1} W^H Ae. \tag{4.4}$$

Considere $P = V(W^H AV)^{-1} W^H A$, podemos verificar que $P^2 = P$, ou seja, P é a projeção dada na Proposição 4.1.
Da equação (4.4) segue que $\delta = Pe$, por hipótese, temos que $\delta \in \mathcal{K}$, então, podemos concluir que δ é a projeção do erro e sobre o subespaço $\mathcal{K}$. $\diamond$

4.3 Subespaço de Krylov

O interesse do tema Subespaço de Krylov reside no fato de ser o subespaço de busca das aproximações da solução de um sistema linear (equação (4.1)) para os métodos iterativos que serão apresentados posteriormente. Demonstraremos de forma detalhada não só as razões que credenciam o subespaço de Krylov como um bom espaço de busca, mas também os teoremas que mostram como se obter uma base para o subespaço de Krylov.

Definição 4.1 *Considere o conjunto de vetores* $V = \{v_1, \ldots, v_m\} \subset \mathbb{C}^n$ *o* ***subespaço vetorial gerado*** *por* V, $S \subset \mathbb{C}^n$, *notado por* $S = [v_1, \ldots, v_m]$, *ou então,* $S = span(V)$.

Definição 4.2 *Um* ***subespaço de Krylov*** *é gerado por uma matriz* A, *de ordem* n, *e por um vetor* $v \in \mathbb{C}^n$, *da seguinte forma:*

$$[v, Av, \ldots, A^{m-1}v].$$

Notamos tal subespaço por $K_m(A, v)$ *[26], [24].*

Definição 4.3 *Seja* A *uma matriz de ordem* n, *o determinante*

$$p(t) = det(A - tI)$$

é chamado ***polinômio característico*** *de* A, *que pode ser escrito como*

$$p(t) = \prod_{j=1}^{d} (t - \lambda_j)^{n_j},\ n_j \leq n \ e \ \sum_{j=1}^{d} n_j = n,$$

onde $\lambda_1, \ldots, \lambda_d$ *são autovalores distintos de* A.

Definição 4.4 *Seja* A *uma matriz de ordem* n. *O* ***polinômio mínimo de um vetor v*** *é o polinômio mônico não-nulo,* p, *de menor grau, tal que,* $p(A)v = 0$.

Definição 4.5 *Seja* A *uma matriz de ordem* n, *o grau do polinômio mínimo de* v *com relação a* A, *é chamado* ***grau de v com relação a*** A.

Teorema 4.4 *O subespaço de Krylov* $\mathcal{K}_m$ *é de dimensão* m, *se o grau* μ *de* v *com relação a* A *é maior ou igual a* m, *ou seja,*

$$dim(\mathcal{K}_m) = m \quad \leftrightarrow \quad grau(v) \geq m.$$

Portanto,

$$dim(\mathcal{K}_m) = \min\{m, grau(v)\}.$$

Prova [26].

Teorema 4.5 *Seja* Q_m *uma projeção sobre* $\mathcal{K}_m$ *e seja* A_m *a restrição de* A *em* $\mathcal{K}_m$, *isto é,* $A|_{\mathcal{K}_m} = A_m = Q_m A$. *Então, para todo polinônio* q *de grau menor ou igual a* $m - 1$, *temos que*

$$q(A)v = q(A_m)v,$$

e, para todo poliônio de grau menor que m,

$$Q_m q(A)v = q(A_m)v.$$

Prova [26].

Procuramos, então, uma forma de se obter uma base para o subespaço de Krylov, $\mathcal{K}_m = K_m(A, v)$.

Teorema 4.6 *Sejam os vetores da base do subespaço de Krylov, $\mathcal{K}_m = K_m(A, v)$, dados por:*

$$u_i = A^{i-1}v$$

Definimos a matriz U_i, de ordem $n \times i$, aquela cujas colunas são os vetores u_i, $i = 1, \ldots, m$. Sejam e_i o i-ésimo vetor da base canônica de $\mathbb{C}^i$, B_i uma matriz de ordem i, tal que

$$B = (b_{ij}) = \begin{cases} 0 & se \quad i \neq j+1 \\ 1 & se \quad i = j+1 \end{cases} \tag{4.5}$$

Então,

$$AU_i = U_i B_i + u_{i+1} e_i^H \tag{4.6}$$

Antes da prova deste teorema, vamos analisá-lo. Reescrevendo a igualdade desejada matricialmente, segue que

$$\begin{bmatrix} | & & | \\ a_1 & \ldots & a_n \\ | & & | \end{bmatrix}_{n\times n} \begin{bmatrix} | & & | \\ u_1 & \ldots & u_i \\ | & & | \end{bmatrix}_{n\times i} = \begin{bmatrix} | & & | \\ u_1 & \ldots & u_i \\ | & & | \end{bmatrix}_{n\times i} \begin{bmatrix} | & & | \\ b_1 & \ldots & b_i \\ | & & | \end{bmatrix}_{i\times i}$$
$$+ \begin{bmatrix} | \\ u_{i+1} \\ | \end{bmatrix}_{n\times 1} \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix}_{1\times i}$$

Suponhamos que

$$\tilde{A} = AU_i = (\tilde{a}_{kj}) \quad , \quad \tilde{U} = U_i B_i = (\tilde{u}_{kj}).$$

Temos, então,

$$\tilde{a}_{kj} = \sum_{p=1}^{n} a_{pk} u_{pj} \quad \text{e} \quad \tilde{u}_{kj} = u_{j+1,k}.$$

Tal fato pode ser interpretado da seguinte forma. O produto U_iB_i faz com que a última coluna dessa matriz seja nula e cada coluna $\tilde{u}_i = u_{i+1}$, ou seja, o produto elimina a primeira coluna de U_i e substitui pela segunda coluna, e assim sucessivamente, até que a i-ésima coluna seja uma coluna nula.

Agora vamos ver o produto $u_{i+1}e_i^H$,

$$u_{i+1}e_i^H = \begin{bmatrix} | & | & & | \\ 0 & 0 & \dots & u_{i+1} \\ | & | & & | \end{bmatrix}_{n\times i}$$

Considere $\tilde{\tilde{U}} = U_iB_i + u_{i+1}e_i^H$, daí segue que, para $i = 1, \dots, j-1$, temos:

$$\tilde{\tilde{u}}_{kj} = u_{j+1,k} = \tilde{a}_{kj},$$

visto que $u_{j+1} = Au_j$, então,

$$AU_i = U_iB_i + u_{i+1}e_i^H.$$

Agora vamos à demonstração formal usando indução matemática.

Prova

1) Suponhamos $i = 1$,

$$A_{n\times n}U_1 = Au_1 = u_2, \text{ pois } u_1 = v,$$

$$U_1B_1 = \begin{bmatrix} u_{11} \\ u_{12} \\ \vdots \\ u_{1n} \end{bmatrix} [0] = 0$$

$$u_2e_1^H = u_2$$

Assim $A_{n\times n}U_1 = U_1B_1 + u_2e_1^H$.

2)Queremos provar que

$$AU_k = U_kB_k + u_{k+1}e_k^H$$

implica

$$AU_{k+1} = U_{k+1}B_{k+1} + u_{k+2}e_{k+1}^H.$$

Podemos escrever

$$AU_{k+1} = A \begin{bmatrix} & | \\ U_k & u_{k+1} \\ & | \end{bmatrix}.$$

Da hipótese da indução e da definição de u_k, temos

$$AU_k = \begin{bmatrix} | & | \\ U_kB_k + u_{k+1}e_k^H & u_{k+2} \\ | & | \end{bmatrix} = \begin{bmatrix} | & | \\ U_{k+1}B_{k+1} & u_{k+2} \\ | & | \end{bmatrix}$$

e segue que

$$AU_{k+1} = U_{k+1}B_{k+1} + u_{k+2}e_{k+1}^H. \qquad \diamond$$

Definição 4.6 *Considere a matriz $H = (h_{ij})$, H é uma matriz **Hessenberg superior** se $h_{ij} = 0$ para todo para i, j, tal que $i > j+1$. Matrizes **Hessenberg inferior** podem ser definidas de forma análoga.*

Teorema 4.7 *Considerando as matrizes B_i dada por (4.5) e R_i , uma matriz triangular superior, tal que, $U_i = Q_iR_i$, com $Q_i^HQ_i = I$, Q_i unitária. Temos que a matriz*

$$H_i = R_iB_iR_i^{-1}$$

é uma matriz Hessenberg superior.

Prova

Façamos uma indução matemática sobre o índice i.

1) Base da indução:

Suponha que $i = 1$.

$$R_1B_1R_1^{-1} = [r_1][0][\frac{1}{r_1}] = [0] = H_1$$

A base é verdadeira para $i = 1$.

2) Vamos provar que

$$H_k = R_kB_kR_k^{-1}$$

onde H_k é uma Hessenberg superior, implica que H_{k+1} é Hessenberg superior.

$$\begin{aligned}
H_{k+1} &= R_{k+1}B_{k+1}R_{k+1}^{-1} \\
&= \begin{bmatrix} R_k & r_{k+1} \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} B_k & 0 \\ e_k^H & 0 \end{bmatrix} \begin{bmatrix} R_k^{-1} & -\frac{1}{\alpha}R_k^{-1}r_{k+1} \\ 0 & \frac{1}{\alpha} \end{bmatrix} \\
&= \begin{bmatrix} R_kB_k + r_{k+1}e_k^H & 0 \\ \alpha e_k^H & 0 \end{bmatrix} \begin{bmatrix} R_k^{-1} & -\frac{1}{\alpha}R_k^{-1}r_{k+1} \\ 0 & \frac{1}{\alpha} \end{bmatrix} \\
&= \begin{bmatrix} R_kB_kR_k^{-1} + r_{k+1}e_k^HR_k^{-1} & -\frac{1}{\alpha}(R_kB_kR_k^{-1} + r_{k+1}e_k^HR_k^{-1}) \\ \alpha e_k^HR_k^{-1} & -e_k^HR_k^{-1}r_{k+1} \end{bmatrix}.
\end{aligned}$$

Vamos calcular cada elemento da matriz H_{k+1}, dada acima. Façamos

$$H_{k+1} = \begin{bmatrix} H_{k+1,11} & H_{k+1,12} \\ H_{k+1,21} & H_{k+1,22} \end{bmatrix}.$$

Segue que $H_{k+1,11} = R_kB_kR_k^{-1} + r_{k+1}e_k^{-H}R_k^{-1}$. Observe que:

$$r_{k+1}e_k^H = \begin{bmatrix} r_{k+1+,1} \\ r_{k+1,2} \\ \vdots \\ r_{k+1,k} \end{bmatrix}_{k\times 1} \begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}_{1\times k} = \begin{bmatrix} | & | & r_{k+1,1} \\ 0 & 0 & \vdots \\ | & | & r_{k+1,k} \end{bmatrix}_{k\times k}$$

$$\begin{aligned}
r_{k+1}e_k^HR_k^{-1} &= \begin{bmatrix} | & r_{k+1,1} \\ 0 & \vdots \\ | & r_{k+1,k} \end{bmatrix} \begin{bmatrix} r_{11}^{-1} & x & \dots & x \\ 0 & r_{22}^{-1} & \dots & x \\ \vdots & \vdots & & x \\ 0 & 0 & \dots & r_{kk}^{-1} \end{bmatrix} \\
&= \begin{bmatrix} | & | & r_{kk}^{-1}r_{k+1,1} \\ 0 & 0 & \vdots \\ | & | & r_{kk}^{-1}r_{k+1,k} \end{bmatrix}_{k\times k} = Col
\end{aligned}$$

Assim, temos que $H_{k+1,1} = \tilde{H}_k + Col$, onde $\tilde{H}_k = R_kB_kR_k^{-1}$ é Hessenberg superior, pela hipótese da indução. Então $H_{k+1,1}$ é uma matriz de ordem k, Hessenberg superior .

$$H_{k+1,21} = \alpha e_k^HR_k^{-1} = \begin{bmatrix} 0 & 0 & \dots & \alpha \end{bmatrix} R_k^{-1} = \begin{bmatrix} 0 & 0 & \dots & \alpha r_{kk}^{-1} \end{bmatrix}_{1\times k}$$

Temos, ainda que

$$H_{k+1,12} = (-\frac{1}{\alpha})H_{k+1,11}r_{k+1}$$

Uma Hessenberg que multiplica um vetor, é um vetor, então $H_{k+1,12}$ é uma matriz de ordem $1 \times k$.

Finalmente,

$$H_{k+1,22} = -e_k^H R_k^{-1} r_{k+1} = r_{kk}^{-1} r_{k+1,k}.$$

Assim, os elemento da matriz H_{k+1}, são tais que

$$H_{k+1} = (h_{ij}) = \begin{cases} h_{ij} \in H_{k+1,11}, & i,j = 1,\ldots,k \\ h_{ij} \in H_{k+1,21}, & i = k+1, \ ,j = 1,\ldots,k \\ h_{ij} \in H_{k+1,12}, & i = 1,\ldots,k, \ j = k+1 \\ h_{ij} \in H_{k+1,22}, & i,j = k+1 \end{cases}$$

$\Rightarrow H_{k+1}$ é uma Hessenberg superior. $\diamond$

Teorema 4.8 *Seja o subespaço de Krylov, $\mathcal{K}_m = K_m(A,v)$, então, $\exists Q_m \in \mathbb{C}^{m\times m}$, tal que*

$$Q_m^H A Q_m = H_m,$$

onde $Q_m^H Q_m = I$ e H_m é uma matriz Hessenberg superior.

Prova

Seja a base de $\mathcal{K}_m$, dada por

$$u_i = A^{i-1}v, \ i = 1,\ldots,m.$$

Definimos a matriz U_i, de ordem $n \times i$, aquela cujas colunas são os vetores u_i, $i = 1,\ldots,m$.

Pelo Teorema 4.6, temos

$$AU_i = U_i B_i + u_{i+1}e_i^H.$$

Podemos fazer a decomposição QR de U_i, pois as colunas de U_i são linearmente independentes, seja

$$U_i = Q_i R_i$$

com $Q_i^H Q_i = I$ e R_i uma matriz triangular superior. Assim,

$$\begin{aligned} AQ_iR_i &= Q_iR_iB_i + u_{i+1}e_i^H \\ AQ_i &= Q_iR_iB_iR_i^{-1} + u_{i+1}e_i^HR_i^{-1} \end{aligned} \tag{4.7}$$

Aplicando o Teorema 4.7 à equação (4.7), obtemos:

$$\begin{aligned} AQ_i &= Q_i\tilde{H}_i + u_{i+1}e_i^HR_i^{-1} \\ &= Q_i\tilde{H}_i + \frac{1}{r_{ii}}u_{i+1}e_i^H, \end{aligned} \tag{4.8}$$

onde $\tilde{H}_i = R_iB_iR_i^{-1}$ é Hessenberg superior.

A matriz U_{i+1} também apresenta fatoração QR, dada por

$$\begin{aligned} U_{i+1} &= Q_{i+1}R_{i+1} \\ &= \begin{bmatrix} Q_i & q_i \\ v_i^H & q_{i+1,i+1} \end{bmatrix} \begin{bmatrix} R_i & \tilde{r} \\ 0 & r_{i+1,r+1} \end{bmatrix} \end{aligned}$$

Como Q_{i+1} é tal que $Q_{i+1}^H Q_{i+1} = I$,

$$\begin{bmatrix} Q_i^H & v_i \\ q_i^H & q_{i+1,i+1} \end{bmatrix} \begin{bmatrix} Q_i & q_i \\ v_i^H & q_{i+1,i+1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

segue o sistema:

$$\begin{cases} Q_i^HQ_i + v_iv_i^H = I \Longrightarrow v_iv_i^H = 0 \Longrightarrow v_i = 0 \\ Q_i^Hq_i = -q_{i+1,i+1}v_i \Longrightarrow v_i = -\frac{1}{q_{i+1,i+1}}Q_i^Hq_i = -\frac{1}{q_{i+1,i+1}}q_i^HQ_i = v_i^H \\ q_i^Hq_i = -q_{i+1,i+1}v_i^H \\ q_i^Hq_i + q_{i+1,i+1}^2 = 1 \end{cases}$$

Assim,

$$Q_{i+1} = \begin{bmatrix} Q_i & q_i \\ 0 & q_{i+1,i+1} \end{bmatrix}$$

$$U_{i+1} = \begin{bmatrix} Q_i & q_i \\ 0 & q_{i+1,i+1} \end{bmatrix} \begin{bmatrix} R_i & \tilde{r} \\ 0 & r_{i+1,r+1} \end{bmatrix}$$

$$\begin{aligned} u_{i+1} &= Q_i\tilde{r} + r_{i+1,i+1}q_i + r_{i+1,i+1}q_{i+1,i+1} \\ &= Q_i\tilde{r} + r_{i+1,i+1}q_{i+1} \end{aligned} \tag{4.9}$$

Substituindo (4.9) em (4.8), segue que

$$\begin{aligned} AQ_i &= Q_i(\tilde{H}_i + \frac{1}{r_{ii}}\tilde{r}e_i^H) + \frac{r_{i+1,i+1}}{r_{i,i}}q_{i+1}e_i^H \\ &= Q_iH_i + \alpha q_{i+1}e_i^H. \end{aligned} \tag{4.10}$$

Então,

$$Q_i^H AQ_i = Q_i^H Q_i H_i + Q_i^H q_{i+1}e_i^H = H_i + Q_i^H q_{i+1}e_i^H.$$

Observe que $Q_i^H q_{i+1}e_i^H = 0$ pois as colunas de Q_i são ortogonais a q_{i+1}, por construção. Então, $Q_i^H AQ_i = H_i$, onde H_i é uma matriz Hessenberg superior. $\diamond$

Observamos que se fizermos

$$\begin{aligned} q_{i+1}^H AQ_i &= q_{i+1}^H Q_i H_i + \alpha q_{i+1}^H q_{i+1}e_i^H \\ q_{i+1}^H AQ_i &= \alpha e_i^H \\ q_{i+1}^H AQ_iQ_i^H &= \alpha e_i^H Q_i^H \\ q_{i+1}Aq_i &= \alpha e_i^H Q_i^H q_i \Longrightarrow q_{i+1}Aq_i = \alpha. \end{aligned}$$

Como $Q_{i+1}^H AQ_{i+1} = H_{i+1}$, temos que $\alpha = h_{i+1,i}$.

Em vários métodos para resolução de sistemas lineares, o subespaço de Krylov é utilizado como espaço de busca para a melhor solução aproximada de $Ax = b$. Vejamos por que tal subespaço é conveniente.
Dado um sistema linear, $Ax = b$, e x^* sua solução exata. Vamos considerar A não-singular.

Teorema 4.9 *Seja A uma matriz não-singular de ordem n, então*

$$A^{-1} = -\frac{1}{\alpha_0}\sum_{j=0}^{m-1}\alpha_{j+1}A^j,$$

onde α_j são os coeficientes do polinômio mínimo associado a A.

Prova

Sejam $\lambda_1, \ldots, \lambda_d$ os autovalores distintos de A, o polinômio característico de A será dado por

$$p(t) = \sum_{j=1}^{d}(t - \lambda_j)^{n_j},\ n_j \leq n.$$

Seja q o polinômio mínimo associado a A,

$$q(t) = \sum_{j=1}^{d} (t - \lambda_j)^{m_j}, \quad \text{onde} \quad 1 \leq m_j \leq n_j.$$

Podemos escrever

$$q(A) = \sum_{j=0}^{m} \alpha_j A^j, \text{ onde } m = \sum_{j=1}^{d} m_j.$$

Como A é não-singular, temos que

$$\alpha_0 = \prod_{j=1}^{d} \lambda_j \neq 0.$$

Sabemos que $q(A) = 0$ então

$$\begin{aligned} \alpha_0 I + \alpha_1 A + \ldots + \alpha_m A^m &= 0 \\ A^{-1} &= -\frac{1}{\alpha_0} \sum_{j=0}^{m-1} \alpha_{j+1} A^j. \qquad \diamond \end{aligned}$$

Teorema 4.10 *Considere a solução exata, x^*, de um sistema linear da forma $Ax = b$, onde A é a matriz não-singular de ordem n, é tal que*

$$x^* \in x_0 + K_m(A, r_0),$$

onde x_0 é a aproximação da solução inicial e m o grau do polinômio mínimo de A.

Prova

Suponhamos que $x_0 = 0$, a solução exata do sistema é dada por $x^* = A^{-1}b$, assim

$$\begin{aligned} x^* &= A^{-1}b \\ &= \big(-\frac{1}{\alpha_0} \sum_{j=0}^{m-1} \alpha_{j+1} A^j \big) b \\ &= -\frac{1}{\alpha_0} \sum_{j=0}^{m-1} \alpha_{j+1} A^j b. \end{aligned}$$

Então x^* é escrito como combinação linear de $(b, Ab, \ldots, A^{m-1}b)$, ou seja, $x^* \in K_m(A, b)$.

Se $x_0 = 0$ podemos ver, facilmente, que $K_m(A, b) = K_m(A, r_0)$ e podemos usar

$K_m(A, r_0)$ como espaço de busca para a solução aproximada.

Suponhamos que $x_0 \neq 0$, então

$$\begin{aligned} x^* &= A^{-1}b \\ &= A^{-1}(Ax_0 + r_0) \\ &= x_0 + A^{-1}r_0, \end{aligned}$$

logo $x^* \in x_0 + K_m(A, r_0)$. $\diamond$

Capítulo 5

Método da Bi-Ortogonalização de Lanczos

5.1 Introdução

Neste capítulo abordaremos o método da Bi-Ortogonalização de Lanczos [13], que é parte do método das "iterações minimizadas". Vamos provar que o algoritmo apresentado gera bases bi-ortogonais para dois subespaços de Krylov convenientes. Sob o enfoque escolhido, será apresentada uma proposição que demonstra a existência de um operador de projeção que atua neste método, sua formulação explícita será exibida.

O método desenvolvido por Lanczos [13] tem como objetivo encontrar soluções para problemas de autovalores de operadores lineares, principalmente os relacionados com a teoria das vibrações. Lanczos observou que, a partir de duas séries, a saber, expansão de Liouville-Neumann ou série de Schmidt, as equações de operadores diferenciais ou integrais podem ser resolvidas.

Ele construiu o método das "iterações minimizadas"(*minimized iterations*) para determinar autovalores e autovetores de uma dada matriz. Lanczos desenvolveu seu algoritmo para matrizes simétricas e não-simétricas e apresentou algumas aplicações: "Vibrações laterais de uma barra", "O problema de autovalores do operador linear integral", "O problema de autovalores do operador linear diferencial", "Equações diferenciais de segunda ordem; Método de Milne".

Em outro artigo [28], Lanczos descreve um algoritmo para resolver sistemas lineares,

com um grande número de variáveis, usando uma sucessão de aproximações.

5.2 Método da Bi-Ortogonalização de Lanczos

Considere a matriz A de ordem n. A proposta do método é gerar duas bases, V e W, bi-ortogonais, para os seguintes subespaços de Krylov,

$$\mathcal{K} = K_m(A, v_1) \quad \text{e} \quad \mathcal{L} = K_m(A^H, w_1),$$

respectivamente.

Sejam

$$V = \{v_1, v_2, \ldots, v_m\} \quad \text{e} \quad W = \{w_1, w_2, \ldots, w_m\}.$$

Mostraremos o algoritmo que gera as bases V e W, de modo que estas sejam bi-ortogonais. Vejamos o algoritmo da Bi-Ortogonalização de Lanczos.

Algoritmo 5.1. Algoritmo da Bi-Ortogonalização de Lanczos

1. Escolha dois vetores v_1, w_1 tais que $(v_1, w_1) = 1$
2. Sejam $\beta_1 \equiv \delta_1 \equiv 0$, $w_0 \equiv v_0 \equiv 0$
3. Para $j = 1, 2, \ldots, m$. **Do**
4. $\quad \alpha_j = (Av_j, w_j)$
5. $\quad \tilde{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
6. $\quad \tilde{w}_{j+1} = A^H w_j - \alpha_j w_j - \delta_j w_{j-1}$
7. $\quad \delta_{j+1} = |(\tilde{v}_{j+1}, \tilde{w}_{j+1})|^{1/2}$. If $\delta_{j+1} = 0$ *Stop*
8. $\quad \beta_{j+1} = \overline{(\tilde{v}_{j+1}, \tilde{w}_{j+1})}/\delta_{j+1}$
9. $\quad w_{j+1} = \tilde{w}_{j+1}/\beta_{j+1}$
10. $\quad v_{j+1} = \tilde{v}_{j+1}/\delta_{j+1}$
11. **EndDo**

Esse algoritmo gera duas bases, V e W, bi-ortogonais.

Teorema 5.1 *Dado o Algoritmo 5.1, os conjuntos gerados por este algoritmo, a cada iteração, dados por:*

$$\begin{aligned} V_i &= \{v_1, \ldots, v_i\}, \\ W_i &= \{w_1, \ldots, w_i\} \end{aligned}$$

são bases de $\mathcal{K}_i = K_i(A, v_1)$ *e de* $\mathcal{L}_i = K_i(A^H, w_1)$*, respectivamente.*

Prova

Vamos provar que tais conjuntos são bases para os subespaços considerados. Para tal, usaremos indução matemática.

Provaremos que $V_i = \{v_1, \ldots, v_i\}$ é base de $\mathcal{K}_i$.

1) $V_1 = \{v_1\}$ gera $\mathcal{K}_1 = K_1(A, v_1)$.

2) Demonstraremos que

$$K_k(A, v_1) = [v_1, \ldots, A^{k-1}v_1] = [v_1, \ldots, v_k]$$

implica

$$K_{k+1}(A, v_1) = [v_1, \ldots, A^k v_1] = [v_1, \ldots, v_{k+1}].$$

Temos que

$$\begin{aligned} A^k v_1 &= A(A^{k-1}v_1) \\ &= A\Big(\sum_{j=1}^{k} a_j v_j\Big), \end{aligned}$$

pois, pela hipótese da indução, $A^{k-1}v_1 \in [v_1, \ldots, v_k]$. Segue que

$$A^k v_1 = \Big(\sum_{j=1}^{k} a_j A v_j\Big).$$

Do passo 5, do Algoritmo 5.1, temos $Av_j = \tilde{v}_{j+1} + \alpha_j v_j + \beta_j v_{j-1}$, onde $\tilde{v}_{j+1} = \delta_{j+1} v_{j+1}$. Assim,

$$\begin{aligned} A^k v_1 &= \sum_{j=1}^{k} a_j(\delta_{j+1}v_{j+1} + \alpha_j v_j + \beta_j v_{j-1}) \\ &= \sum_{j=1}^{k+1} b_j v_j, \text{ com } b_j = \begin{cases} a_j\delta_j & , \quad j = k+1 \\ a_j\alpha_j + a_{j-1}\delta_j & , \quad j = k \\ a_{j+1}\beta_{j+1} + a_j\alpha_j + a_{j-1}\delta_{j-1} & , \quad j = 1, \ldots, k-1 \end{cases} \end{aligned}$$

Então,

$$A^k v_1 \in [v_1, \ldots, v_{k+1}] \Rightarrow [v_1, \ldots, A^k v_1] \subset [v_1, \ldots, v_{k+1}]. \tag{5.1}$$

Dos passos 5 e 10 , do Algoritmo 5.1, segue que

$$v_{k+1} = \frac{1}{\delta_{k+1}}(Av_k - \alpha_k v_k - \beta_k v_{k-1}). \tag{5.2}$$

Pela hipótese da indução,

$$v_k \in [v_1, \ldots, A^{k-1}v_1] \;\Rightarrow\; v_k = \sum_{j=1}^{k} a_j A^{j-1} v_1, \tag{5.3}$$

$$v_{k-1} \in [v_1, \ldots, A^{k-2}v_1] \;\Rightarrow\; v_{k-1} = \sum_{j=1}^{k-1} b_j A^{j-1} v_1. \tag{5.4}$$

Substituindo as (5.3) e (5.4) em (5.2),

$$\begin{aligned}
v_{k+1} &= \frac{1}{\delta_{k+1}}(A\sum_{j=1}^{k} a_j A^{j-1}v_1 - \alpha_k \sum_{j=1}^{k} a_j A^{j-1}v_1 - \beta_k \sum_{j=1}^{k-1} b_j A^{j-1}v_1) \\
&= \frac{1}{\delta_{k+1}}(\sum_{j=1}^{k} a_j A^{j}v_1 - \alpha_k \sum_{j=1}^{k} a_j A^{j-1}v_1 - \beta_k \sum_{j=1}^{k-1} b_j A^{j-1}v_1) \\
&= \frac{1}{\delta_{k+1}} \sum_{j=1}^{k} c_j A^j v_1, \text{ onde } c_j = \begin{cases} a_j & , \quad j = k \\ -\alpha_k a_{j+1} + a_{j-1} & , \quad j = k-1 \\ -\alpha_k a_j - \beta_k b_j + a_{j-1} & , \quad j = 2, \ldots, k-1 \\ -\alpha_k a_j - \beta_k b_j & , \quad j = 1 \end{cases}
\end{aligned}$$

Então,

$$v_{k+1} \in [v_1, \ldots, A^k v_1] \Rightarrow [v_1, \ldots, v_{k+1}] \subset [v_1, \ldots, A^k v_1] \tag{5.5}$$

De (5.1) e (5.5), concluímos que $K_i(A, v_1) = [v_1, \ldots, A^{i-1}v_1] = [v_1, \ldots, v_i]$. Analogamente, provaríamos que $\mathcal{L}_i = K_i(A^H, w_1) = [w_1, \ldots, w_i]$. $\diamond$

Teorema 5.2 *Caso o Algoritmo 5.1 cumpra m passos temos ,então,*

$$(v_i, w_j) = \begin{cases} 1, se \quad i = j \\ \\ 0, se \quad i \neq j \end{cases}, 1 \leq i, j \leq m.$$

Prova [26].

Teorema 5.3 *Caso o Algoritmo 5.1 cumpra m passos, este gera bases bi-ortogonais, V e W, aos subespaços $\mathcal{K} = K_m(A, v_1)$ e $\mathcal{L} = K_m(A^H, w_1)$, respectivamente.*

Prova

A demonstração decorre imediatamente dos Teoremas 5.1 e 5.2. $\diamond$

Em aritmética exata, o ponto central do algoritmo de Lanczos são os passos

5 e 6, onde temos três vetores envolvidos em cada equação, v_{j-1}, v_j, v_{j+1}, e w_{j-1}, w_j, w_{j+1}. Cada uma delas é chamada *recorrência de três termos* [13], [28]. Nessa formulação, cada vetor das bases, V e W, tem norma unitária e $(v_j, w_j) = 1$.

Proposição 5.1 *Seja o Algoritmo 5.1. Então, existe um operador de projeção P_i que projeta o vetor Av_i ortogonalmente a $\mathcal{L}_i$ sobre o subespaço*[1] *$\mathcal{K}_i$, onde $i = 1, \ldots, m$. A matriz que representa tal operador é dada por*

$$P_i = V_i W_i^H,$$

onde V_i e W_i, são bases bi-ortogonais de $\mathcal{K}_i$ e $\mathcal{L}_i$, respectivamente, geradas pelo Algoritmo 5.1.

Temos, também, o operador de projeção $Q_i = P_i^H$, que projeta o vetor $A^H w_i$ sobre o subespaço $\mathcal{L}_i$, ortogonalmente a $\mathcal{K}_i$, onde $i = 1, \ldots, m$. A matriz que representa tal operador é dada por

$$Q_i = W_i V_i^H.$$

Prova

Considerando o passo 5, e fazendo $\alpha_i v_i + \beta_i v_{i-1} = V_i y = P_i(x)$, temos

$$W_i^H \tilde{v}_{i+1} = W_i^H A v_i - W_i^H V_i y.$$

Pelo Teorema 5.3, temos que $W_i^H \tilde{v}_{i+1} = 0$ e $W_i^H V_i = I$, então

$$y = W_i^H A v_i,$$

ou seja,

$$P_i(Av_i) = V_i W_i^H A v_i \Rightarrow P_i = V_i W_i^H.$$

Observe que $P_i^2 = P_i$, logo, P_i é uma projeção. Analogamente, podemos mostrar que $Q_i = W_i V_i^H$.

Para provarmos que $Av_i - P(Av_i) \perp \mathcal{L}_i$, devemos provar que

$$(Av_i - P(Av_i), w_k) = 0 \quad \text{para} \quad k = 1, \ldots, i.$$

Como

$$P_i(Av_i) = V_i W_i^H A v_i,$$

[1] Os subespaços, $\mathcal{L}_i$ e $\mathcal{K}_i$, são dados por: $\mathcal{L}_i = K_i(A^H, w_1)$ e $\mathcal{K}_i = K_i(A, v_1)$.

temos que

$$\begin{aligned}(Av_i - P_i(Av_i), w_k) &= (Av_i - V_i W_i^H Av_i, w_k) = W_i^H (Av_i - V_i W_i^H Av_i) \\ &= W_i^H Av_i - W_i^H V_i W_i^H Av_i,\end{aligned}$$

V_i e W_i são bi-ortogonais. Assim,

$$W_i^H V_i W_i^H Av_i = W_i^H Av_i,$$

então,

$$(Av_i - P_i Av_i, w_k) = 0, \quad \text{para} \quad k = 1, \ldots, i,$$

ou seja, P_i projeta o vetor Av_i ortogonalmente a $\mathcal{L}_i$ sobre o subespaço $\mathcal{K}_i$. Poderíamos provar um resultado análogo para Q_i. $\diamond$

Capítulo 6

Método de Arnoldi

6.1 Introdução

Neste capítulo, vamos apresentar um dos principais métodos iterativos para se gerar uma base ortonormal para um subespaço de Krylov. Os teoremas apresentados serão demonstrados detalhadamente, por serem fundamentais na compreensão do método. Abordando o método de Arnoldi [14], sob o ponto de vista das projeções, teremos como resultado uma proposição que demonstra a existência do operador de projeção, bem como sua representação matricial. Tal operador é usado neste método.

6.2 Método de Arnoldi

O trabalho de Arnoldi [14] é baseado no artigo de Lanczos [13], onde foi discutido o método das "iterações minimizadas"(*minimized iterations*) [13]. Apresentou tal método, aplicando-o para solucionar a equação característica de uma matriz, tanto no caso homogêneo como no não-homogêneo. Como resultado, obteve uma redução no "volume de trabalho numérico" em aplicações práticas. É importante frisar que, assim como Lanczos, Arnoldi busca soluções para problemas de autovalores. Vamos abordar parte desse trabalho, onde temos um algoritmo que constrói uma base ortonormal, $V = \{v_1, v_2, \ldots, v_m\}$, de um subespaço de Krylov, $K_m(A, v_1)$. Este método é conhecido como método de Arnoldi [14]. A ortonormalização admite diferentes procedimentos, um deles é o método de Gram-Schmidt.

Algoritmo 6.1. Algoritmo de Arnoldi

1. Escolha um vetor v_1 com $\|v_1\| = 1$.
2. For $j = 1, \ldots, m-1$, do
3. For $i = 1, 2, \ldots, j$, do
4. $h_{i,j} = (Av_j, v_i)$
5. end
6. $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
7. $h_{j+1,j} = \|\hat{v}_{j+1}\|$ if $h_{j+1,j} = 0$ then end
8. $v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$
9. end.

Em termos matriciais, os passos 2 a 8 do Algoritmo 6.1 podem ser escritos como $V_m^H A V_m = H_m$, onde H_m é uma matriz de Hessenberg superior, ou seja, $h_{i,j} = 0$, para $i > j + 1$. Para demonstração de tal afirmação, ver Teorema 2.8.

A proposição a seguir prova que os vetores $v_1, v_2, \ldots, v_m$ gerados pelo Algoritmo 6.1 geram o subespaço de Krylov $K_m(A, v_1)$.

Teorema 6.1 *Seja o Algoritmo 6.1. Então,*

$$[v_1, Av_1, \ldots, A^{m-1}v_1] = [v_1, v_2, \ldots, v_m].$$

Prova

Vamos fazer uma indução matemática sobre a ordem da base.

1) Para $k = 1$, temos $[v_1] = [v_1]$, ou seja, o caso base se verifica.

2) Se

$$[v_1, Av_1, \ldots, A^{k-1}v_1] = [v_1, v_2, \ldots, v_k].$$

Então,

$$[v_1, Av_1, \ldots, A^{k-1}v_1, A^k v_1] = [v_1, v_2, \ldots, v_k, v_{k+1}].$$

Temos que

$$A^k v_1 = A(A^{k-1}v_1).$$

Por hipótese, $A^{k-1}v_1 \in [v_1, v_2, \ldots, v_k]$. Então, existem $a_j \in \mathbb{C}$, com $j = 1, \ldots, k$, tais que

$$A^{k-1}v_1 = \sum_{j=1}^{k} a_j v_j.$$

Assim,

$$A^k v_1 = A\Big(\sum_{j=1}^{k} a_j v_j\Big) = \sum_{j=1}^{k} a_j A v_j.$$

Entretanto, pelo passo 6 do Algoritmo 6.1, temos que

$$\begin{aligned} Av_j &= h_{j+1,j} v_{j+1} + \sum_{i=1}^{j} h_{i,j} v_i \\ &= \sum_{i=1}^{j+1} h_{i,j} v_j, \\ &\Rightarrow A^k v_1 \in [v_1, v_2, \ldots, v_k, v_{k+1}], \\ &\Rightarrow [v_1, Av_1, \ldots, A^{k-1}v_1, A^k v_1] \subset [v_1, v_2, \ldots, v_k, v_{k+1}]. \end{aligned} \tag{6.1}$$

Resta provar que

$$v_{k+1} \in [v_1, Av_1, \ldots, A^{k-1}v_1, A^k v_1].$$

Dos passos 6 e 8 do Algoritmo 6.1, segue que

$$v_{k+1} = \frac{1}{h_{k+1,k}}\Big(Av_k - \sum_{j=1}^{k} h_{j,k} v_j\Big).$$

Como, por hipótese, $v_1, \ldots, v_k \in [v_1, Av_1, \ldots, A^{k-1}v_1]$, então existem $b_j \in \mathbb{C}$, $j = 0, \ldots, k-1$, tais que

$$\sum_{j=1}^{k} h_{j,k} v_j = \sum_{j=0}^{k-1} b_j A^j v_1.$$

Temos também que $v_k \in [v_1, Av_1, \ldots, A^{k-1}v_1]$, então existem $c_j \in \mathbb{C}, j = 0, \ldots, k-1$ tais que

$$v_k = \sum_{j=0}^{k-1} c_j A^j v_1.$$

Assim,

$$Av_k = \sum_{j=1}^{k} c_j A^j v_1.$$

Então,

$$v_{k+1} = \frac{1}{h_{k+1,k}}\Big(\sum_{j=1}^{k} c_j A^j v_1 - \sum_{j=0}^{k-1} b_j A^j v_1\Big).$$

Segue que

$$\begin{aligned} &\Rightarrow v_{k+1} \in [v_1, Av_1, \ldots, A^{k-1}v_1, A^k v_1] \\ &\Rightarrow [v_1, v_2, \ldots, v_k, v_{k+1}] \subset [v_1, Av_1, \ldots, A^{k-1}v_1, A^k v_1]. \end{aligned} \tag{6.2}$$

Das equações (6.1) e (6.2), podemos concluir que

$$[v_1, Av_1, \ldots, A^k v_1] = [v_1, v_2, \ldots, v_{k+1}]. \qquad \diamond$$

Na próxima proposição, vamos obter uma outra base para o subespaço de Krylov $K_m(A, v_1)$.

Teorema 6.2 *Seja o Algoritmo 6.1. Então,*

$$[v_1, v_2, \ldots, v_m] = [v_1, Av_2, \ldots, Av_{m-1}].$$

Prova

Pelo passo 6 do Algoritmo 6.1, temos que

$$h_{j+1,j} v_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i.$$

Então,

$$Av_j = \sum_{i=1}^{j+1} h_{i,j} v_i, \quad \forall j = 1, \ldots, k.$$

Assim,

$$\begin{aligned} &\Rightarrow \quad Av_j \in [v_1, v_2, \ldots, v_{k+1}], \quad \forall j = 1, \ldots, k; \\ &\Rightarrow \quad [v_1, Av_2, \ldots, Av_k] \subset [v_1, v_2, \ldots, v_{k+1}]. \end{aligned} \tag{6.3}$$

Faremos uma indução matemática para a outra inclusão.

1) O caso base se verifica, pois $[v_1] \subset [v_1]$

2)Provaremos que

$$[v_1, v_2, \ldots, v_k] \subset [v_1, Av_2, \ldots, Av_{k-1}]$$

implica

$$[v_1, v_2, \ldots, v_{k+1}] \subset [v_1, Av_2, \ldots, Av_k].$$

Sabemos que

$$v_{k+1} = \frac{1}{h_{k+1,k}} \Big(Av_k - \sum_{i=1}^{k} h_{i,k} v_i \Big).$$

Como $v_1, \ldots, v_k \in [v_1, Av_2, \ldots, Av_{k-1}]$, por hipótese, segue que

$$v_p = a_{1p} v_1 + \sum_{s=2}^{k-1} a_{sp} Av_s, \text{ para } p = 1, \ldots, k.$$

Assim

$$\begin{aligned} v_{k+1} &= \frac{1}{h_{k+1,k}}\Big(Av_k - \sum_{i=1}^{k} h_{i,k}(a_{1i}v_1 + \sum_{j=2}^{k-1} a_{ji}Av_j)\Big) \\ &= b_1 v_1 + \sum_{j=2}^{k} b_j Av_j \end{aligned}$$

Então,

$$\begin{aligned} &\Rightarrow \quad v_{k+1} \in [v_1, Av_2, \ldots, Av_k], \\ &\Rightarrow \quad [v_1, v_2, \ldots, v_{k+1}] \subset [v_1, Av_2, \ldots, Av_k]. \end{aligned} \tag{6.4}$$

Das equações (6.3) e (6.4), temos que

$$[v_1, v_2, \ldots, v_{k+1}] = [v_1, Av_2, \ldots, Av_k]. \qquad \diamond$$

A partir dos Teoremas 6.1 e 6.2, temos que

$$[v_1, Av_1, \ldots, A^{m-1}v_1] = [v_1, v_2, \ldots, v_m] = [v_1, Av_2, \ldots, Av_{m-1}],$$

ou seja, temos três conjuntos geradores para o subespaço de Krylov $K_m(A, v_1)$.
Na proposição a seguir vamos analisar o método de Arnoldi [14], usando o enfoque de projeções.

Proposição 6.1 *Seja o Algoritmo 6.1. Então, existe um operador de projeção P_i, que projeta ortogonalmente o vetor Av_i sobre o subespaço de Krylov $\mathcal{K}_i = K_i(A, v_1)$, onde $i = 1, \ldots, m$. A matriz que representa tal operador é dada por:*

$$P_i = V_i V_i^H,$$

onde V_i é uma base ortonormal de $\mathcal{K}_i$.

Prova

Temos V_i uma base para o subespaço de Krylov, $\mathcal{K}_i$. Considerando o passo 6 do Algoritmo 6.1, teremos

$$\hat{v}_{i+1} = Av_i - \sum_{t=1}^{i} h_{t,i} v_t.$$

Façamos $V_i y = \sum_{t=1}^{i} h_{t,i} v_t = P(x)$

$$\begin{aligned} \hat{v}_{i+1} &= Av_i - V_i y \\ V_i^H \hat{v}_{i+1} &= V_i^H x - V_i^H V_i y \end{aligned}$$

Mas, por hipótese, as colunas de V_i são ortogonais a $\hat{v}_{i+1}$. Assim, $V_i^H \hat{v}_{i+1} = 0$ e temos também que $V_i^H V_i = I$, então:

$$y = V_i^H A v_i, \text{ assim } \quad P_i(Av_i) = V_i V_i^H A v_i, \Rightarrow P_i = V_i V_i^H.$$

Observe que $P_i^2 = P_i$, logo P_i é uma projeção.

Temos $P_i = V_i V_i^H = (V_i V_i^H)^H = P_i^H$, P ortogonal, ou seja, P_i projeta ortogonalmente Av_i sobre $\mathcal{K}_i$.

Podemos concluir que a projeção $P_i = V_i V_i^H$ projeta ortogonalmente o vetor Av_i sobre o subespaço de Krylov, $K_i(A, v_1)$, com $i = 1, \ldots, m$. $\quad \diamond$

Capítulo 7

Método do Resíduo Mínimo Generalizado (GMRES)

7.1 Introdução

O método do Resíduo Mínimo Generalizado (GMRES) foi proposto por Saad e Schultz [15] no ano de 1986. É um método iterativo, para resolução de sistemas lineares, onde a cada passo a norma do resíduo é minimizada sobre um subespaço de Krylov escolhido. O GMRES [15] utiliza o método de Arnoldi [14] para o cálculo de uma base ortonormal para um subespaço de Krylov.

Inicialmente, apresentaremos os métodos do Resíduo Conjugado Generalizado (GCR) [16] e da Ortogonalização Completa (FOM) [14]. Posteriormente, apresentaremos o GMRES [15] na sua forma clássica. Faremos uma análise dos métodos sob o enfoque de projeções sobre subespaços de Krylov.

Neste capítulo, serão apresentadas cinco proposições, que demonstram a existência e representação matricial dos operadores de projeção, que fazem parte dos métodos estudados.

7.2 Método do Resíduo Mínimo Generalizado

7.2.1 Método do Resíduo Conjugado Generalizado

Vamos abordar o método do Resíduo Conjugado Generalizado (GCR) [16], método iterativo utilizado para resolver sistemas lineares da forma $Ax = b$, onde A é uma matriz de ordem n. Neste método, devemos ter como hipótese que a parte simétrica[1] de A seja positiva-definida . O GCR [16] é matematicamente equivalente ao GMRES [15], ou seja, os dois métodos produzem a mesma aproximação, x_k, em aritmética exata.

Segue o algoritmo do GCR [16].

Algoritmo 7.1. Algoritmo do Resíduo Conjugado Generalizado

1. Escolha x_0 e calcule $r_0 = b - Ax_0$, $p_0 = r_0$
2. Para $j = 0, \ldots$ até convergir **Do**
3. $\quad \alpha_j = \frac{(r_j, Ap_j)}{(Ap_j, Ap_j)}$
4. $\quad x_{j+1} = x_j + \alpha_j p_j$
5. $\quad r_{j+1} = r_j - \alpha_j Ap_j$
6. $\quad$ Para $i = 1, \ldots, j$ **Do**
7. $\quad\quad$ Calcule $\beta_{ij} = -\frac{(Ar_{j+1}, Ap_i)}{(Ap_i, Ap_i)}$
8. $\quad$ **EndDo**
9. $\quad p_{j+1} = r_{j+1} + \sum_{i=0}^{j} \beta_{ij} p_i$
10. **EndDo**

Teorema 7.1 *Considerando o Algoritmo 7.1, as seguintes relações ocorrem:*

1. $(Ap_i, Ap_j) = 0$, *se* $i \neq j$;
2. $(r_i, Ap_j) = 0$, *se* $i > j$;
3. $(r_i, Ap_i) = (r_i, Ar_i)$;
4. $(r_i, Ar_j) = 0$, *se* $i > j$;
5. $(r_i, Ap_i) = (r_0, Ap_i)$, *se* $i \geq j$;
6. $\langle p_0, \ldots, p_i \rangle = \langle p_0, Ap_0, \ldots, A^i p_0 \rangle = \langle r_0, \ldots, r_i \rangle$;
7. *Se* $r_i \neq 0$, *então* $p_i \neq 0$;
8. x_{i+1} *minimiza* $E(\omega) = \|b - A\omega\|_2$ *sobre o espaço afim* $x_0 + \langle p_0, \ldots, p_i \rangle$.

[1] Toda matriz A de ordem n pode ser escrita como $A = A_1 + A_2$, onde , $A_1 = A_1^T$, A_1 é simétrica e, $A_2 = -A_2^T$, A_2 é dita anti-simétrica.

Prova [16]

Teorema 7.2 *Seja o Algoritmo 7.1. Então,*

$$x_m \in x_0 + K_m(A, r_0).$$

Prova

Vamos usar indução matemática.

1) $x_0 \in x_0 + K_m(A, r_0)$, $\forall m \in \mathbb{N}$.

2) Suponhamos que

$$x_k \in x_0 + K_k(A, r_0).$$

Existe um polinômio $P_{k-1}(A)$, tal que

$$x_k = x_0 + P_{k-1}(A)r_0.$$

Queremos provar que

$$x_{k+1} \in x_0 + K_{k+1}(A, r_0).$$

Pelo passo 4 do Algoritmo 7.1, temos

$$x_{k+1} = x_k + \alpha_k p_k.$$

Pela hipótese da indução, $x_{k+1} = x_0 + P_{k-1}(A)r_0 + \alpha_k p_k$. Do Teorema 7.1, item 6, temos que $p_k \in K_{k+1}(A, r_0)$, então existe $Q_k(A)$, tal que $p_k = Q_k(A)r_0$. Assim,

$$\begin{aligned} x_{k+1} &= x_0 + P_{k-1}(A)r_0 + \alpha_k Q_k(A)r_0 \\ &= x_0 + T_k(A)r_0, \text{ com } T_k(A) = P_{k-1}(A) + \alpha_k Q_k(A). \end{aligned}$$

Segue que $x_{k+1} \in x_0 + K_{k+1}(A, r_0)$.

Logo,

$$x_m \in x_0 + K_m(A, r_0), \ \forall m \in \mathbb{N}. \qquad \diamond$$

Teorema 7.3 *Seja o Algoritmo 7.1. Considere $\mathcal{K} = K_m(A, p_0)$ e $\mathcal{L} = A\mathcal{K}$ subespaços de Krylov. Então, o conjunto $\{Ap_0, Ap_1, \ldots, Ap_{m-1}\}$ é uma base de $\mathcal{L}$.*

Prova

Do item 1, Teorema 7.1, segue que o conjunto $\{Ap_0, Ap_1, \ldots, Ap_{m-1}\}$ é um conjunto ortogonal. Como $dim(\mathcal{L}) = m$, temos que o conjunto dado é uma base de $\mathcal{L}$. $\diamond$

Proposição 7.1 *Dado o Algoritmo 7.1. Existe uma projeção ortogonal $P_1^{(i)}$ sobre $\mathcal{L}_i$, que projeta o resíduo r_i, dada por*

$$P_1^{(i)} = W_i(W_i^H W_i)^{-1}W_i^H \quad ou \quad P_1^{(i)} = AV_i[(AV_i)^H AV_i]^{-1}(AV_i)^H,$$

onde V_i e W_i são bases de $\mathcal{K}_i = K_{i+1}(A, p_0)$ e $\mathcal{L}_i = A\mathcal{K}_i$, respectivamente.

Prova

Do passo 5 do Algoritmo 7.1, temos que

$$r_{i+1} = r_i - \alpha_i A p_i,$$

onde $\alpha_i A p_i = W_i y$, com $y \in \mathbb{C}^m$. Podemos concluir essa afirmação a partir do Teorema 7.3.

Do Teorema 7.1, item 2, temos que

$$(r_{i+1}, Ap_j) = 0, \text{ para } j = 1, \ldots, i.$$

Então,

$$\begin{aligned} W_i^H r_{i+1} &= W_i^H(r_i - W_i y) \\ y &= (W_i^H W_i)^{-1} W_i^H r_i. \end{aligned}$$

Observe que $(W_i^H W_i)^{-1}$ existe, pois W tem posto completo.

Temos a projeção $P_1^{(i)} = W_i(W_i^H W_i)^{-1}W_i^H$. $(P_1^{(i)})^2 = P_1^{(i)}$, além disso $P_1^{(i)}$ é Hermitiana, ou seja, $(P_1^{(i)})^H = P_1^{(i)}$, logo $P_1^{(i)}$ é ortogonal.$\diamond$

Observação:

As bases geradas pelo Algoritmo 7.1 não são ortonormais. Podemos concluir, apenas, que $W = \{Ap_0, Ap_1, \ldots, Ap_{m-1}\}$ é uma base ortogonal de $\mathcal{L}$, e, $V = \{p_0, p_1, \ldots, p_{m-1}\}$ é uma base $A^H A$-ortogonal de $\mathcal{K}$.

Proposição 7.2 *Dado o Algoritmo 7.1. Então, existe uma projeção oblíqua $P_2^{(i)}$ que projeta o erro, $e_i = x^* - x_i$, sobre $\mathcal{K}_i$, com direção ortogonal a $\mathcal{L}_i$, dada por*

$$P_2^{(i)} = V_i(W_i^H AV_i)^{-1}W_i^H A \quad ou \quad P_2^{(i)} = V_i(V_i^H A^H AV_i)^{-1}V_i^H A^H A.$$

Onde V_i e W_i são bases de $\mathcal{K}_i = K_{i+1}(A, p_0)$ e $\mathcal{L}_i = A\mathcal{K}_i$, respectivamente.

Prova

Do passo 5 do Algoritmo 7.1, temos que

$$\begin{aligned} r_{i+1} &= r_i - \alpha_i A p_i \\ &= r_i - A\alpha_i p_i. \end{aligned}$$

Temos

$$P_2^{(i)}(x) = V_i y = \alpha_i p_i.$$

Do Teorema 7.1, item 2, segue que

$$\begin{aligned} (r_{i+1}, Ap_j) &= 0, \text{ para } j = 1, \ldots, i. \\ (r_i - AV_i y, Ap_j) &= 0 \\ W_i^H (r_i - AV_i y) &= 0 \\ y &= (W_i^H AV_i)^{-1} W_i^H r_i. \end{aligned}$$

Como $r_i = Ae_i$, logo,

$$P_2^{(i)} = V_i (W_i^H AV_i)^{-1} W_i^H A \text{ ou } P_2^{(i)} = V_i (V_i^H A^H AV_i)^{-1} V_i^H A^H A. \qquad \diamond$$

7.2.2 Método da Ortogonalização Completa

O método da Ortogonalização Completa (FOM) [14] é um método iterativo para solução de sistemas lineares, foi apresentado em 1951 por Arnodi [14].

Considerando um sistema linear $Ax = b$, sendo A uma matriz de ordem n, o método consiste em determinar a solução x_m do sistema.

Usando o método de Arnoldi [14] de geração de bases ortonormais, determinamos uma base ortonormal, partindo-se das colunas da matriz A. Uma matriz H é também obtida após todas as iterações, como no método de Arnoldi [14].

Considerando uma solução inicial x_0 e o resíduo $r_0 = b - Ax_0$, tomamos o subespaço de Krylov $\mathcal{K}_m = K_m(A, r_0)$ e adotamos a hipótese de Galerkin, ou seja, o resíduo ortogonal ao subespaço considerado,

$$b - Ax_i \perp \mathcal{K}_{i+1},\ i = 0, \ldots, m-1.$$

Sendo $v_1 = \frac{r_0}{\|r_0\|}$ e aplicando o método de Arnoldi [14], temos

$$V_m^T A V_m = H_m.$$

A solução aproximada do sistema é dada por

$$\begin{aligned} x_m &= x_0 + V_m y_m, \quad \text{com} \\ y_m &= H_m^{-1}(\|r_0\| e_1). \end{aligned}$$

A seguir, vamos escrever o algoritmo do FOM [14], porém é importante que a característica principal desse algoritmo fique clara. No FOM [14], executamos os m passos da iteração e, ao final, obtemos a matriz H_m. Podemos então calcular y_m, usando a matrix inversa de H_m, e determinar a solução x_m do sistema $Ax = b$.
Vejamos o algoritmo da Ortogonalização Completa[14].

Algoritmo 7.2. Algoritmo da Ortogonalização Completa

1. Escolha um vetor x_0, compute $r_0 = b - Ax_0$ e $v_1 = r_0/\|r_0\|$.
2. Para $j = 1, \ldots, m-1$, **Do**
3. Para $i = 1, \ldots, j$ **Do**
4. $h_{i,j} = (Av_j, v_i)$
5. **EndDo**
6. $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
7. $h_{j+1,j} = \|\hat{v}_{j+1}\|$ if $h_{j+1,j} = 0$ go to 10
8. $v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$
9. **EndDo**
10. Forme a solução: $x_k = x_0 + V_k y_k$, onde $y_k = H_k^{-1}\|r_0\| e_1$.

Podemos observar que o algoritmo falha, se o grau do polinômio mínimo é no mínimo m, e a matriz H_m é não-singular.
Nesse algoritmo, obtemos o subespaço de Krylov $\mathcal{K}$, a partir de r_0 e da matriz A. Uma base ortonormal é obtida e, então, determina-se uma solução para o sistema linear considerado. Observe que uma solução aproximada não é obtida a cada nova direção, a solução é obtida ao final de todas as iterações.

Proposição 7.3 *Seja o Algoritmo 7.2. Então, existe uma projeção ortogonal P_i sobre o subespaço $\mathcal{K}_i$, que projeta o erro e_0, de tal forma que o resíduo r_i é ortogonal a $\mathcal{K}_i$. A matriz que representa tal operador é dada por*

$$P_i = V_i (V_i^H A V_i)^{-1} V_i^H A,$$

onde V_i é uma base ortonormal de $\mathcal{K}_i$.

Prova

Do Algoritmo 7.2, passo 10, temos que

$$x_i = x_0 + \delta_i, \text{ onde } \delta_i = V_i y_i \in \mathcal{K}.$$

Pela hipótese dada, segue que

$$\begin{aligned} (r_i, v_j) &= 0, \text{ para } j = 1, \ldots, i \\ (r_0 - AV_i y_i, v_j) &= 0 \\ V_i^H (r_0 - Av_i y_i) &= 0 \\ y_i &= (V_i A V_i)^{-1} V_i^H r_0. \end{aligned}$$

Assim,

$$\delta_i = V_i (V_i A V_i)^{-1} V_i^H e_0,$$

ou seja,

$$P_i = V_i (V_i^H A V_i)^{-1} V_i^H A.$$

7.2.3 Método do Mínimo Resíduo Generalizado

O método do Resíduo Mínimo Generalizado[15] é um método iterativo que vai determinar a solução aproximada x_i de um sistema linear da forma $Ax = b$.
Serão considerados os subespaços de Krylov $\mathcal{K}_i = K_i(A, r_0)$ e as soluções aproximadas serão da forma $x_0 + \delta_i$, onde $\delta_i \in \mathcal{K}_i$.
Tal algoritmo usa o algoritmo de Arnoldi [14], determinando, assim, uma base ortonormal V_m e a matriz de Hessenberg H_m.
Temos que a igualdade abaixo

$$V_{i+1}^H A V_i = H_{i+1,i}$$

é valida e podemos reescrevê-la como:

$$AV_i = V_{i+1} H_{i+1,i}.$$

O GMRES [15] tem como ponto principal minimizar a norma do resíduo sobre $\mathcal{K}_i$, ou seja, o que se quer é resolver o seguinte problema:

$$\min_{\delta_i \in \mathcal{K}_i} \|b - A(x_0 + \delta_i)\|_2 = \min_{\delta_i \in \mathcal{K}_i} \|r_0 - A\delta_i\|_2 \tag{7.1}$$

Façamos $\delta_i = V_i y_i$. Vamos definir a função J, a qual desejamos minimizar:

$$J(y_i) = \|r_0 - A\delta_i\|_2.$$

Seja $\beta = \|r_0\|_2$, segue que

$$\begin{aligned} J(y_i) &= \|\beta v_1 - AV_i y_i\|_2; \\ J(y_i) &= \|V_i(\beta e_1 - H_i y_i)\|_2, \end{aligned}$$

onde e_1 é o primeiro vetor da base canônica de $\mathbb{R}^m$.

Como V é uma base ortonormal, a função J fica sendo dada por

$$J(y_i) = \|\beta e_1 - H_i y_i\|_2$$

Assim, temos a seguinte solução para o problema de mínimos quadrados dado pela equação (7.1):

$$x_m = x_0 + V_m y_m,$$

onde y_m minimiza a função J em $\mathbb{R}^m$.

Segue o algoritmo do GMRES [15].

Algoritmo 7.3. Algoritmo do Resíduo Mínimo Generalizado

1. Escolha x_0 e calcule $r_0 = b - Ax_0$ e $v_1 = \frac{r_0}{\|r_0\|}$
2. *for* $j = 1, \ldots, m$
3. *for* $i = 1, \ldots, j$
4. $h_{ij} = (Av_j, v_i)$
5. $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$
6. $h_{j+1,j} = \|\hat{v}_{j+1}\|$ if $h_{j+1,j} = 0$ go to 9
7. $v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$
8. *end*
9. *end*
10. Forme a solução: $x_m = x_0 + V_m y_m$ onde y_m minimiza $J(y_i) = \|\beta e_1 - H_m y_m\|$, onde $\beta = \|r_0\|$

Uma questão natural que surge quando estudamos um algoritmo é:

Sob quais condições o algoritmo falha?

Teorema 7.4 *A solução x_j produzida pelo GMRES [15] no passo j é exata se, e somente se, as seguintes condições equivalentes ocorrem:*

1. *O algoritmo falha no passo j;*

2. $\hat{v}_{j+1} = 0$*;*

3. $h_{j+1,j} = 0$*;*

4. *O grau do polinômio mínimo do resíduo inicial r_0 é igual a j.*

Prova [15].

Corolário 7.1 *Para uma matriz de ondem $n \times n$, o GMRES [15] termina em no máximo n passos.*

Prova [15].

Teorema 7.5 *Seja A uma matriz não-singular. Então, o algoritmo do GMRES [15] falha no passo j, isto é, $h_{j+1,j} = 0$, se a solução aproximada x_j a solução exata.*

Prova [26].

Proposição 7.4 *Seja o Algoritmo 7.3. Então, existe uma projeção oblíqua P_i sobre o subespaço $\mathcal{K}_i$, que projeta o erro e_0, de tal forma que o resíduo r_i é ortogonal a $\mathcal{L}_i = A\mathcal{K}_i$. A matriz que representa tal operador é dada por*

$$P_i = V_i(V_i^H A^H A V_i)^{-1} V_i^H A^H A,$$

onde V_i e W_i são bases de $\mathcal{K}_i$ e $\mathcal{L}_i$, respectivamente.

Prova

Segue do passo 10, do Algortimo 5.3, que

$$r_i = r_0 - AV_i y_i.$$

Como queremos que a norma do resíduo seja mínima, devemos ter que o resíduo é ortogonal a $\mathcal{L}_i$, pois $\mathcal{L}_i = A\mathcal{K}_i$. Assim,

$$\begin{aligned}
(r_i, w_j) &= 0, \text{ para } j = 1, \ldots, i. \\
(r_0 - AV_i y_i, w_j) &= 0 \\
W_i^H (r_0 - AV_i y_i) &= 0 \\
y_i &= (W_i^H A V_i)^{-1} W_i^H r_0 \\
y_i &= (W_i^H A V_i)^{-1} W_i^H A e_0.
\end{aligned}$$

Então, P_i será dada por

$$P_i = V_i(W_i^H A V_i)^{-1} W_i^H A.$$

Como $W_i = \{Av_1, \ldots, Av_i\}$, segue que

$$P_i = V_i(V_i^H A^H A V_i)^{-1} V_i^H A^H A. \qquad \diamond$$

Capítulo 8

Método do Gradiente Bi-Conjugado Estabilizado (BiCGStab)

8.1 Introdução

O método do Gradiente Bi-Conjugado Estabilizado, desenvolvido por Van der Vorst [17], é um método iterativo para resolução de sistemas lineares, onde a matriz principal do sistema é qualquer.

Vamos estudar os algoritmos do Gradiente Bi-Conjugado [18] e do Gradiente Conjugado Quadrado [19], para que possamos abordar o algoritmo do Gradiente Bi-Conjugado Estabilizado [17].

Apresentaremos proposições sobre os algoritmos estudados, tendo como enfoque projeções sobre subespaços de Krylov convenientes.

8.2 Método do Gradiente Bi-Conjugado

Dado um sistema linear

$$Ax = b,$$

onde A é uma matriz de $\mathbb{C}^{n\times n}$. Sejam x_0 a aproximação inicial da solução do sistema e $r_0 = b - Ax_0$ o resíduo inicial.

Consideremos dois subespaços de Krylov,

$$\mathcal{K}_i = K_i(A, r_0) \quad \text{e} \quad \mathcal{L}_i = K_i(A^H, \hat{r}_0),$$

$\hat{r}_0$ é escolhido, tal que $(r_0, \hat{r}_0) \neq 0$.

As soluções aproximadas $x_i \in x_0 + K_i(A, r_0)$ serão obtidas de tal forma que o resíduo associado r_i seja ortogonal ao subespaço $\mathcal{L}_i$ e, $\hat{r}_i$, ortogonal a $\mathcal{K}_i$.

O método do BiCG [18] termina em n passos, mas não há nenhuma propriedade de minimização nos passos intermediários, como no método do Gradiente Conjugado [18] ou GMRES [15].

Abaixo, segue o algoritmo do Gradiente Bi-Conjugado [18].

Algoritmo 8.1. Algoritmo do Gradiente Bi-Conjugado

1. Seja x_0 uma solução inicial. Compute $r_0 = b - Ax_0$
2. Escolha $\hat{r}_0$ tal que $(r_0, \hat{r}_0) \neq 0$, por exemplo, $r_0 = \hat{r}_0$
3. $\rho_0 = 1$
4. $\hat{p}_0 = p_0 = 0$
5. for $i = 1, 2, 3, \ldots$
6. $\quad \rho_i = (\hat{r}_{i-1}, r_{i-1}); \beta_{i-1} = (\frac{\rho_i}{\rho_{i-1}})$
7. $\quad p_i = r_{i-1} + \beta_{i-1} p_{i-1}$
8. $\quad \hat{p}_i = \hat{r}_{i-1} + \beta_{i-1} \hat{p}_{i-1}$
9. $\quad v_i = Ap_i$
10. $\quad \alpha_i = \frac{\rho_i}{(\hat{p}_i, v_i)}$
11. $\quad x_i = x_{i-1} + \alpha_i p_i$
12. $\quad$ if x_i é suficientemente acurado então end
13. $\quad r_i = r_{i-1} - \alpha_i v_i$
14. $\quad \hat{r}_i = \hat{r}_{i-1} - \alpha_i A^H \hat{p}_i$
15. end

Teorema 8.1 *Os vetores produzidos pelo algoritmo do Gradiente Bi-Conjugado[18] satisfazem às seguintes propriedades de ortogonalidade:*

$$\begin{aligned} (r_j, \hat{r}_i) &= 0 \qquad \textit{para} \qquad i \neq j, \\ (Ap_j, \hat{p}_i) &= 0 \qquad \textit{para} \qquad i \neq j. \end{aligned}$$

Prova [26].

Do passo 2 ao 9, mais os passos 13 e 14, temos a geração das bases bi-ortogonais para os subespaços $\mathcal{K}$ e $\mathcal{L}$, respectivamente.

Podemos identificar as recorrências de três termos nos passos 7 e 8, e , também, em 13 e 14. Vê-se a relação entre este algoritmo e o da Bi-Ortogonalização de Lanczos [13]. De fato, aquele pode ser derivado a partir deste.

Vejamos algumas proposições que decorrem do Algoritmo 8.1, estabelecendo possíveis bases para o subespaço $\mathcal{K}$.

Teorema 8.2 *Dado o Algoritmo 8.1, então Ar_i pode ser escrito como combinação linear de $\{r_0, \ldots, r_{i+1}\}$, ou seja, existem escalares $y_j \in \mathbb{C}$, tais que*

$$Ar_i = \sum_{j=0}^{i+1} y_j r_j, \ i = 0, \ldots, m-1. \tag{8.1}$$

Prova

Usaremos indução matemática para desenvolver a demonstração.

1) Do passo 13 do Algoritmo 8.1 segue que

$$\begin{aligned} r_1 &= r_0 - \alpha_1 v_1 \\ &= r_0 - \alpha_1 A p_1, \text{ mas } \quad p_1 = r_1 \\ Ar_1 &= \frac{1}{\alpha_1}(r_0 - r_1). \end{aligned}$$

2) Suponhamos a equação (8.1), ou seja,

$$Ar_k = \sum_{j=0}^{k+1} y_j r_j.$$

Vamos provar que ela implica que (8.1) é valida para $k+1$.

Do passo 13, temos

$$\begin{aligned} r_{k+2} &= r_{k+1} - \alpha_{k+2} v_{k+2} \\ &= r_{k+1} - \alpha_{k+2} A p_{k+2} \\ &= r_{k+1} - \alpha_{k+2} A(r_{k+1} + \beta_{k+1} p_{k+1}) \\ \alpha_{k+2} A r_{k+1} &= -r_{k+2} + r_{k+1} - \alpha_{k+2}\beta_{k+1} A p_{k+1}. \end{aligned}$$

Pelos passos 9 e 13, $Ap_{k+1} = \frac{1}{\alpha_{k+1}}(r_k - r_{k+1})$. Assim

$$\alpha_{k+2} A r_{k+1} = -r_{k+2} + r_{k+1} - \frac{\alpha_{k+2}\beta_{k+1}}{\alpha_{k+1}}(r_k - r_{k+1}).$$

Então,

$$\begin{aligned} Ar_{k+1} &= \frac{1}{\alpha_{k+2}}\Big[\frac{\alpha_{k+2}\beta_{k+1}}{\alpha_{k+1}}r_k + \Big(1 - \frac{\alpha_{k+2}\beta_{k+1}}{\alpha_{k+1}}\Big)r_{k+1} - r_{k+2}\Big] \\ Ar_{k+1} &= \frac{\beta_{k+1}}{\alpha_{k+1}}r_k + \frac{\alpha_{k+1} - \alpha_{k+2}\beta_{k+1}}{\alpha_{k+1}\alpha_{k+2}}r_{k+1} - \frac{1}{\alpha_{k+2}}r_{k+2}, \end{aligned}$$

o que conclui a indução e a prova do teorema. $\diamond$

Teorema 8.3 *Seja o Algoritmo 8.1, então,*

$$K_i(A, r_0) = [r_0, r_1, \ldots, r_{i-1}]. \tag{8.2}$$

Prova

Façamos uma indução matemática sobre a dimensão do subespaço considerado.

1) Temos que

$$K_1(A, r_0) = [r_0].$$

2) Se

$$K_k(A, r_0) = [r_0, \ldots, r_{k-1}],$$

então,

$$K_{k+1}(A, r_0) = [r_0, \ldots, A^k r_0] = [r_0, \ldots, r_k].$$

a) $([r_0, \ldots, A^k r_0] \subset [r_0, \ldots, r_k])$

Sabemos que

$$A^k r_0 = A(A^{k-1} r_0),$$

mas, pela hipótese da indução, segue que

$$A^{k-1} r_0 = \sum_{j=0}^{k-1} z_j r_j.$$

Assim,

$$\begin{aligned} A^k r_0 &= A\Big(\sum_{j=0}^{k-1} z_j r_j\Big); \\ &= \sum_{j=0}^{k-1} z_j A r_j. \end{aligned} \tag{8.3}$$

Do Teorema 8.2, temos que

$$Ar_j = \sum_{s=0}^{j+1} y_s r_s. \tag{8.4}$$

Substituindo a equação (8.3) na equação (8.4), vem

$$\begin{aligned} A^k r_0 &= \sum_{j=0}^{k-1} z_j \Big(\sum_{s=0}^{j+1} y_s r_s \Big) \\ &= \sum_{j=0}^{k} t_j r_j. \end{aligned}$$

Então, $A^k r_0 \in [r_0, \ldots, r_k]$ o que implica $[r_0, \ldots, A^k r_0] \subset [r_0, \ldots, r_k]$.

b) ($[r_0, \ldots, r_k] \subset [r_0, \ldots, A^k r_0]$)

Dos passos 6 e 13, segue que

$$\begin{aligned} r_k &= r_{k-1} - \alpha_k v_k \\ &= r_{k-1} - \alpha_k A p_k. \end{aligned} \tag{8.5}$$

Do passo 7, temos,

$$p_k = r_{k-1} + \beta_{k-1} p_{k-1}. \tag{8.6}$$

Substituindo (8.6) em (8.5) vem

$$\begin{aligned} r_k &= r_{k-1} - \alpha_k A(r_{k-1} + \beta_{k-1} p_{k-1}) \\ &= r_{k-1} - \alpha_k A r_{k-1} - \alpha_k \beta_{k-1} A p_{k-1}. \end{aligned} \tag{8.7}$$

Da hipótese da indução, temos

$$r_{k-1} = \sum_{j=0}^{k-1} z_j A^j r_0. \tag{8.8}$$

Do passo 13, podemos determinar

$$A p_{k-1} = \frac{1}{\alpha_{k-1}} (r_{k-2} - r_{k-1}). \tag{8.9}$$

Substituindo (8.9) e (8.8) em (8.7), temos

$$\begin{aligned} r_k &= r_{k-1} - \alpha_k A r_{k-1} - \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}} (r_{k-2} - r_{k-1}) \\ &= -\frac{\alpha_k \beta_k}{\alpha_{k-1}} r_{k-2} + \Big(1 + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}\Big) r_{k-1} - \alpha_k A \Big(\sum_{j=0}^{k-1} z_j A^j r_0 \Big) \\ &= -\frac{\alpha_k \beta_k}{\alpha_{k-1}} r_{k-2} + \Big(1 + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}\Big) r_{k-1} - \sum_{j=0}^{k-2} \alpha_k z_j A^{j+1} r_0 + \alpha_k z_{k-1} A^k r_0 \\ &= -\frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}} r_{k-2} + \Big(1 + \frac{\alpha_k \beta_{k-1}}{\alpha_{k-1}}\Big) \sum_{j=0}^{k-1} z_j A^j r_0 - \\ &- \sum_{j=0}^{k-2} \alpha_k z_j A^{j+1} r_0 + \alpha_k z_{k-1} A^k r_0. \end{aligned} \tag{8.10}$$

Como $r_{k-2} \in K_{k-2}(A, r_0)$, podemos reescrever a equação (8.10):

$$\begin{aligned} r_k &= \alpha_k z_{k-1} A^k r_0 + \sum_{j=0}^{k-1} y_j A r_0 \\ &= \sum_{j=0}^{k} y_j A r_0, \text{ sendo } x_k = \alpha_k z_{k-1}. \end{aligned}$$

Então, $r_k \in [r_0, \ldots, A^k r_0]$. Segue que $[r_0, \ldots, r_k] \subset [r_0, \ldots, A^k r_0]$.

Logo, de (a) e (b), $[r_0, \ldots, r_k] = [r_0, \ldots, A^k r_0]$. $\diamond$

Teorema 8.4 *Seja o Algoritmo 8.1, então*

$$K_i(A, r_0) = [p_1, p_2, \ldots, p_i]. \tag{8.11}$$

Prova

Faremos uma indução matemática sobre a dimensão do subespaço $K_i(A, r_0)$.

1) Temos que $p_1 = r_0$, logo, $[p_1] = [r_0] = K_1(A, r_0)$.

2) Suponhamos a igualdade (8.11) para k, isto é,

$$K_k(A, r_0) = [p_1, p_2, \ldots, p_k]. \tag{8.12}$$

Provaremos que tal hipótese implica na igualdade

$$K_{k+1}(A, r_0) = [r_0, \ldots, A^k r_0] = [p_1, \ldots, p_{k+1}],$$

provando as duas inclusões a seguir:

a) ($[r_0, \ldots, A^k r_0] \subset [p_1, \ldots, p_{k+1}]$)

Do Teorema 8.3, segue que

$$A^k r_0 = \sum_{j=0}^{k} y_j r_j. \tag{8.13}$$

Do passo 7 do Algoritmo 8.1, temos

$$r_j = p_{j+1} - \beta_j p_j. \tag{8.14}$$

Substiuindo a equação (8.14) em (8.13),

$$\begin{aligned} A^k r_0 &= \sum_{j=0}^{k} y_j (p_{j+1} - \beta_j p_j) \\ &= \sum_{j=1}^{k+1} y_j p_j, \text{ pois } p_0 = 0. \end{aligned} \tag{8.15}$$

De (8.15), temos que $A^k r_0 \in [p_1, \ldots, p_{k+1}]$, ou seja, $[r_0, \ldots, A^k r_0] \subset [p_1, \ldots, p_{k+1}]$.

b) ($[p_1, \ldots, p_{k+1}] \subset [r_0, \ldots, A^k r_0]$)

Do passo 7 do Algoritmo 8.1, segue que

$$p_{k+1} = r_k + \beta_k p_k. \tag{8.16}$$

Do Teorema 8.3, temos

$$r_k = \sum_{j=0}^{k} y_j A^j r_0. \tag{8.17}$$

Da hipótese da indução (8.12), podemos escrever

$$p_k = \sum_{j=0}^{k-1} y_j A^j r_0 \tag{8.18}$$

Substituindo as equações (8.17) e (8.18) em (8.16), podemos concluir que

$$p_{k+1} = \sum_{j=0}^{k} z_j A^j r_0.$$

Assim, $p_{k+1} \in [r_0, \ldots, A^k r_0]$, então $[p_1, \ldots, p_{k+1}] \subset [r_0, \ldots, A^k r_0]$.

De (a) e (b), $K_{k+1}(A, r_0) = [r_0, \ldots, A^k r_0] = [p_1, \ldots, p_{k+1}]$. $\diamond$

Teorema 8.5 *Dado o Algoritmo 8.1, então existem os polinômios G_i, de grau i, e F_{i-1}, de grau $i-1$, tais que*

$$\begin{aligned} r_i &= G_i(A) r_0, \ com \ G_0(A) = I, \\ \hat{r}_i &= G_i(A^H) r_0, \ com \ G_0(A) = I, \end{aligned} \tag{8.19}$$

e

$$\begin{aligned} p_i &= F_{i-1}(A) r_0, \ com \ F_0(A) = I, \\ \hat{p}_i &= F_{i-1}(A^H) r_0, \ com \ F_0(A) = I. \end{aligned} \tag{8.20}$$

Prova

Segue dos Teoremas 8.3 e 8.4. $\diamond$

Teorema 8.6 *Dado o Algoritmo 8.1, a relação entre os polinômios $G_i(A)$ e $F_i(A)$, vistos em (8.19) e (8.20), respectivamente, é dada por*

$$\begin{aligned} G_i(A)r_0 &= (G_{i-1}(A) - \alpha_i A F_{i-1}(A))r_0, \\ F_i(A)r_0 &= (G_i(A) + \beta_i F_{i-1}(A))r_0, \end{aligned}$$

ou,

$$G_i(A) = G_{i-1}(A) - \alpha_i A F_{i-1}(A), \tag{8.21}$$

$$F_i(A) = G_i(A) + \beta_i F_{i-1}(A), \tag{8.22}$$

onde β_i e α_i são dados pelo Algoritmo 8.1.

Prova

A demonstração será feita usando a indução matemática.

1) Segue dos passos 7 e 13 do Algoritmo 8.1 que

$$\begin{aligned} r_1 &= r_0 - \alpha_1 A r_0, \\ p_2 &= r_1 + \beta_1 p_1, \end{aligned}$$

das equações (8.19) e (8.20), fazendo $r_1 = G_1(A)r_0$, $p_1 = F_0(A)r_0$ e $p_2 = F_1(A)r_0$, segue que

$$\begin{aligned} G_1(A)r_0 &= (G_0(A) - \alpha_1 A F_0(A))r_0, \\ F_1(A)r_0 &= (G_1(A) + \beta_1 F_0(A))r_0. \end{aligned}$$

2) Suponhamos que

$$\begin{aligned} F_k(A)r_0 &= (G_k(A) + \beta_k F_{k-1}(A))r_0, \\ G_k(A)r_0 &= (G_{k-1}(A) - \alpha_k A F_{k-1}(A))r_0. \end{aligned}$$

Queremos provar que implica em

$$\begin{aligned} F_{k+1}(A)r_0 &= (G_{k+1}(A) + \beta_{k+1} F_k(A))r_0, \\ G_{k+1}(A)r_0 &= (G_k(A) - \alpha_{k+1} A F_k(A))r_0. \end{aligned}$$

Dos passos 7 e 13, temos que

$$r_{k+1} = r_k - \alpha_k A p_k, \tag{8.23}$$

$$p_{k+2} = r_{k+1} + \beta_{k+1} p_{k+1}. \tag{8.24}$$

Das equações (8.19) e (8.20), temos

$$\begin{aligned} r_k &= G_k(A)r_0, \\ r_{k+1} &= G_{k+1}(A)r_0 \\ p_k &= F_{k-1}(A)r_0, \\ p_{k+1} &= F_k(A)r_0, \\ p_{k+2} &= F_{k+1}(A)r_0. \end{aligned}$$

Substituindo nas equações (8.23) e (8.24), finalizamos a demonstração do teorema. $\diamond$

Na proposição seguinte, veremos que o Algoritmo 8.1 usa uma projeção oblíqua sobre $\mathcal{K}_i = K_i(A, r_0)$, com direção ortogonal a $\mathcal{L}_i = K_i(A^H, \hat{r}_0)$.

Proposição 8.1 *Seja o Algoritmo 8.1. Então, existe uma projeção oblíqua P_i sobre o subespaço $\mathcal{K}_i = K_i(A, r_0)$, que projeta o erro e_0, de tal forma que o resíduo r_i é ortogonal a $\mathcal{L}_i = K_i(A^H, \hat{r}_0)$. Considere V_i e W_i bases de $\mathcal{K}_i$ e $\mathcal{L}_i$, respectivamente. A matriz que representa tal operador é dada por*

$$P_i = V_i(W_i^H A V_i)^{-1} W_i^H A.$$

Prova

Análoga à da Proposição 2.1.

8.3 Método do Gradiente Conjugado Quadrado

O método do Gradiente Conjugado Quadrado (CGS), apresentado por Sonneveld [19] em 1989, é aplicado na resolução de sistemas lineares, onde a matriz principal do sistema é não-simérica e esparsa.

Pode ser classificado como método de Lanczos [13], e uma de suas diferenças em relação ao BiCG [18] é que não utiliza a matriz adjunta na multiplicação matriz-vetor, a matriz adjunta nem mesmo chega a ser formada.

Vejamos do CGS [19].

Algoritmo 8.2. Algoritmo do Gradiente Conjugado Quadrado

1. Seja x_0 uma solução inicial. Compute $r_0 = b - Ax_0$
2. Escolha $\hat{r}_0$ tal que $(r_0, \hat{r}_0) \neq 0$, i.e., $r_0 = \hat{r}_0$
3. $\rho_0 = 1$; $p_0 = q_0 = 0$
4. **For** $i = 1, 2, 3, \ldots$
5. $\quad \rho_i = (\hat{r}_0, r_{i-1})$
6. $\quad \beta_{i-1} = (\frac{\rho_i}{\rho_{i-1}})$
7. $\quad u_i = r_{i-1} + \beta_{i-1} q_{i-1}$
8. $\quad p_i = u_i + \beta_{i-1}(q_{i-1} + \beta_{i-1} p_{i-1})$
9. $\quad v_i = Ap_i$
10. $\quad \alpha_i = \frac{\rho_i}{(\hat{r}_0, v_i)}$
11. $\quad q_i = u_i - \alpha_i v_i$
12. $\quad \hat{u}_i = u_i + q_i$
13. $\quad x_i = x_{i-1} + \alpha_i \hat{u}_i$
14. $\quad$ se x_i é suficientemente acurado então End
15. $\quad r_i = r_{i-1} - \alpha_i A\hat{u}_i$
16. Fim

Em (8.19), vimos que no método do BiCG [18], o resíduo r_i pode ser obtido como

$$r_{i(BiCG)} = G_i(A) r_0.$$

Teorema 8.7 *Dado o Algoritmo 8.2, temos que*

$$\begin{aligned} r_i &= G_i^2(A) r_0 \\ q_i &= G_i(A) F_{i-1}(A) r_0 \\ p_i &= F_{i-1}^2(A) r_0 \\ u_i &= G_{i-1}(A) F_{i-1}(A) r_0, \end{aligned}$$

onde $G_i(A)$ e $F_{i-1}(A)$ são dadas pelas equações (8.22) e (8.21), do Teorema 8.6.

Prova

Façamos uma indução matemática.

1) Para $i = 0$, temos

$$\begin{aligned} r_1 &= r_0 - \alpha_1(u_1 + q_1) \\ &= r_0 - \alpha_1 A(r_0 + \beta_0 q_0 + q_1) \\ &= r_0 - \alpha_1 A r_0 - \alpha_1 A(u_1 - \alpha_1 A p_1) \\ &= r_0 - \alpha_1 A r_0 - \alpha_1 A r_0 + \alpha_1^2 A p_1 \\ &= r_0 - 2\alpha_1 A r_0 + \alpha_1^2 A r_0 \\ &= G_1^2(A) r_0. \end{aligned}$$

e

$$\begin{aligned} q_1 &= r_0 - \alpha_1 A r_0 \\ &= [G_0(A) - \alpha_1 A F_0(A)] F_0(A) r_0 \\ &= G_1(A) F_0(A) r_0 \\ p_1 &= r_0 = F_0^2(A) r_0 \\ u_1 &= r_0 = G_0(A) F_0(A) r_0. \end{aligned}$$

2) Queremos provar que

$$r_k = G_k^2(A) r_0 \tag{8.25}$$

$$q_k = G_k(A) F_{k-1}(A) r_0 \tag{8.26}$$

$$p_k = F_{k-1}^2(A) r_0 \tag{8.27}$$

$$u_k = G_{k-1}(A) F_{k-1}(A) r_0. \tag{8.28}$$

Implica

$$\begin{aligned} r_{k+1} &= G_{k+1}^2(A) r_0 \\ q_{k+1} &= G_{k+1}(A) F_k(A) r_0 \\ p_{k+1} &= F_k^2(A) r_0 \\ u_{k+1} &= G_k(A) F_k(A) r_0. \end{aligned}$$

Das equações (8.21) e (8.22), temos que

$$G_{k+1}(A) = G_k(A) - \alpha_{k+1} A F_k(A), \tag{8.29}$$

$$F_k(A) = G_k(A) + \beta_k F_{k-1}(A), . \tag{8.30}$$

Segue que

$$\begin{aligned} G_{k+1}^2(A) &= G_k^2(A) - 2\alpha_{k+1}AG_k(A)F_k(A) + \alpha_{k+1}^2A^2F_k^2(A), \\ F_k^2(A) &= G_k^2(A) + 2\beta_kG_k(A)F_{k-1}(A) + \beta_k^2F_{k-1}(A)^2. \end{aligned}$$

Substituindo-se $F_k(A)$, dado pela equação (8.22), no produto $G_k(A)F_k(A)$, vem

$$G_k(A)F_k(A) = G_k^2(A) + \beta_kG_k(A)F_{k-1}(A). \tag{8.31}$$

Multiplicando a equação (8.21, com $i = k$), por $F_k(A)$, temos

$$G_k(A)F_{k-1}(A) = G_{k-1}(A)F_{k-1}(A) - \alpha_kAF_{k-1}(A)^2. \tag{8.32}$$

Assim, das equações (8.21, com $i = k$), (8.30), (8.31) e (8.32),

$$\begin{aligned} G_{k+1}(A)^2 &= G_k^2(A) - \alpha_{k+1}A(2G_k^2(A) + 2\beta_kG_k(A)F_{k-1}(A) - \alpha_{k+1}AF_k^2(A)), \\ G_{k+1}(A)F_k(A) &= G_k^2(A)F_k(A) - \alpha_{k+1}AF_k^2(A), \\ F_k^2(A) &= G_k^2(A) + 2\beta_kG_k(A)F_{k-1}(A) + \beta_k^2F_{k-1}(A)^2. \end{aligned}$$

Então,

$$\begin{aligned} G_{k+1}(A)^2 &= G_k^2(A) - \alpha_{k+1}A(2G_k^2(A) + 2\beta_kG_k(A)F_{k-1}(A) - \alpha_{k+1}AF_k^2(A)), \\ G_{k+1}(A)F_k(A) &= G_k^2(A) + \beta_kG_k(A)F_{k-1}(A) - \alpha_{k+1}AF_k^2(A), \\ F_k^2(A) &= G_k^2(A) + 2\beta_kG_k(A)F_{k-1}(A) + \beta_k^2F_{k-1}(A)^2, \\ G_k(A)F_k(A) &= G_k^2(A) + \beta_kG_k(A)F_{k-1}(A). \end{aligned}$$

Da hipótese da indução, dada pelas equações (8.25), (8.26) e (8.27), temos que

$$\begin{aligned} G_{k+1}(A)^2r_0 &= r_k - \alpha_{k+1}A(2r_k + 2\beta_kq_k - \alpha_{k+1}Ap_k) = \\ &= r_{k+1} \qquad (8.33) \\ G_{k+1}(A)F_k(A)r_0 &= r_k + \beta_kq_k - \alpha_{k+1}Ap_k = q_{k+1} \\ F_k^2(A)r_0 &= r_k + 2\beta_kq_k + \beta_k^2p_k = p_{k+1} \\ G_k(A)F_k(A) &= r_k + \beta_kq_k = u_{k+1}. \end{aligned}$$

Então,

$$\begin{aligned} r_i &= G_i^2(A)r_0 \\ q_i &= G_i(A)F_{i-1}(A)r_0 \\ p_i &= F_{i-1}^2(A)r_0 \\ u_i &= G_{i-1}(A)F_{i-1}(A)r_0. \qquad \diamond \end{aligned}$$

Observação:

Fazendo

$$\hat{u}_i = 2r_i + 2\beta_i q_{i-1} - \alpha_i A p_i,$$

segue da equação (8.33) o passo 15 do Algoritmo 8.2.

Sob o ponto de vista de operadores de projeção, a Proposição 8.1 também se aplica ao CGS [19]. Considerando o Algoritmo 8.2, não aparecem, explicitamente, as recorrências de três termos de Lanczos, para se obter as bases bi-ortogonais dos subespaços de Krylov $\mathcal{K}_i$ e $\mathcal{L}_i$. Porém, tal método pode ser obtido a partir do BiCG [18], ou seja, este também é um método onde o resíduo é obtido usando-se uma projeção oblíqua do erro e_0 sobre $\mathcal{K}_i$, com direção ortogonal a $\mathcal{L}_i$.

8.4 Método do Gradiente Bi-Conjugado Estabilizado

Desenvolvido por Van der Vorst em 1992, o método do Gradiente Bi-Conjugado Estabilizado [17] aplica-se para resolução de sistemas lineares com matrizes não-simétricas. Também pode ser classificado como um método de Lanczos [13].
O BiCGStab [17] pode ser obtido a partir do BiCG [18] e, como o CGS [19], não usa a matriz conjugada.
Vejamos o método do Gradiente Bi-Conjugado Estabilizado.

Algoritmo 8.3. Algoritmo do Gradiente Bi-Conjugado Estabilizado

1. Seja x_0 uma solução inicial. Compute $r_0 = b - Ax_0$
2. Escolha $\hat{r}_0$ tal que $(r_0, \hat{r}_0) \neq 0$, i.e., $r_0 = \hat{r}_0$
3. $\rho_0 = \alpha_0 = \omega_0 = 1$
4. $v_0 = p_0 = 0$
5. Para $i = 1, 2, 3 \ldots .$ **Do**
6. $\quad \rho_i = (r_0, \hat{r}_0)$; $\beta_{i-1} = (\frac{\rho_i}{\rho_{i-1}})(\frac{\alpha_{i-1}}{\omega_{i-1}})$
7. $\quad p_i = r_{i-1} + \beta_{i-1}(p_{i-1} - \omega_{i-1} v_{i-1})$
8. $\quad v_i = Ap_i$
9. $\quad \alpha_i = \frac{\rho_i}{(\hat{r}_0, v_i)}$
10. $\quad s_i = r_{i-1} - \alpha_i v_i$
11. $\quad t_i = As_i$
12. $\quad \omega_i = \frac{(t_i, s_i)}{(t_i, t_i)}$
13. $\quad x_i = x_{i-1} + \alpha_i p_i + \omega_i s_i$
14. $\quad$ se x_i é suficientemente acurado então **EndDo**
15. $\quad r_i = s_i - \omega_i t_i$
16. Fim

Teorema 8.8 *Dado o Algoritmo 8.3, temos que*

$$\begin{aligned} p_i &= Q_{i-1}(A) F_{i-1}(A) r_0 \\ r_i &= Q_i(A) G_i(A) r_0, \end{aligned}$$

onde $G_i(A)$ e $F_{i-1}(A)$ são dadas pelas equações (8.22) e (8.21), do Teorema 8.6 e

$$Q_{i+1}(A) = (I - \omega_i A) Q_i(A), \ \textit{com} \ \ Q_0(A) = I.$$

Prova

Vamos usar uma indução matemática para demonstrar este teorema.

1) Para $i = 1$, temos que

$$\begin{aligned} p_1 &= r_0 = Q_0(A) F_0(A) \\ r_1 &= s_1 - \omega_1 t_1 \\ &= r_0 - \alpha_1 A r_0 - \omega_1 A r_0 - \omega_1 \alpha_1 A^2 r_0 \\ &= (I - \omega_1 A) r_0 - (I - \omega_1 A) \alpha_1 A r_0 \\ &= (I - \omega_1 A)(I - \alpha_1 A) r_0 = Q_1(A) G_1(A). \end{aligned}$$

2) Provaremos que

$$\begin{aligned} p_k &= Q_{k-1}(A)F_{k-1}(A)r_0 \quad (8.34)\\ r_k &= Q_k(A)G_k(A)r_0 \quad (8.35)\end{aligned}$$

implica

$$\begin{aligned} p_{k+1} &= Q_k(A)F_k(A)r_0\\ r_{k+1} &= Q_{k+1}(A)G_{k+1}(A)r_0.\end{aligned}$$

Podemos calcular os seguintes produtos:

$$\begin{aligned} Q_k(A)F_k(A) &= (I-\omega_k A)Q_{k-1}(A)F_k(A)\\ Q_{k+1}(A)G_{k+1}(A) &= (I-\omega_{k+1}A)Q_k(A)G_{k+1}(A).\end{aligned}$$

Das equações (8.22) e (8.21), segue que

$$\begin{aligned} Q_k(A)F_k(A) &= (I-\omega_{k-1}A)Q_{k-1}(A)[G_k(A)+\beta_k F_{k-1}(A)]\\ Q_{k+1}(A)G_{k+1}(A) &= (I-\omega_{k+1}A)Q_k(A)[G_k(A)-\alpha_{k+1}AF_k(A)].\end{aligned}$$

Assim,

$$\begin{aligned} Q_k(A)F_k(A) &= (I-\omega_k A)[Q_{k-1}(A)G_k(A)+\beta_k Q_{k-1}(A)F_{k-1}(A)]\\ Q_{k+1}(A)G_{k+1}(A) &= (I-\omega_{k+1}A)[Q_k(A)G_k(A)-\alpha_{k+1}AQ_k(A)F_k(A)].\end{aligned}$$

Vamos calcular $Q_{k-1}(A)G_k(A)$:

$$\begin{aligned} Q_{k-1}(A)G_k(A) &= Q_{k-1}(A)[G_{k-1}(A)-\alpha_k AF_{k-1}(A)]\\ &= Q_{k-1}(A)G_{k-1}(A)-\alpha_k AQ_{k-1}(A)F_{k-1}(A). \quad (8.36)\end{aligned}$$

Da equação (8.36), segue que

$$\begin{aligned} Q_k(A)F_k(A) &= (I-\omega_k A)[Q_{k-1}(A)G_{k-1}(A)-\alpha_k AQ_{k-1}(A)F_{k-1}(A)+\\ &+ \beta_k Q_{k-1}(A)F_{k-1}(A)],\\ Q_{k+1}(A)G_{k+1}(A) &= (I-\omega_{k+1}A)[Q_k(A)G_k(A)-\alpha_{k+1}AQ_k(A)F_k(A)].\end{aligned}$$

Da hipótese da indução dada pelas equações (8.35) e (8.34),

$$\begin{aligned}
Q_k(A)F_k(A)r_0 &= (I-\omega_k A)(r_{k-1}-\alpha_k Ap_k+\beta_k p_k)\\
&= r_{k-1}-\alpha_k Ap_k+\beta_k p_k-\omega_k Ar_{k-1}-\omega_k\alpha_k A^2p_k+\omega_k\beta_k Ap_k\\
&= r_{k-1}-\alpha_k Ap_k-\omega_k A(r_{k-1}-\alpha_k Ap_k)+\beta_k p_k-\omega_k\beta_k Ap_k\\
&= s_k-\omega_k t_k+\beta_k p_k-\beta_k\omega_k Ap_k\\
&= r_k+\beta_k p_k-\beta_k\omega_k Ap_k\\
&= p_{k+1}\\
Q_{k+1}(A)G_{k+1}(A)r_0 &= (I-\omega_{k+1}A)(r_k-\alpha_{k+1}Ap_{k+1})\\
&= r_k-\alpha_{k+1}Ap_{k+1}-\omega_{k+1}Ar_k+\omega_{k+1}\alpha_{k+1}A^2p_{k+1}\\
&= r_k-\alpha_{k+1}Ap_{k+1}-\omega_{k+1}A(r_k-\alpha_{k+1}Ap_{k+1})\\
&= s_{k+1}-\omega_{k+1}t_{k+1}\\
&= r_{k+1},
\end{aligned}$$

concluindo, assim, a indução matemática. $\diamond$

A Proposição 8.1 também se aplica ao BiCGStab [17]. Tal método pode ser obtido a partir do BiCG [18], ou seja, neste método, o resíduo também é obtido usando-se uma projeção oblíqua do erro e_0 sobre $\mathcal{K}_i$, com direção ortogonal a $\mathcal{L}_i$.

Capítulo 9

Método do Gradiente Bi-Conjugado Estabilizado em Bloco (Bl-BiCGStab)

9.1 Introdução

Os métodos tratados neste capítulo são métodos iterativos em bloco para resolução de sistemas lineares da forma

$$AX = B, \tag{9.1}$$

onde $A \in \mathbb{C}^{n\times n}$ é uma matriz não-singular, $X, B \in \mathbb{C}^{n\times s}$.

Um dos objetivos dos algoritmos em bloco é o de resolver sistemas com múltiplos lados direitos de forma mais eficiente do que os algoritmos que não são em bloco. Nem sempre isso é possível, mas quando a matriz A é densa, ou quando trabalhamos com um pré-condicionador [15],pag. 262, o método pode ser bem atrativo.

O'Leary [20], em 1980, apresentou o método Bl-BiCG , discutiu sua implementação e resultados obtidos a partir de testes realizados.

Dai [29], no ano de 1998, propôs dois algoritmos: Bi-diagonalização de Lanczos em bloco e Bi-diagonalização do Mínimo Resíduo (Minres) em bloco, ambos para resolução de sistemas lineares não-simétricos com múltiplos lados direitos. Pelos seus testes, concluiu que os métodos de bi-diagonalização em bloco têm, na prática, uma melhor performance que o GMRES em bloco e o BiCG em bloco, para a resolução do problema proposto.

Guennouni , Jbilou e Sadok [7] (2003) apresentaram o método do Bl-BiCGSTab [7], partindo no método do Bl-BiCG, usando como ferramenta polinômios ortogonais de valores matriciais. Em [30] (2004), propuseram o método de Lanczos em bloco, cuja iteração é obtida com um complemento de Shur, e mostraram a conexão deste método como polinômios ortogonais de valores matriciais.

Iniciaremos com o algoritmo do Gradiente Bi-Conjugado em bloco , desenvolvido por O'Leary [20]. Proposições e teoremas relacionados a este método serão demonstrados, bem como a proposição obtida, usando o enfoque de projeções sobre subespaços de Krylov em bloco convenientes. A partir daí, veremos o algoritmo do Gradiente Bi-Conjugado Estabilizado em bloco [7],m como proposições e teoremas decorrentes deste.

9.2 Método do Gradiente Bi-Conjugado em Bloco (Bl-BiCG)

O algoritmo do Gradiente Bi-Conjugado em bloco (Bl-BiCG) foi proposto por O'Leary [20], sendo obtido como uma generalização do método do Gradiente Bi-Conjugado [18], [13].

Definição 9.1 *O subespaço de Krylov em bloco é gerado pelas matrizes $A \in \mathbb{C}^{n\times n}$ e $V \in \mathbb{C}^{n\times s}$, da seguinte forma:*

$$[V, AV, \ldots, A^{m-1}V].$$

Notamos tal subespaço por $K_m(A, V)$ [7], [20].

Vamos considerar uma solução inicial X_0 e o resíduo inicial $R_0 = B - AX_0$, com $X_0, R_0 \in \mathbb{C}^{n\times s}$. Sejam os subespaços de Krylov em bloco de dimensão i

$$\mathbf{K}_i = K_i(A, R_0) \quad \text{e} \quad \mathbf{L}_i = K_i(A^H, \tilde{R}_0),$$

onde $\tilde{R}_0$ é escolhida arbitrariamente e $R_0, \tilde{R}_0 \in \mathbb{C}^{n\times s}$ têm posto s.

As soluções aproximadas $X_i \in X_0 + K_i(A, R_0)$, serão obtidas de tal forma que o resíduo associado R_i seja ortogonal ao subespaço $\mathbf{L}_i$ e $\tilde{R}_i$ ortogonal a $\mathbf{K}_i$.

No passo i, teremos $X_i = X_0 + \Delta_i$, uma aproximação da solução do sistema (9.1), onde $\Delta_i \in \mathbf{K}_i$.

O método do Bl-BiCG termina em no máximo $\lceil n/s \rceil$[1] iterações , quando há s sistemas a serem resolvidos (ver [20]). Assim como o BCG [18], não há nenhuma propriedade de minimização nos passos intermediários, ou seja, não podemos assegurar que $\|R_{i+1}\| \leq \|R_i\|$.

Vejamos o algoritmo do Gradiente Bi-Conjugado em bloco [20].

Algoritmo 9.1. Algoritmo do Gradiente Bi-Conjugado em bloco

1. Seja $X_0 \in \mathbb{C}^{n\times s}$. Compute, $R_0 = B - AX_0$, matriz de posto s
2. Escolha $\tilde{R}_0 \in \mathbb{C}^{n\times s}$ matriz de posto s
3. $P_0 = R_0$, $\tilde{P}_0 = \tilde{R}_0$
4. Para $i = 0, 1, \ldots$ **Do**
5. $\quad \alpha_i = (\tilde{P}_i^H A P_i)^{-1} \tilde{R}_i^H R_i$
6. $\quad X_{i+1} = X_i + P_i \alpha_i$
7. $\quad R_{i+1} = R_i - A P_i \alpha_i$
8. $\quad \tilde{\alpha}_i = (P_i^H A^H \tilde{P}_i)^{-1} R_i^H \tilde{R}_i$
9. $\quad \tilde{R}_{i+1} = \tilde{R}_i - A^H \tilde{P}_i \tilde{\alpha}_i$
10. $\quad \beta_i = (\tilde{R}_i^H R_i)^{-1} \tilde{R}_{i+1}^H R_{i+1}$
11. $\quad \tilde{\beta}_i = (R_i^H \tilde{R}_i)^{-1} R_{i+1}^H \tilde{R}_{i+1}$
12. $\quad P_{i+1} = R_{i+1} + P_i \beta_i$
13. $\quad \tilde{P}_{i+1} = \tilde{R}_{i+1} + \tilde{P}_i \tilde{\beta}_i$
14. **EndDo**

O algoritmo falha se as matrizes $\tilde{P}_i^H A P_i$ e $\tilde{R}_i^H R_i$ são singulares. Para os teoremas e proposições a seguir, estamos considerando que o Algoritmo 9.1 não falha.

Teorema 9.1 *Dado o Algoritmo 9.1, então $AP_i\alpha_i$ e $A^H\tilde{P}_i\tilde{\alpha}_i$ podem ser escritos como combinação linear de $\{R_0, \ldots, R_{i+1}\}$ e $\{\tilde{R}_0, \ldots, \tilde{R}_{i+1}\}$, respectivamente, ou seja, existem escalares $a_j, \tilde{a}_j \in \mathbb{C}$, tais que*

$$AP_i\alpha_i = \sum_{j=0}^{i+1} a_j R_j; \tag{9.2}$$

$$A\tilde{P}_i\tilde{\alpha}_i = \sum_{j=0}^{i+1} \tilde{a}_j \tilde{R}_j. \tag{9.3}$$

[1] $\lceil n/s \rceil = n_0, n_0 \in \mathbb{N}$, n_0 é o menor natural tal que $n/s \leq n_0$.

Prova

Usaremos indução matemática para demonstrar (9.2).

1) Do passo 7 do Algoritmo 9.1, segue que

$$\begin{aligned} R_1 &= R_0 - AP_0\alpha_0, \\ AP_0\alpha_0 &= R_0 - R_1. \end{aligned}$$

2) Suponhamos que (9.2) se verifica para k, ou seja,

$$AP_k\alpha_k = \sum_{j=0}^{k+1} a_j R_j, \text{ com } \quad a_j \in \mathbb{C}.$$

Vamos provar que

$$AP_{k+1}\alpha_{k+1} = \sum_{j=0}^{k+2} b_j R_j, \text{ com } \quad b_j \in \mathbb{C}.$$

Do passo 7, temos que

$$\begin{aligned} R_{k+2} &= R_{k+1} - AP_{k+1}\alpha_{k+1} \\ AP_{k+1}\alpha_{k+1} &= R_{k+1} - R_{k+2} \\ &= R_k - AP_k\alpha_k - R_{k+2} \\ &= R_k - \sum_{j=0}^{k+1} a_j R_j - R_{k+2} \\ &= \sum_{j=0}^{k+2} b_j R_j \ , \text{ onde } b_j = \begin{cases} -a_j & , \ j \in \{0,\ldots,k+1\}\backslash\{k\}; \\ 1 - a_k & , \quad j = k; \\ -1 & , \quad j = k+2. \end{cases} \end{aligned}$$

Concluímos (9.2); de maneira análoga, podemos provar (9.3). $\diamond$

Teorema 9.2 *Dado o Algoritmo 9.1, então*

$$K_i(A, R_0) = [R_0, \ldots, R_{i-1}] = [P_0, \ldots, P_{i-1}]; \tag{9.4}$$

$$K_i(A, \tilde{R}_0) = [\tilde{R}_0, \ldots, \tilde{R}_{i-1}] = [\tilde{P}_0, \ldots, \tilde{P}_{i-1}]. \tag{9.5}$$

Prova

Vamos provar (9.4) usando indução matemática. Começaremos pela igualdade $[R_0, \ldots, R_{i-1}] = [P_0, \ldots, P_{i-1}]$.

1) Pelo passo 3 do Algoritmo 9.1, temos que $[R_0] = [P_0]$.

2) Supondo que

$$[R_0, \ldots, R_{k-1}] = [P_0, \ldots, P_{k-1}],$$

vamos provar que

$$[R_0, \ldots, R_k] = [P_0, \ldots, P_k].$$

Do passo 7, segue que

$$R_k = R_{k-1} - AP_{k-1}\alpha_{k-1}.$$

Mas, pela hipótese da indução, temos que $R_s = \sum_{j=0}^{s} b_{sj}P_j$ para $s = 0, \ldots, k-1$, e pelo Teorema 9.1, $AP_{k-1}\alpha_{k-1} = \sum_{j=0}^{k} a_j R_j$, com $a_j, b_j \in \mathbb{C}$. Assim,

$$\begin{aligned} R_k &= \sum_{j=0}^{k-1} b_{kj}P_j - \sum_{j=0}^{k} a_j R_j \\ &= \sum_{j=0}^{k-1} b_{kj}P_j - \sum_{j=0}^{k} a_j \sum_{t=0}^{j-1} b_{jt}P_t \\ &= \sum_{j=0}^{k} c_j P_j. \end{aligned}$$

Então,

$$R_k \in [P_0, \ldots, P_k] \Rightarrow [R_0, \ldots, R_k] \subset [P_0, \ldots, P_k]. \tag{9.6}$$

Pelo passo 12,

$$\begin{aligned} P_k &= R_k - P_{k-1}\beta_{k-1}, \text{ mas, pela hipótese da indução, } P_{k-1} = \sum_{j=0}^{k-1} a_j R_j \\ &= \sum_{j=0}^{k} c_j R_j. \end{aligned}$$

Segue que

$$P_k \in [R_0, \ldots, R_k] \Rightarrow [P_0, \ldots, P_k] \subset [R_0, \ldots, R_k] \tag{9.7}$$

De (9.6) e (9.7), temos que

$$[R_0, \ldots, R_k] = [P_0, \ldots, P_k]. \tag{9.8}$$

Para concluir, provaremos que $K_i(A, R_0) = [R_0, \ldots, R_{i-1}]$.

1)Pelo passo 3 do Algoritmo 9.1, temos que $[R_0] = [P_0]$.

2) Suponhamos que a igualdade $K_k(A, R_0) = [R_0, \ldots, R_{k-1}]$ se verifica. Vamos provar que $K_{k+1}(A, R_0) = [R_0, \ldots, R_k]$.

$$\begin{aligned} A^k R_0 &= A(A^{k-1}R_0), \text{ da hipótese da indução, } A^{k-1}R_0 = \sum_{j=0}^{k-1} b_j P_j \\ &= A\Big(\sum_{j=0}^{k-1} b_j P_j\Big) \\ &= \sum_{j=0}^{k-1} b_j A P_j. \end{aligned}$$

Do passo 7, $AP_j = (R_j - R_{j-1})\alpha_j^{-1}$,

$$\begin{aligned} A^k R_0 &= \sum_{j=0}^{k-1} b_j (R_j - R_{j-1})\alpha_j^{-1} \\ &= \sum_{j=0}^{k} c_j R_j. \end{aligned}$$

Então,

$$A^k R_0 \in [R_0, \ldots, R_k] \Rightarrow K_{k+1}(A, R_0) \subset [R_0, \ldots, R_k]. \tag{9.9}$$

Do passo 7, vem

$$\begin{aligned} R_k &= R_{k-1} - AP_{k-1}\alpha_{k-1} \\ &= R_{k-1} - A(R_{k-1} - P_{k-2}\beta_{k-2})\alpha_{k-1} \\ &= R_{k-1} - AR_{k-1}\alpha_{k-1} + P_{k-2}\beta_{k-2}\alpha_{k-1}. \end{aligned}$$

De (9.8), $R_{k-1} = \sum_{j=0}^{k-1} b_j P_j$,

$$R_k = \sum_{j=0}^{k-1} b_j P_j - A\sum_{j=0}^{k-1} b_j P_j + P_{k-2}\beta_{k-2}\alpha_{k-1}.$$

Da hipótese da indução, segue que $P_j = \sum_{s=0}^{j} c_j A^s R_0$, para $j = 1, \ldots, k-1$, então

$$\begin{aligned} R_k &= \sum_{j=0}^{k-1} b_j \sum_{s=0}^{j} c_j A^s R_0 - A\sum_{j=0}^{k-1} b_j \sum_{s=0}^{j} c_j A^s R_0 + \sum_{s=0}^{j} c_j A^{k-2} R_0 \beta_{k-2}\alpha_{k-1} \\ &= \sum_{j=0}^{k} d_j A^j R_0. \end{aligned}$$

Assim,

$$R_k \in K_{k+1}(A, R_0) \Rightarrow [R_0, \ldots, R_k] \subset K_{k+1}(A, R_0). \tag{9.10}$$

Das equações (9.9) e (9.10), segue que

$$K_{k+1}(A, R_0) = [R_0, \ldots, R_k]. \tag{9.11}$$

Das igualdades (9.8) e (9.11), finalizamos as duas induções, verificando assim, a igualdade (9.4); de maneira análoga, provaríamos (9.5). $\diamond$

Teorema 9.3 *Dado o Algoritmo 9.1, as seguintes relações se verificam*

(a) $\tilde{R}_i^H R_j = 0$ *e* $\tilde{P}_i A P_j = 0$, *para* $j < i$;

(b) $R_i^H \tilde{R}_j = 0$ *e* $P_i^H A^H \tilde{P}_j = 0$, *para* $j < i$.

Prova

Provaremos o item (a) usando indução matemática.

1) Suponhamos que $i = 1$ e $j = 0$ do Algoritmo 9.1, temos

$$\begin{aligned}
\tilde{R}_1^H R_0 &= (\tilde{R}_0 - A^H \tilde{P}_0 \tilde{\alpha}_0)^H R_0 \\
&= \tilde{R}_0^H R_0 - \tilde{\alpha}_0^H \tilde{P}_0^H A R_0 \\
&= \tilde{R}_0^H R_0 - [(P_0^H A^H \tilde{P}_0)^{-1} R_0^H \tilde{R}_0]^H \tilde{P}_0^H A R_0 \\
&= \tilde{R}_0^H R_0 - \tilde{R}_0^H R_0 (\tilde{P}_0^H A P_0)^{-1} \tilde{P}_0^H A R_0, \text{ mas } P_0 = R_0 \\
&= 0
\end{aligned}$$

e

$$\begin{aligned}
\tilde{P}_1^H A P_0 &= (\tilde{R}_1 + \tilde{P}_0 \tilde{\beta}_0)^H A P_0 \\
&= \tilde{R}_1^H A P_0 + \tilde{\beta}_0^H \tilde{P}_0^H A P_0 \\
&= \tilde{R}_1^H (R_0 - R_1) \alpha_0^{-1} + \tilde{\beta}_0^H \tilde{P}_0^H A P_0 \\
&= \tilde{R}_1^H R_0 \alpha_0^{-1} - \tilde{R}_1^H R_1 \alpha_0^{-1} + \tilde{\beta}_0^H \tilde{P}_0^H A P_0, \text{ mas } \tilde{R}_1^H R_0 = 0 \\
&= -\tilde{R}_1^H R_1 \alpha_0^{-1} + \tilde{\beta}_0^H \tilde{P}_0^H A P_0 \\
&= -\tilde{R}_1^H R_1 [(\tilde{P}_0^H A P_0)^{-1} \tilde{R}_0^H R_0]^{-1} + [(R_0^H \tilde{R}_0)^{-1} R_1^H \tilde{R}_1]^H \tilde{P}_0^H A P_0 \\
&= -\tilde{R}_1^H R_1 (\tilde{R}_0^H R_0)^{-1} \tilde{P}_0^H A P_0 + \tilde{R}_1^H R_1 (\tilde{R}_0^H R_0)^{-1} \tilde{P}_0^H A P_0 \\
&= 0.
\end{aligned}$$

2) Suponhamos que as igualdades, $\tilde{R}_k^H R_j = 0$ e $\tilde{P}_k^H A P_j = 0$, para $j < k$, se verificam. Vamos provar que $\tilde{R}_{k+1}^H R_j = 0$ e $\tilde{P}_{k+1}^H A P_j = 0$.

$$\begin{aligned}
\tilde{R}_{k+1}^H R_j &= (\tilde{R}_k - A^H \tilde{P}_k \tilde{\alpha}_k)^H R_j \\
&= \tilde{R}_k^H R_j - \tilde{\alpha}_k^H \tilde{P}_k^H A R_j, \ \text{, mas } \tilde{R}_k^H R_j = 0 \\
&= -\tilde{\alpha}_k^H \tilde{P}_k^H A R_j \\
&= -\tilde{\alpha}_k^H \tilde{P}_k^H A (P_j - P_{j-1}\beta_{j-1}) \\
&= -\tilde{\alpha}_k^H \tilde{P}_k^H A P_j + \tilde{\alpha}_k^H \tilde{P}_k^H A P_{j-1}\beta_{j-1}.
\end{aligned}$$

Da hipótese da indução, temos que $\tilde{P}_k^H A P_j = 0$ e $\tilde{P}_k^H A P_{j-1} = 0$. Então, $\tilde{R}_{k+1}^H R_j = 0$.

$$\begin{aligned}
\tilde{P}_{k+1}^H A P_j &= (\tilde{R}_{k+1} + \tilde{P}_k \tilde{\beta}_k)^H A P_j \\
&= \tilde{R}_{k+1}^H A P_j + \tilde{\beta}_k^H \tilde{P}_k^H A P_j, \ \text{, mas } \tilde{P}_k^H A P_j = 0 \\
&= \tilde{R}_{k+1}^H A P_j \\
&= \tilde{R}_{k+1}^H (R_{j+1} - R_j)\alpha_j^{-1} \\
&= \tilde{R}_{k+1}^H R_{j+1}\alpha_j^{-1} - \tilde{R}_{k+1}^H R_j \alpha_j^{-1} \\
&= 0.
\end{aligned}$$

Observamos que $\tilde{R}_{k+1}^H R_k = 0$ e $\tilde{P}_{k+1}^H A P_k = 0$, pois ambas cumprem a hipótese da indução matemática. Logo, podemos concluir que o item (a) se verifica. Provaríamos o item (b), analogamente. $\diamond$

Teorema 9.4 *Dado o Algoritmo 9.1, então, as colunas de R_i são ortogonais a $K_i(A^H, \tilde{R}_0)$. Além disso, se o algoritmo não termina antes de $t = \lceil n/s \rceil$ passos, $R_t = 0$. Analogamente, temos que as colunas de $\tilde{R}_k$ são ortogonais a $K_i(A, R_0)$, e $\tilde{R}_t = 0$.*

Prova [20].

Vamos definir um *polinômio de valores matriciais.*

Definição 9.2 *O polinômio de valores matriciais $\mathcal{P}$, de grau i, é dado por um polinônio da forma*

$$\mathcal{P}(t) = \sum_{j=0}^{i} t^j \Omega_j,$$

onde $\Omega_j \in \mathbb{C}^{s \times s}$ e $t \in \mathbb{C}$.

Definição 9.3 *Sejam $\mathcal{P}$ um polinômio de valores matriciais, as matrizes $A \in \mathbb{C}^{n\times s}$, $Y \in \mathbb{C}^{n\times s}$ e $\Theta \in \mathbb{C}^{s\times s}$, podemos definir as seguintes operações:*

(a) Produto $\circ$ [31]

$$\mathcal{P}(A) \circ Y = \sum_{j=0}^{i} A^j Y \Omega_j;$$

(b) Polinômio matricial $\mathcal{P}\Theta$

$$(\mathcal{P}\Theta)(t) = \sum_{j=0}^{i} t^j \Omega_j \Theta.$$

Teorema 9.5 *Sejam $\mathcal{P}$ e $\mathcal{Q}$ polinômios de valores matriciais, as matrizes $A \in \mathbb{C}^{n\times s}$, $Y \in \mathbb{C}^{n\times s}$, $\Theta \in \mathbb{C}^{s\times s}$ e $\Omega_j \in \mathbb{C}^{s\times s}$. Temos que*

(1) $(\mathcal{P}(A) \circ Y)\Theta = (\mathcal{P}\Theta)(A) \circ Y$;

(2) $(\mathcal{P} + \mathcal{Q})(A) \circ Y = \mathcal{P}(A) \circ Y + \mathcal{Q}(A) \circ Y$.

Prova [7].

Teorema 9.6 *Dado o Algoritmo 9.1, então existem os polinômios, $\mathcal{G}_i$ e $\mathcal{F}_i$, de grau i, tais que*

$$R_i = \mathcal{G}_i(A) \circ R_0, \text{ com } \mathcal{G}_0(A) = I \tag{9.12}$$

$$P_i = \mathcal{F}_i(A) \circ R_0, \text{ com } \mathcal{F}_0(A) = I, \tag{9.13}$$

onde

$$\mathcal{G}_{i+1}(t) = \mathcal{G}_i(t) - t\mathcal{F}_i(t)\alpha_i \tag{9.14}$$

$$\mathcal{F}_{i+1}(t) = \mathcal{G}_{i+1}(t) + \mathcal{F}_i(t)\beta_i, \tag{9.15}$$

e, α_i e βi são dados pelos passos 5 e 10 do Algoritmo 9.1.

Prova

Faremos a demonstração usando indução matemática, para demonstrar as equações (9.12) e (9.13).

1) Para $i = 0$, pelo passo 3 do Algoritmo 9.1,temos

$$R_0 = \mathcal{G}_0(A) \circ R_0$$

$$P_0 = R_0 = \mathcal{F}_0(A) \circ R_0.$$

2) Vamos supor que as igualdades abaixo se verificam:

$$\begin{aligned} R_k &= \mathcal{G}_k(A) \circ R_0 \\ P_k &= \mathcal{F}_k(A) \circ R_0. \end{aligned}$$

Queremos provar que

$$\begin{aligned} R_{k+1} &= \mathcal{G}_{k+1}(A) \circ R_0 \\ P_{k+1} &= \mathcal{F}_{k+1}(A) \circ R_0. \end{aligned}$$

Dos passo 7 e 12, e do Teorema (9.5), segue que

$$\begin{aligned} R_{k+1} &= R_k - AP_k\alpha_k \\ &= \mathcal{G}_k(A) \circ R_0 - A(\mathcal{F}_k(A) \circ R_0)\alpha_k \\ &= \mathcal{G}_k(A) \circ R_0 - (A\mathcal{F}_k(A)\alpha_k) \circ R_0 \\ &= [\mathcal{G}_k - A\mathcal{F}_k\alpha_k](A) \circ R_0 \\ &= \mathcal{G}_{k+1}(A)R_0 \\ P_{k+1} &= R_{k+1} + P_k\beta_k \\ &= \mathcal{G}_{k+1} \circ R_0 + (\mathcal{F}_k(A) \circ R_0)\beta_k \\ &= \mathcal{G}_{k+1} \circ R_0 + (\mathcal{F}_k(A)\beta_k) \circ R_0 \\ &= [\mathcal{G}_{k+1} + \mathcal{F}_k\beta_k](A) \circ R_0) \\ &= \mathcal{F}_{k+1}(A) \circ R_0. \qquad \diamond \end{aligned}$$

Proposição 9.1 *Dado o Algoritmo 9.1. Então, existe uma projeção Q, que projeta o erro E sobre o subespaço de Krylov em bloco, $K = K(A, R_0)$. A matriz que representa este operador é dada por*

$$Q = V(W^H AV)^{-1}W^H A,$$

onde V e W são bases de $K = K(A, R_0)$ e $L = K(A^H, \tilde{R}_0)$, respectivamente.

Prova

Observe que os passos 6 e 7 podem ser reescritos como

$$\hat{X} = X + \Delta \quad \text{e} \tag{9.16}$$

$$\hat{R} = R - A\Delta. \tag{9.17}$$

Com $\Delta \in K(A, R_0)$, tomamos $\Delta = VY$.

Por hipótese, o resíduo é ortogonal ao subespaço, $L = K(A^H, \tilde{R}_0)$. Assim,

$$W^H \hat{R} = W^H R - W^H AVY. \tag{9.18}$$

Mas, $W^H \hat{R} = 0$, então,

$$Y = (W^H AV)^{-1} W^H R. \tag{9.19}$$

Sabemos que o resíduo R é dado por $R = AE$, onde E é o erro. Então, temos que

$$Y = (W^H AV)^{-1} W^H AE \tag{9.20}$$

e

$$\Delta = V(W^H AV)^{-1} W^H AE. \tag{9.21}$$

Vamos considerar $Q = V(W^H AV)^{-1} W^H A$. Podemos verificar que $Q^2 = Q$, ou seja, Q é uma projeção que projeta o erro E sobre o subespaço $K = K(A, R_0)$. $\diamond$

9.3 Método do Gradiente Bi-Conjugado Estabilizado em Bloco (Bl-BiCGStab)

O método do Gradiente Bi-Conjugado Estabilizado em bloco [7] é uma generalização do BiCGStab [17]. O Bl-BiCGStab [7] difere do Bl-BiCG [20], entre outras razões, por não calcular a matriz conjugada do sistema (9.1).

Definição 9.4 *Sejam as matrizes $X, Y \in \mathbb{C}^{n\times s}$, podemos definir o seguinte produto interno*

$$\langle X, Y \rangle_F = tr(X^H Y),$$

onde, $Z \in \mathbb{C}^{n\times n}$, $tr(Z) = \sum_{j=1}^{n} z_{ii}$. A norma associada a este produto interno é a norma de Frobenius*, denotada por, $\| \cdot \|_F$.*

Segue o algoritmo do Bl-BiCGStab [7].

Algoritmo 9.2. Algoritmo Bl-BiCGStab (2003)

1. Seja $X_0 \in \mathbb{C}^{n\times s}$. Compute $R_0 = B - AX_0$ matriz de posto s
2. Escolha $\tilde{R}_0 \in \mathbb{C}^{n\times s}$ matriz de posto s
3. $P_0 = R_0$, $\tilde{P}_0 = \tilde{R}_0$
4. Para $i = 0, 1, \ldots$ **Do**
5. $\quad V_i = AP_i$
6. $\quad$ Resolva $(\tilde{R}_0^H V_i)\alpha_i = \tilde{R}_0^H R_i$
7. $\quad S_i = R_i - V_i\alpha_i$
8. $\quad T_i = AS_i$
9. $\quad \omega_i = \langle T_i, S_i\rangle_F / \langle T_i, T_i\rangle_F$
10. $\quad X_{i+1} = X_i + P_i\alpha_i + \omega_i S_i$
11. $\quad R_{i+1} = S_i - \omega_i T_i$
12. $\quad$ Resolva $(\tilde{R}_0^H V_i)\beta_i = -\tilde{R}_0^H T_i$
13. $\quad P_{i+1} = R_{i+1} + (P_i - \omega_i V_i)\beta_i$
14. **EndDo**

Teorema 9.7 *Dado o Algoritmo 9.2, então,*

$$R_i = (\mathcal{Q}_i\mathcal{G}_i)(A) \circ R_0 \tag{9.22}$$

$$P_i = (\mathcal{Q}_i\mathcal{F}_i)(A) \circ R_0, \tag{9.23}$$

onde $\mathcal{G}_i$ e $\mathcal{F}_i$ são dadas pelas equações (9.14) e (9.15), respectivamente, e

$$\mathcal{Q}_{i+1}(t) = (1 - \omega_i t)\mathcal{Q}_i(t),$$

e $\mathcal{Q}_0(t) = I_s$, $I \in \mathbb{C}^{s\times s}$, matriz identidade, ω_i dado pelo passo 9 do Algoritmo.

Prova

Vamos provar a igualdade (9.22) por indução matemática.

1) Dos passos 11 e 13 do Algoritmo 9.2, segue que

$$\begin{aligned}
R_1 &= S_0 - \omega_0 T_0 \\
&= R_0 - V_0\alpha_0 - \omega_0 A S_0 \\
&= R_0 - AP_0\alpha_0 - \omega_0 A(R_0 - V_0\alpha_0) \\
&= R_0 - AP_0\alpha_0 - \omega_0 AR_0 + \omega_0 A^2 P_0\alpha_0 \\
&= (I - \omega_0 A)R_0 - (I - \omega_0 A)AP_0\alpha_0, \text{mas } \ P_0 = R_0 \\
&= (I - \omega_0 A)(I - A)R_0\alpha_0 \\
&= [(I - \omega_0 A)(I - A\alpha_0)] \circ R_0 \\
&= (\mathcal{Q}_1\mathcal{G}_1)(A) \circ R_0 \qquad (9.24) \\
P_1 &= R_1 + (P_0 - \omega_0 V_0)\beta_0 \\
&= R_1 + (P_0 - \omega_0 AP_0)\beta_0, \text{de } () \\
&= (\mathcal{Q}_1\mathcal{G}_1)(A) \circ R_0 + (I - \omega_0 A)P_0\beta_0 \\
&= (\mathcal{Q}_1\mathcal{G}_1)(A) \circ R_0 + (I - \omega_0 A)(\mathcal{F}_0(A) \circ R_0)\beta_0 \\
&= (\mathcal{Q}_1\mathcal{G}_1)(A) \circ R_0 + (\mathcal{Q}_1\mathcal{F}_0\beta_0) \circ R_0 \\
&= [\mathcal{Q}_1(\mathcal{G}_1 + \mathcal{F}_0\beta_0](A) \circ R_0 \\
&= (\mathcal{Q}_1\mathcal{F}_1)(A) \circ R_0. \qquad (9.25)
\end{aligned}$$

2) Suponhamos que as igualdades

$$\begin{aligned}
R_k &= (\mathcal{Q}_k\mathcal{G}_k)(A) \circ R_0 \\
P_k &= (\mathcal{Q}_k\mathcal{F}_k)(A) \circ R_0
\end{aligned}$$

se verificam. Vamos provar que

$$\begin{aligned}
R_{k+1} &= (\mathcal{Q}_{k+1}\mathcal{G}_{k+1})(A) \circ R_0 \\
P_{k+1} &= (\mathcal{Q}_{k+1}\mathcal{F}_{k+1})(A) \circ R_0.
\end{aligned}$$

Do passos 11 e 13 do Algoritmo 9.2, temos que

$$
\begin{aligned}
R_{k+1} &= S_k - \omega_k T_k \\
&= (1 - \omega_k A)(R_k - AP_k\alpha_0), \text{ aplicando a hipótese da indução,} \\
&= (1 - \omega_k A)[(\mathcal{Q}_k\mathcal{G}_k)(A) \circ R_0 + A(\mathcal{Q}_k\mathcal{F}_k)(A) \circ R_0] \\
&= (1 - \omega_k A)(\mathcal{Q}_k(A))[\mathcal{G}_k(A) \circ R_0 + A\mathcal{F}_k(A) \circ R_0] \\
&= [\mathcal{Q}_{k+1}(A)\mathcal{G}_{k+1}(A)] \circ R_0 \\
&= (\mathcal{Q}_{k+1}\mathcal{G}_{k+1})(A) \circ R_0 \qquad (9.26) \\
P_{k+1} &= R_{k+1} + (P_k - \omega_k V_k)\beta_k \\
&= R_{k+1} + (P_k - \omega_k AP_k)\beta_k \\
&= R_{k+1} + (I - \omega_k A)P_k\beta_k, \text{ de ()} \\
&= (\mathcal{Q}_{k+1}\mathcal{G}_{k+1})(A) \circ R_0 + (I - \omega_k A)(\mathcal{Q}_k\mathcal{F}_k)(A) \circ R_0\mathcal{F}_k \\
&= (\mathcal{Q}_{k+1}\mathcal{G}_{k+1})(A) \circ R_0 + [(I - \omega_k A)\mathcal{Q}_k(\mathcal{F}_k\mathcal{F}_k)](A) \circ R_0 \\
&= (\mathcal{Q}_{k+1}\mathcal{G}_{k+1})(A) \circ R_0 + [(\mathcal{Q}_{k+1})(\mathcal{F}_k\beta_k)](A) \circ R_0 \\
&= [\mathcal{Q}_{k+1}(\mathcal{G}_{k+1} + \mathcal{F}_k\beta_k)](A) \circ R_0 \\
&= (\mathcal{Q}_{k+1}\mathcal{F}_{k+1})(A) \circ R_0. \qquad \diamond \qquad (9.27)
\end{aligned}
$$

Capítulo 10

Resultados Numéricos

10.1 Introdução

Neste capítulo, vamos verificar a viabilidade do método estudado neste trabalho, bem como sua comparação com outros métodos da literatura, são eles: Bl-GMRES [21], BiCGStab-S e GMRES-S, os dois últimos serão definidos na seção 10.2.2. Implementaremos o código desenvolvido em Matlab do Bl-BiCGStab [7] com precondicionador. Optamos por gerar tabelas a partir dos resultados numéricos gerados. Analisando as tabelas vamos concluir que o Bl-BiCGStab [7] com precondicionador se mostrou eficiente na resolução de sistemas lineares (1.1) e equivalente ao métodos BiCGStab-S e GMRES-S.

10.2 Tabelas

Apresentaremos alguns resultados da implementação do código desenvolvido, Bl-BiCGStab [7] com precondicionador, considerando o sistema (1.1). As matrizes principais dos sistemas, A, foram obtidas de [12] . Para o número de lados direitos s, escolhemos os seguintes valores: $4, 8, 12, 16$ e 20. Usamos a tolerância de 10^{-6} para a norma do resíduo relativo. O número máximo de iterações escolhido foi 20. Para cada sistema, a solução exata é conhecida, ou seja, não foi determinada a partir de nenhum outro método iterativo. Assim, para cada experimento, foi obtido o erro em relação à solução exata, o que é bastante representativo, pois, em geral, utiliza-se apenas a limitação da norma do resíduo relativo.

10.2.1 Resultados da aplicação do Bl-BiCGStab com precondicionadores sobre matrizes de teste [12]

As tabelas que seguem foram obtidas, aplicando-se o algoritmo desenvolvido nesta tese para um conjunto de matrizes selecionadas de [12]. Para cada matriz, foi gerada uma tabela. A cada valor de s, $s = 4, 8, 12, 16$ e 20, variando os tipos de precondicionadores, geramos sete linhas de dados. Iremos adotar como precondicionadores a decomposição LU da matriz A, ou seja, o produto de uma matriz L por uma matriz unitária triangular superior, U. Por sua vez, L é o produto de uma matriz triangular inferior por uma matriz de permutação. Usamos a função "luinc" do MATLAB e variamos o grau de tolerância da esparsidade das matrizes L e U. Adotaremos a nomenclatura a seguir:

* LU $\rightarrow$ decomposição LU com tolerância 0, ou seja, LU completa;
* LU-6 $\rightarrow$ decomposição LU com tolerância 10^{-6};
* LU-4 $\rightarrow$ decomposição LU com tolerância 10^{-4};
* LU-2 $\rightarrow$ decomposição LU com tolerância 10^{-2};
* DiagOnes $\rightarrow$ matriz diagonal sendo seus elementos os da diagonal principal da matriz A, substituindo valores 0 por 1;
* DiagSum $\rightarrow$ matriz diagonal sendo seus elementos os da diagonal principal da matriz A, substituindo valores 0 pela soma dos valores absolutos dos elementos desta linha de A;
* None $\rightarrow$ nenhum precondicionador.

Para número máximo de iterações, ficou estabelecido 20. A coluna "Erro" representa o erro obtido entre as soluções exata e a gerada pelo Bl-BiCGStab [7]. Considerando X^* a solução exata e X a solução gerada pelo Bl-BiCGStab [7], o erro E será dado por $\frac{\|X^*-X\|_2}{\|X^*\|_2}$. A norma relativa do resíduo é dada por $\frac{\|R\|_2}{\|R_0\|_2}$, onde $R = B - AX$ e $R_0 = B - AX_0$; X, solução obtida pelo Bl-BiCGStab [7] e X_0, solução inicial do sistema (1.1). Na coluna "Flag", cada número representa uma informação a respeito do processo iterativo, são elas:

* 0 → Bl-BiCGStab converge para a tolerância desejada até o número de iterações definido;

* 1 → Bl-BiCGStab não converge até o número de iterações definido;

* 2 → um dos precondicionadores, $M1$ ou $M2$, é mal condicionado;

* 3 → uma das matrizes usadas para cálculos intermediários do algoritmo não é inversível.

As tabelas geradas a partir dos dados obtidos pela implementação do Bl-BiCGStab são as tabelas 10.1 a 10.11. A partir destas, podemos observar que, excetuando a matriz Watt 2 [12], o Bl-BiCGStab deve ser implementado usando um precondicionador, e que os melhores preconcionadores foram os obtidos a partir da decomposição LU da matriz A.

Tabela 10.1: Sherman4 - Matriz real, não-simétrica, de ordem 1104, com 3786 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	3.31e-015	1	7e-031	0
	LU-6	2.96e-006	1	1.26e-009	0
	LU-4	0.00219	1	8.46e-007	0
	LU-2	0.368	20	0.00012	1
	DiagOnes	0.658	20	0.000759	1
	DiagSum	0.664	20	0.000431	1
	None	1.12	20	0.136	1
8	LU	2.08e-015	1	7.8e-031	0
	LU-6	3.12e-006	1	1.28e-009	0
	LU-4	0.00187	1	7e-007	0
	LU-2	0.0739	3	9.46e-007	0
	DiagOnes	5.94e+042	20	7.05e+041	1
	DiagSum	1.59e+016	20	1.32e+015	1
	None	–	–	–	2
12	LU	2.67e-015	1	6.28e-030	0
	LU-6	3.05e-006	1	9.76e-010	0
	LU-4	0.00222	1	6.47e-007	0
	LU-2	0.00396	1	9.85e-007	0
	DiagOnes	2.26e+032	20	2.66e+030	1
	DiagSum	1.63e+118	20	6.39e+116	1
	None	–	–	–	2
16	LU	2.47e-015	1	1.23e-030	0
	LU-6	3.23e-006	1	1.35e-009	0
	LU-4	0.00219	1	8.41e-007	0
	LU-2	–	–	–	2
	DiagOnes	–	–	–	2
	DiagSum	1.06e+053	20	5.17e+051	1
	None	3.75e+076	20	1.13e+076	1
20	LU	3.81e-015	1	2.34e-030	0
	LU-6	3.13e-006	1	1.35e-009	0
	LU-4	0.000908	1	3.62e-007	0
	LU-2	0.00271	1	9.91e-007	0
	DiagOnes	–	–	–	–
	DiagSum	7.16e+008	20	6.44e+007	1
	None	1.15e+042	20	2.95e+041	1

Tabela 10.2: Pde900 - Matriz real, não-simétrica, de ordem 900, com 4380 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	2.89e-016	1	1.3e-030	0
	LU-6	1.66e-009	1	4.92e-012	0
	LU-4	6.97e-006	1	1.77e-008	0
	LU-2	0.000253	1	5.97e-007	0
	DiagOnes	2.47e+037	20	3.61e+035	1
	DiagSum	3.03e+067	20	1.87e+065	1
	None	1.15	20	0.0383	1
8	LU	3.01e-016	1	2.07e-031	0
	LU-6	4.02e-010	1	2.52e-012	0
	LU-4	3.76e-005	1	5.94e-008	0
	LU-2	0.000439	1	9.48e-007	0
	DiagOnes	9.46e+014	20	8.8e+012	1
	DiagSum	8.85e+038	20	2.89e+037	1
	None	4.7e+042	20	2.09e+041	1
12	LU	3e-016	1	4.58e-031	0
	LU-6	2.6e-009	1	4.47e-012	0
	LU-4	2.76e-005	1	2.84e-008	0
	LU-2	0.000748	1	8.1e-007	0
	DiagOnes	3.02e+115	20	1.09e+114	1
	DiagSum	3.38e+087	20	8.51e+085	1
	None	6.09e+060	20	2.13e+059	1
16	LU	3.17e-016	1	9.18e-032	0
	LU-6	4.53e-010	1	1.37e-012	0
	LU-4	2.03e-005	1	7.04e-008	0
	LU-2	0.000665	1	9.49e-007	0
	DiagOnes	1.05e+012	20	2.55e+010	1
	DiagSum	6.69e+037	20	1.58e+036	1
	None	–	–	–	2
20	LU	3.98e-016	1	1.99e-031	0
	LU-6	8.05e-009	1	5.16e-011	0
	LU-4	1.71e-005	1	5.15e-008	0
	LU-2	0.000583	1	7.32e-007	0
	DiagOnes	–	–	–	2
	DiagSum	4.61e+117	20	1.17e+116	1
	None	–	–	–	2

Tabela 10.3: Tols4000 - Matriz real, não-simétrica, de ordem 4000, com 8784 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	7.14e-017	1	2.01e-031	0
	LU-6	5.42e-017	1	4.21e-033	0
	LU-4	2.96e-009	1	9.43e-013	0
	LU-2	0.000132	1	3.86e-008	0
	DiagOnes	4.43e+046	20	1.34e+042	1
	DiagSum	2.02	20	1.81	1
	None	7.57e+047	20	6.18e+044	1
8	LU	6.96e-017	1	9.58e-030	0
	LU-6	0.000118	1	4.45e-008	0
	LU-4	2.72e-009	1	9.3e-013	0
	LU-2	0.000118	1	4.45e-008	0
	DiagOnes	5.69e+068	20	1.29e+065	1
	DiagSum	2.23e+065	20	2.43e+065	1
	None	3.28e+064	20	1.94e+061	1
12	LU	4.38e-017	1	7.86e-033	0
	LU-6	6.84e-017	1	2.49e-031	0
	LU-4	2.19e-009	1	7.94e-013	0
	LU-2	0.000119	1	4.42e-008	0
	DiagOnes	3.25e+060	20	7.66e+056	1
	DiagSum	2.66e+014	20	2.33e+014	1
	None	3.07e+005	20	1.02	1
16	LU	6.33e-017	1	8.07e-033	0
	LU-6	6.36e-017	1	5.68e-032	0
	LU-4	2.6e-009	1	9.09e-013	0
	LU-2	0.000154	1	5.34e-008	0
	DiagOnes	1.51e+080	20	3.08e+076	1
	DiagSum	7.42	20	0.00184	1
	None	7.01e+003	20	5.67	1
20	LU	6.47e-017	1	1.04e-031	0
	LU-6	6.71e-017	1	2.03e-031	0
	LU-4	3.24e-009	1	1.03e-012	0
	LU-2	0.000145	1	5.35e-008	0
	DiagOnes	1.44e+006	20	184	1
	DiagSum	69.2	20	0.00905	1
	None	2.66e+004	20	14.7	1

Tabela 10.4: Dw2048 - Matriz real, não-simétrica, de ordem 2048, com 10114 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	4.01e-015	1	5.79e-032	0
	LU-6	6.47e-010	1	1.4e-013	0
	LU-4	6.45e-009	1	1.32e-012	0
	LU-2	0.00872	1	9.63e-007	0
	DiagOnes	4.3e+009	20	6.41e+006	1
	DiagSum	3.57	20	0.00159	1
	None	0.992	20	0.000571	1
8	LU	7.39e-015	1	1.84e-031	0
	LU-6	2.51e-009	1	3.54e-013	0
	LU-4	4.59e-009	1	7.2e-013	0
	LU-2	0.00731	1	7.55e-007	0
	DiagOnes	8.45	20	0.0109	1
	DiagSum	0.335	20	0.000144	1
	None	0.2	20	0.000112	1
12	LU	7.01e-015	1	4.36e-031	0
	LU-6	7.86e-009	1	9.49e-013	0
	LU-4	1.68e-009	1	4.09e-013	0
	LU-2	0.00319	1	5.9e-007	0
	DiagOnes	3.25	20	0.00496	1
	DiagSum	0.158	20	7.34e-005	1
	None	0.255	20	0.000255	1
16	LU	5.95e-015	1	1.33e-031	0
	LU-6	1.81e-009	1	3.38e-013	0
	LU-4	3.73e-009	1	7.63e-013	0
	LU-2	0.00642	1	8.62e-007	0
	DiagOnes	5.02	20	0.00653	1
	DiagSum	1.35	20	0.000595	1
	None	0.681	20	0.000629	1
20	LU	9.94e-015	1	1.49e-030	0
	LU-6	7.9e-009	1	1.05e-012	0
	LU-4	8.69e-009	1	1.37e-012	0
	LU-2	0.0074	1	9.21e-007	0
	DiagOnes	4.71	20	0.00437	1
	DiagSum	1.44	20	0.000705	1
	None	0.934	20	0.00075	1

Tabela 10.5: Watt 1 - Matriz real, não-simétrica. de ordem 1865, com 11360 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	3.13e-015	1	1.5e-032	0
	LU-6	0.000432	1	6.16e-015	0
	LU-4	0.235	1	1.43e-012	0
	LU-2	0.849	1	1.25e-011	0
	DiagOnes	0.925	1	6.01e-010	0
	DiagSum	1.14	1	3.13e-009	0
	None	–	–	–	2
8	LU	3.52e-015	1	9.12e-033	0
	LU-6	0.000419	1	1.95e-015	0
	LU-4	0.229	1	4.59e-013	0
	LU-2	0.86	1	4.04e-012	0
	DiagOnes	0.938	1	1.71e-010	0
	DiagSum	1.05	1	9.1e-010	0
	None	–	–	–	2
12	LU	2.62e-015	1	1.87e-032	0
	LU-6	0.000433	1	1.63e-015	0
	LU-4	0.237	1	3.82e-013	0
	LU-2	0.866	1	3.36e-012	0
	DiagOnes	0.943	1	1.15e-010	0
	DiagSum	1.03	1	5.78e-010	0
	None	7.99e+069	20	5.37e+067	1
16	LU	2.19e-015	1	1.67e-033	0
	LU-6	0.000422	1	1.71e-015	0
	LU-4	0.233	1	4.05e-013	0
	LU-2	0.875	1	3.55e-012	0
	DiagOnes	0.947	1	1.09e-010	0
	DiagSum	1.03	1	5.69e-010	0
	None	–	–	–	2
20	LU	3.9e-015	1	3.22e-032	0
	LU-6	0.000441	1	1.76e-015	0
	LU-4	0.24	1	4.12e-013	0
	LU-2	0.875	1	3.6e-012	0
	DiagOnes	0.948	1	1.03e-010	0
	DiagSum	1.01	1	5.26e-010	0
	None	–	–	–	2

Tabela 10.6: Watt 2 - Matriz real, não-simétrica, de ordem 1865, com 11550 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	0.905	2	9.91e-007	0
	LU-6	0.611	1	8.2e-011	0
	LU-4	0.685	1	5.42e-011	0
	LU-4	1.02	1	5.01e-010	0
	DiagOnes	0.919	5	3.85e-007	0
	DiagSum	35.6	6	4.84e-008	0
	None	0.979	2	1.19e-008	0
8	LU	0.918	3	5.44e-007	0
	LU-6	0.756	1	1.62e-010	0
	LU-4	0.786	1	1.82e-010	0
	LU-2	0.947	1	5.51e-010	0
	DiagOnes	0.929	7	2.38e-007	0
	DiagOnes	55.7	4	9.45e-007	0
	None	0.987	3	9.5e-010	0
12	LU	0.924	3	5.72e-007	0
	LU-6	0.788	1	1.11e-010	0
	LU-4	0.824	1	1.38e-010	0
	LU-2	0.948	1	3.31e-010	0
	DiagOnes	1.71	3	5.69e-008	0
	DiagSum	129	6	8.04e-007	0
	None	0.986	2	1.9e-007	0
16	LU	0.916	3	5.25e-007	0
	LU-6	0.659	1	1.06e-010	0
	LU-4	0.693	1	1.28e-010	0
	LU-2	0.962	1	3.31e-010	0
	DiagOnes	0.967	3	1.09e-007	0
	DiagSum	16	7	6.24e-008	0
	None	0.98	2	2.31e-008	0
20	LU	0.933	2	1.72e-007	0
	LU-6	0.674	1	1.16e-010	0
	LU-4	0.734	1	1.29e-010	0
	LU-2	1.22	1	9.96e-010	0
	DiagOnes	0.966	3	3.2e-007	0
	DiagSum	128	5	7.49e-007	0
	None	0.969	2	5.95e-007	0

Tabela 10.7: Cry2500 - Matriz real, não-simétrica, de ordem 2500, com 12349 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	1	1	1	3
	LU-6	0.528	2	8.21e-009	0
	LU-4	2.19	4	7.6e-007	0
	LU-2	1	1	1	3
	DiagOnes	1.96e+005	20	0.259	1
	DiagSum	6.75e+004	20	0.268	1
	None	2.55	20	0.287	1
8	LU	1	1	1	3
	LU-6	0.542	2	1.88e-008	0
	LU-4	3.05	4	3.82e-008	0
	LU-2	1	1	1	3
	DiagOnes	3.86e+004	20	0.0242	1
	DiagSum	4.27e+004	20	0.176	1
	None	1.09	20	0.103	1
12	LU	1	1	1	3
	LU-6	0.546	2	2.56e-008	0
	LU-4	2.72	4	1.86e-008	0
	LU-2	1	1	1	3
	DiagOnes	4.23e+004	20	0.0579	1
	DiagSum	4.45e+003	20	1.08	1
	None	1.19	20	0.122	1
16	LU	1	1	1	3
	LU-6	0.522	2	1.26e-008	0
	LU-4	2.35	3	9.46e-007	0
	LU-4	1	1	1	3
	DiagOnes	1.65e+004	20	0.0188	1
	DiagSum	1.85e+003	20	0.0359	1
	None	0.941	20	0.0206	1
20	LU	1	1	1	3
	LU-6	0.531	2	2.2e-008	0
	LU-4	3.42	4	3.94e-008	0
	LU-2	1	1	1	3
	DiagOnes	6.88e+004	20	0.0414	1
	DiagSum	38.5	20	0.031	1
	None	0.967	20	0.0374	1

Tabela 10.8: Pde2961 - Matriz real, não-simétrica, de ordem 2961, com 14585 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	0.0371	20	3.88e-006	1
	LU-6	7.13e-008	1	4.25e-011	0
	LU-4	0.000278	1	2.56e-008	0
	LU-2	0.000429	1	4.43e-008	0
	DiagOnes	2.14	20	0.00177	1
	DiagSum	8.91	20	0.00631	1
	None	1.2	20	0.00125	1
8	LU	0.00408	17	7.81e-007	0
	LU-6	1.82e-007	1	2.89e-011	0
	LU-4	0.000137	1	2.35e-008	0
	LU-2	0.000622	1	1.75e-007	0
	DiagOnes	1.78	20	0.00108	1
	DiagSum	1.23	20	0.000879	1
	None	3.56	20	0.00522	1
12	LU	0.00332	15	5.03e-007	0
	LU-6	3.82e-007	1	4.49e-011	0
	LU-4	0.000595	1	9.86e-008	0
	LU-2	0.000847	1	1.05e-007	0
	DiagOnes	2.43	20	0.00134	1
	DiagSum	2.33	20	0.00125	1
	None	3.2	20	0.00465	1
16	LU	0.0031	14	6.59e-007	0
	LU-6	2.91e-007	1	7.99e-011	0
	LU-4	4.88e-009	2	1.18e-012	0
	LU-2	0.000989	1	7.47e-007	0
	DiagOnes	10.5	20	0.00709	1
	DiagSum	6.86	20	0.00562	1
	None	5.14	20	0.00827	1
20	LU	0.000356	13	9.1e-008	0
	LU-6	8.2e-008	1	1.72e-011	0
	LU-4	0.000127	1	3.05e-008	0
	LU-2	0.000114	1	5.52e-008	0
	DiagOnes	16.1	20	0.0114	1
	DiagSum	58.3	20	0.0328	1
	None	9.63	20	0.0156	1

Tabela 10.9: Rdb3200l - Matriz real, não-simétrica, de ordem 3200, com 18880 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	29.5	20	0.114	1
	LU-6	1.63e-009	1	7.69e-012	0
	LU-4	8.99e-005	1	3.3e-007	0
	LU-2	0.000124	13	5.55e-007	0
	DiagOnes	47.7	20	0.211	1
	DiagSum	4.58	20	0.0223	1
	None	1.51	20	0.00993	1
8	LU	27.7	20	0.124	1
	LU-6	2e-009	1	7.03e-012	0
	LU-4	6.3e-005	1	3.13e-007	0
	LU-2	0.000148	10	6.72e-007	0
	DiagOnes	14.7	20	0.0979	1
	DiagSum	158	20	0.735	1
	None	1.49	20	0.0099	1
12	LU	21.2	20	0.0707	1
	LU-6	4.29e-009	1	2.97e-011	0
	LU-4	9.85e-005	1	3.64e-007	0
	LU-2	3.39e-005	8	1.01e-007	0
	DiagOnes	36.9	20	0.157	1
	DiagSum	20.5	20	0.0521	1
	None	23.7	20	0.118	1
16	LU	10.1	20	0.035	1
	LU-6	3.03e-008	1	2.72e-010	0
	LU-4	6.22e-005	1	3.21e-007	0
	LU-2	4.93e-006	9	2.87e-008	0
	DiagOnes	240	20	1.08	1
	DiagSum	264	20	0.734	1
	None	6.63	20	0.0533	1
20	LU	10.5	20	0.0266	1
	LU-6	3.7e-009	1	2.06e-011	0
	LU-4	7.88e-005	1	3.71e-007	0
	LU-2	5.15e-006	9	2.37e-008	0
	DiagOnes	12.1	20	0.0342	1
	DiagSum	21.5	20	0.046	1
	None	4.97	20	0.0185	1

Tabela 10.10: Fidap021 - Matriz real, não-simétrica, de ordem 656, com 18962 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	1	0	1	2
	LU-6	0.113	1	1.49e-011	0
	LU-4	26.2	1	6.61e-009	0
	LU-2	1	0	1	2
	DiagOnes	0.581	2	6.64e-007	0
	DiagSum	0.877	8	5.58e-007	0
	None	1.83	20	2.86e-005	1
8	LU	1	0	1	2
	LU-6	0.801	1	1.25e-011	0
	LU-4	46.8	1	3.62e-009	0
	LU-2	1	0	1	2
	DiagOnes	0.874	3	9.98e-007	0
	DiagSum	0.97	7	3.67e-007	0
	None	2.92	20	2.93e-005	1
12	LU	1	0	1	2
	LU-6	0.241	1	1.91e-011	0
	LU-4	53.8	1	4.7e-009	0
	LU-2	1	0	1	2
	DiagOnes	0.55	3	4.9e-007	0
	DiagSum	1.19	7	2.54e-007	0
	None	2.42	20	2.02e-005	1
16	LU	1	0	1	2
	LU-6	0.112	1	1.29e-011	0
	LU-4	33.7	1	4.3e-009	0
	LU-2	1	0	1	2
	DiagOnes	0.579	3	4.02e-007	0
	DiagSum	1.13	8	3.75e-007	0
	None	1.34	20	7.05e-006	1
20	LU	1	0	1	2
	LU-6	0.0918	1	1.45e-011	0
	LU-4	51.8	1	6.17e-009	0
	LU-2	1	0	1	2
	DiagOnes	0.526	4	1.86e-007	0
	DiagSum	0.956	6	4.38e-007	0
	None	36.6	20	0.000296	1

Tabela 10.11: Cavity05 - Matriz real, não-simétrica, de ordem 1182, com 32633 elementos não-nulos.

Número de lados direitos	Pré-Condicionadores	Erro	Iterações	Norma relativa do resíduo	Flag
4	LU	1	1	1	3
	LU-6	0.001	1	3.97e-010	0
	LU-4	0.0538	1	4.31e-008	0
	LU-2	1	0	1	2
	DiagOnes	4.28	20	3.43e-005	1
	DiagSum	1.93	20	4.89e-005	1
	None	0.621	20	0.000187	1
8	LU	1	1	1	3
	LU-6	0.000685	1	2.87e-010	0
	LU-4	0.0753	1	3.46e-008	0
	LU-2	1	0	1	2
	DiagOnes	6.94	20	5.99e-005	1
	DiagSum	2.38	20	2.07e-005	1
	None	0.538	20	6.47e-005	1
12	LU	1	1	1	3
	LU-6	0.000538	1	3.9e-010	0
	LU-4	0.0488	1	3.59e-008	0
	LU-2	1	0	1	2
	DiagOnes	12.6	20	0.000129	1
	DiagSum	12.6	20	0.000129	1
	None	7.39	20	0.00354	1
16	LU	1	1	1	3
	LU-6	0.000437	1	2.87e-010	0
	LU-4	0.0325	1	3.8e-008	0
	LU-2	1	0	1	2
	DiagOnes	2.1	20	9.36e-006	1
	DiagSum	662	20	0.00587	1
	None	3.22	20	0.000839	1
20	LU	1	1	1	3
	LU-6	0.000455	1	2.88e-010	0
	LU-4	0.0342	1	2.78e-008	0
	LU-2	1	0	1	2
	DiagOnes	7.87	20	3.22e-005	1
	DiagSum	2.15	20	1.94e-005	1
	None	0.575	20	0.000122	1

10.2.2 Resultado da comparação do Bl-BiCGStab com precondicionadores com outros métodos iterativos

Tomando o sistema linear (1.1), escolhemos para matriz A a matriz real Pde2961 [12], que é não simétrica, de ordem 2961 e com 14585 elementos não-nulos. Variamos o número de lados direitos, $s = 4, 8, 12, 16$ e 20. Para as tabelas 10.12 e 10.13, tomamos os precondicionadores LU-6 e LU-4, respectivamente. O objetivo destas tabelas é gerar dados para se comparar o Bl-BiCGStab com outros métodos da literatura abordada. O Bl-GMRES é o método GMRES em bloco desenvolvido por Vital [21], e implementado com o uso de precondicionadores. Confrontando estes dois métodos em bloco, temos que o Bl-BiCGStab é mais eficiente que o Bl-GMRES [21]. Isto fica claro observando as colunas de erro, número de iterações e norma do resíduo relativo. Os métodos BiCGStab-S e GMRES-S são os métodos BiCGStab [17] e GMRES [15] aplicados a cada um dos s lados direitos do sistema (1.1). Comparando o Bl-BiCGStab com o BiCGStab-S e GMRES-S, observamos que o método proposto apresenta as colunas erro, número de iterações e norma do resíduo relativo, com valores que nos permitem concluir que esse método é uma boa opção para resolução de sistemas lineares cuja matriz principal, A, seja Pde2961 [12].

As tabelas 10.14 e 10.15 foram geradas fazendo $A = Rdb3200l$. A matriz $Rdb3200l$ [12] é uma não simétrica, de ordem 3200, com 18880 elementos não-nulos. Comparando os métodos em bloco, Bl-BiCGStab com precondicionador e Bl-GMRES [21], temos que a performance do Bl-BiCGStab com precondicionador é a melhor. Observando o Bl-BiCGStab em relação aos métodos BiCGStab-S e GMRES-S, usando os precondicionadores LU-6, o Bl-BiCGStab com precondicionador é melhor que o GMRES-S e semelhante ao BiCGStab-S, em termos das colunas erro, número de iterações e norma relativa do resíduo. Já com os precondicionadores LU-4, o Bl-BiCGStab com precondicionador apresenta uma performance equivalente aos métodos BiCGStab-S e GMRES-S.

Considerando os resultados obtidos nas tabelas 10.12, 10.13, 10.14 e 10.15, as colunas Erro, Iterações e Norma relativa do resíduo, podemos concluir que o método Bl-BiCGStab com precondicionador se apresenta como uma opção para resolução de sistemas lineares de grande porte com múltiplos lados direitos.

Tabela 10.12: Matriz Pde2961 com precondicionador LU-6

Número de lados direitos	Método	Erro	Iterações	Norma relativa do resíduo	Flag
4	Bl-BiCGStab	1.59e-007	1	2.46e-011	0
	BiCGStab-S	4.03e-008	1	6.24e-012	0
	Bl-GMRES	0.00704	13	9.78e-007	0
	GMRES-S	2.83e-008	1	6.99e-011	0
8	Bl-BiCGStab	1.52e-007	1	1.41e-011	0
	BiCGStab-S	3.5e-008	1	6.28e-012	0
	Bl-GMRES	0.000729	15	9.99e-007	0
	GMRES-S	2.38e-008	1	7.18e-011	0
12	Bl-BiCGStab	1.55e-007	1	6.67e-011	0
	BiCGStab-S	3.7e-008	1	6.31e-012	0
	Bl-GMRES	0.00353	13	7.89e-007	0
	GMRES-S	2.55e-008	1	7.4e-011	0
16	Bl-BiCGStab	2.13e-007	1	2.81e-011	0
	BiCGStab-S	3.23e-008	1	6.38e-012	0
	Bl-GMRES	0.000782	13	9.96e-007	0
	GMRES-S	2.16e-008	1	7.82e-011	0
20	Bl-BiCGStab	6.72e-008	1	1.21e-011	0
	BiCGStab-S	3.44e-008	1	6.31e-012	0
	Bl-GMRES	0.000492	16	9.84e-007	0
	GMRES-S	2.33e-008	1	7.4e-011	0

Tabela 10.13: Matriz Pde2961 com precondicionador LU-4

Número de lados direitos	Método	Erro	Iterações	Norma relativa do resíduo	Flag
4	Bl-BiCGStab	0.000197	1	2.96e-008	0
	BiCGStab-S	0.000307	1	2.49e-008	0
	Bl-GMRES	0.00704	13	9.78e-007	0
	GMRES-S	0.000194	1	4.75e-007	0
8	Bl-BiCGStab	0.000142	1	6.92e-008	0
	BiCGStab-S	0.00031	1	2.77e-008	0
	Bl-GMRES	0.000729	15	9.99e-007	0
	GMRES-S	0.000161	1	4.83e-007	0
12	Bl-BiCGStab	0.000237	1	1.08e-007	0
	BiCGStab-S	0.000316	1	2.79e-008	0
	Bl-GMRES	0.00353	13	7.89e-007	0
	GMRES-S	0.000179	1	5.15e-007	0
16	Bl-BiCGStab	0.000116	1	3.67e-008	0
	BiCGStab-S	0.000315	1	3.21e-008	0
	Bl-GMRES	0.000782	13	9.96e-007	0
	GMRES-S	0.000147	1	5.29e-007	0
20	Bl-BiCGStab	0.000147	1	3.67e-008	0
	BiCGStab-S	0.000312	1	2.9e-008	0
	Bl-GMRES	0.000492	16	9.84e-007	0
	GMRES-S	0.000159	1	5.03e-007	0

Tabela 10.14: Matriz Rdb3200l com precondicionador LU-6

Número de lados direitos	Método	Erro	Iterações	Norma relativa do resíduo	Flag
4	Bl-BiCGStab	1.63e-009	1	7.69e-012	0
	BiCGStab-S	2.45e-009	1	5.73e-012	0
	Bl-GMRES	2.74	20	0.0288	1
	GMRES-S	0.00032	1	8.26e-007	0
8	Bl-BiCGStab	2e-009	1	7.03e-012	0
	BiCGStab-S	2.41e-009	1	6.5e-012	0
	Bl-GMRES	2.91	20	0.155	1
	GMRES-S	0.000244	1	8.27e-007	0
12	Bl-BiCGStab	4.29e-009	1	2.97e-011	0
	BiCGStab-S	2.46e-009	1	5.44e-012	0
	Bl-GMRES	2.87	20	0.242	0
	GMRES-S	0.000338	1	8.22e-007	0
16	Bl-BiCGStab	3.03e-008	1	2.72e-010	0
	BiCGStab-S	2.45e-009	1	6.56e-012	0
	Bl-GMRES	2.79	20	0.137	0
	GMRES-S	0.000255	1	8.28e-007	0
20	Bl-BiCGStab	3.7e-009	1	2.06e-011	0
	BiCGStab-S	2.43e-009	1	5.77e-012	0
	Bl-GMRES	2.87	20	0.241	0
	GMRES-S	0.000298	1	8.25e-007	0

Tabela 10.15: Matriz Rdb3200l com precondicionador LU-4

Número de lados direitos	Método	Erro	Iterações	Norma relativa do resíduo	Flag
4	Bl-BiCGStab	8.99e-005	1	3.3e-007	0
	BiCGStab-S	6.16e-005	1	2.27e-007	0
	Bl-GMRES	2.74	20	0.0288	1
	GMRES-S	5.78e-005	1	1.49e-007	0
8	Bl-BiCGStab	6.3e-005	1	3.13e-007	0
	BiCGStab-S	4.91e-005	1	2.38e-007	0
	Bl-GMRES	2.91	20	0.155	1
	GMRES-S	4.81e-005	1	1.63e-007	0
12	Bl-BiCGStab	9.85e-005	1	3.64e-007	0
	BiCGStab-S	5.78e-005	1	2.03e-007	0
	Bl-GMRES	2.87	20	0.242	1
	GMRES-S	5.53e-005	1	1.34e-007	0
16	Bl-BiCGStab	6.22e-005	1	3.21e-007	0
	BiCGStab-S	5.29e-005	1	2.53e-007	0
	Bl-GMRES	2.79	20	0.137	1
	GMRES-S	5.28e-005	1	1.72e-007	0
20	Bl-BiCGStab	7.88e-005	1	3.71e-007	0
	BiCGStab-S	5.67e-005	1	2.33e-007	0
	Bl-GMRES	2.87	20	0.241	1
	GMRES-S	5.55e-005	1	1.54e-007	0

Capítulo 11

Conclusão

O objetivo deste trabalho foi estudar métodos iterativos baseados no método de Arnoldi [14], para se gerar bases ortonormais, e no método da Bi-Ortogonalização de Lanczos [13], visando ao método do Gradiente Bi-Conjugado Estabilizado em Bloco com precondicionadores. O ponto de vista deste estudo foi o de utilizar projeções, ortogonais ou oblíquas, sobre subespaços de Krylov apropriados.

No Capítulo 2, apresentamos uma análise e síntese de artigos relevantes para este trabalho, tendo assim o estado da arte.

No Capítulo 3, foram feitas algumas definições e alguns teoremas que foram usados ao longo deste trabalho.

No Capítulo 4, apresentamos como resultado a Proposição 4.1, onde, a partir do Algoritmo 4.1, obtivemos o operador de projeção oblíqua sobre o subespaço de Krylov, utilizado no método de projeção apresentado.

No Capítulo 5, nas Proposições 5.1 e 5.2, temos como resultado que, pelo Algoritmo 5.1, geramos bases bi-ortogonais para os subespaços de Krylov considerados. Na Proposição 5.3, tratamos o método da Bi-Ortogonalização de Lanczos [13] sob o enfoque de projeções e, como resultado, obtivemos o operador de projeção que atua neste algoritmo.

No Capítulo 6, mostramos que, de fato, o Algoritmo 6.1 gera bases para o subespaço de Krylov escolhido. Na Proposição 6.1, obtivemos o operador de projeção ortogonal do método de Arnoldi [14].

No Capítulo 7, nas Proposições 7.1 e 7.2, mostramos que a solução aproximada gerada a partir do Algoritmo 7.1 está em um subespaço de Krylov, $\mathcal{K}$, e que a

base obtida para o subespaço de Krylov $\mathcal{L} = A\mathcal{K}$ pode ser gerada a partir da base do subespaço $\mathcal{K}$, usando-se , para isto, o Algoritmo 5.1. Nas Proposições 7.3 e 7.4, obtivemos como resultado os operadores de projeção do Algoritmo 5.1. Nas Proposições 7.5 e 7.6, temos os operadores de projeção envolvidos nos Algoritmos 7.2 e 7.3, respectivamente.

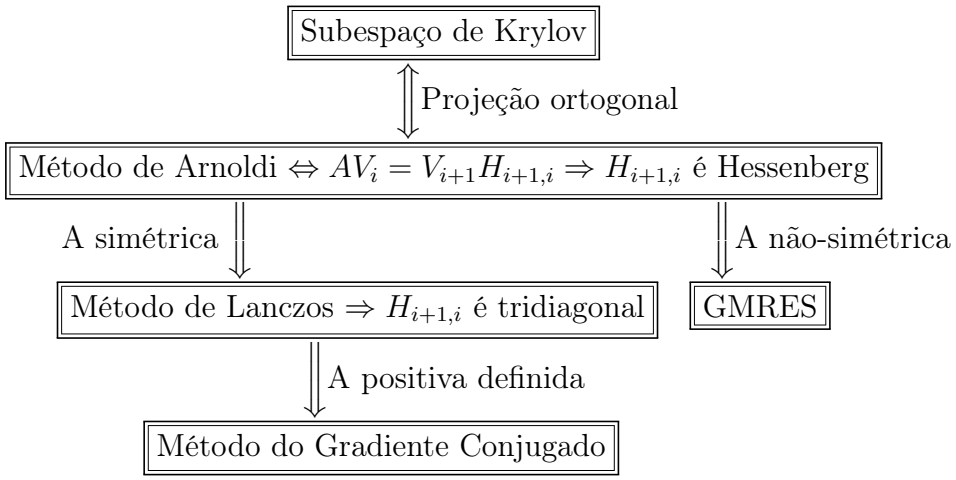
No Capítulo 8, a Proposição 8.1 assegura que Ar_i pode ser escrito como combinação linear dos vetores r_i, $i = 0, \ldots, m-1$. Nas Proposições 8.2 e 8.3, certificamos que os conjuntos obtidos a partir do Algoritmo 8.1 são bases para o subespaço de Krylov considerado. Na Proposição 8.4, apresentamos o operador de projeção do Algoritmo 8.1. Na Proposição 8.5, mostramos que os vetores, obtidos a partir do Algoritmo 8.2, podem ser expressos pelo produto de um polinômio em A pelo resíduo inicial, r_0.

No Capítulo 9, na Proposição 9.1, a partir do Algoritmo 9.1, mostramos que as direções AP_i e $A\tilde{P}_i$ podem ser expressas, respectivamente, como combinação linear de R_i e $\tilde{R}_i$, $i = 0, \ldots$. Na Proposição 9.2, provamos que os conjuntos gerados pelo Algoritmo 9.1 são bases para os subespaços de Krylov considerados. Na Proposição 9.3, provamos a ortogonalidade das direções R_i e $\tilde{R}_j$ e a A-ortogonalidade das direções $\tilde{P}_i$ e P_j, com $i > j$. Na Proposição 9.4, obtivemos o operador de projeção que atua no Algoritmo 9.1. Na Proposição 9.5, mostramos que as matrizes R_i e P_i, geradas a partir do Algoritmo 9.1, podem ser escritas como um produto de um polinômio matricial de A pelo resíduo inicial R_0.

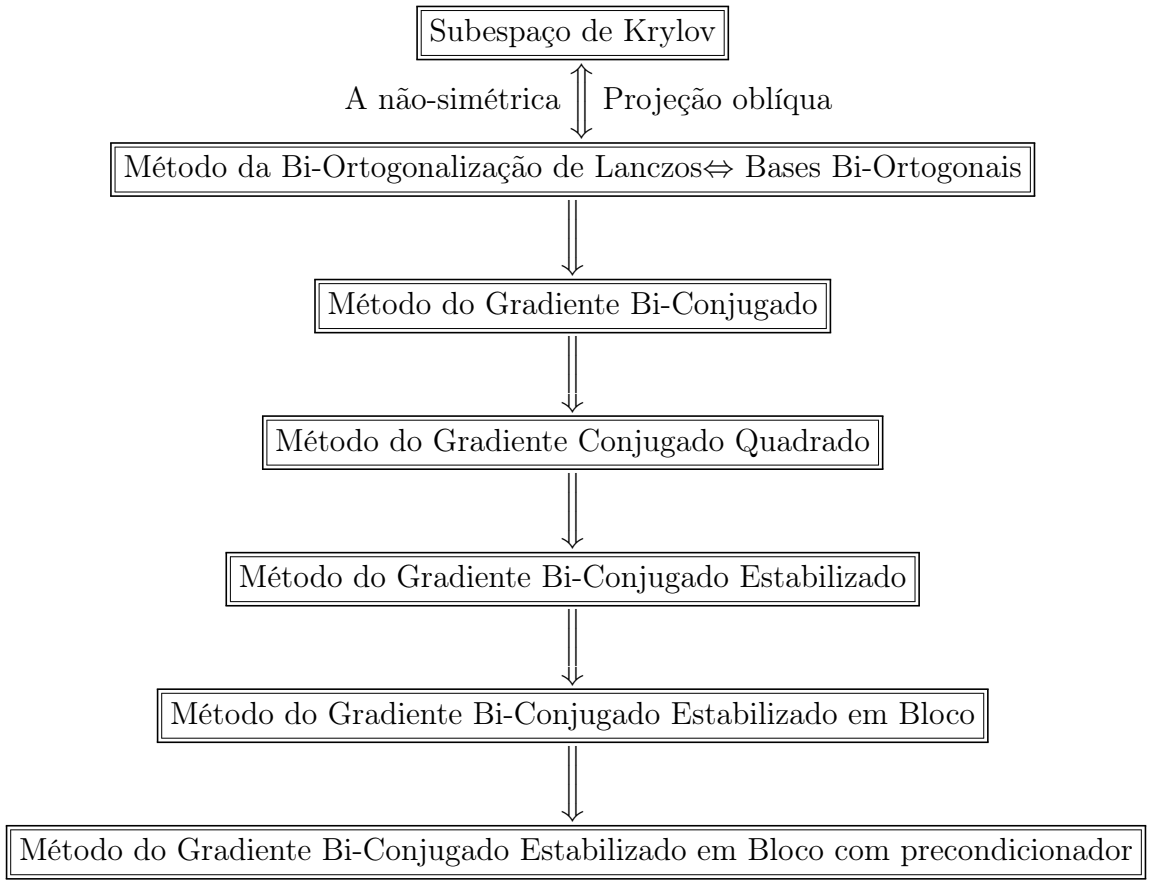
As proposições apresentadas são propostas novas, que não foram encontradas em nenhum dos textos pesquisados.

No capítulo 10, analisando as tabelas 1.1 a 1.15, podemos concluir que o método proposto e implementado, Bl-BiCGStab [7] com precondicionadores, se apresenta como uma opção para resolução de sistemas lineares de grande porte com múltiplos lados direitos. Ao compará-lo com o Bl-GMRES [21], temos que o Bl-BiCGStab [7] com precondicionadores apresenta uma melhor performance em um número menor de iterações. Quando confrontamos com os métodos do BiCGStab [17] e GMRES [15], aplicados s vezes, $s = 4, 8, 12, 16$ e 20, a performance do Bl-BICGStab [7] é equivalente à dos outros dois métodos.

Vejamos, a partir dos seguintes diagramas, a correlação entre os métodos abordados:



e



Vemos algumas hipóteses para trabalhos futuros, que podem ser divididos em três focos. Um deles seria manter o código do Bl-BiCGStab [7] e refinar a busca de

precondicionadores que pudessem tornar o método mais atraente. Um segundo caminho seria o de trabalhar teoricamente com o método. A partir daí, gerar novos teoremas e lemas pertinentes. Como terceira opção, vemos a possibilidade de gerar um novo método. Partindo do método de Lanczos [13], seria feito um procedimento análogo ao Bl-BiCGStab [7], porém escolhendo escalares distintos dos mais adotados na literatura e que resultam no Bl-BiCGStab [7]. Simultaneamente, seriam construídos argumentos teóricos, teoremas e lemas, que respaldassem o novo método desenvolvido.

Referências Bibliográficas

[1] SIMONCINI, V., GALLOPOULOS, E., "An Iterative Method for Nonsymmetric Systems with Multiple Right-Hand Sides", *SIAM Journal on Scientific Computing*, v. 16, n. 4, pp. 917–933, 1995.

[2] C. BREZINSKI, M. REDIVO-ZAGLIA, H. S., "Breakdowns in the Implementation of the Lánczos Method for Solving Linear Systems", *Computers & Mathematics with Applications*, v. 33, n. 1-2, pp. 31–44, January 1997.

[3] J. I. ALIAGA, D. L. BOLEY, R. W. F. V. H., "A Lanczos-Type Method for Multiple Starting Vectors", *Mathematics of Computation*, v. 69, n. 232, pp. 1577–1601, May 1999.

[4] DAI, H., "Block bidiagonalization methods for multiple nonsymmetric linear systems", *Numerical Mathemathics, A Journal of Chinese Universities*, v. 10, pp. 209–225, 2001.

[5] GRAVES-MORRIS, P. R., "The breakdowns of BiCGStab", *Numerical Algorithms*, v. 29, pp. 97–105, 2002.

[6] M. ROBBÉ, M. S., "A priori error bounds on invariant subspace approximations by block Krylov subspaces", *Linear Algebra and its Applications*, v. 350, pp. 89–103, 2002.

[7] GUENNOUNI, A. E., JBILOU, K., SADOK, H., "A Block Version of BICGSTAB for Linear Systems with Multiple Rigth-Hand Sides", *ETNA*, v. 16, pp. 129–142, 2003.

[8] K. JBILOU, A. MESSAOUDI, H., "Global FOM and GMRES algorithms for matrix equations", *Applied Numerical Mathematics*, v. 31, pp. 4963, 1999.

[9] GUTKNECHT, M. H., "Block Krylov Space Methods for Linear Systems With Multiple Right-hand Sides: an Introduction", *Modern Mathematical Models, Methods and Algorithms for Real World Systems (A.H. Siddiqi, I.S. Duff, and O. Christensen, eds.)*, pp. 420–447, 2007.

[10] BOUYOULI, R., JBILOU, K., SADAKA, R., etal., "Convergence properties of some block Krylov subspace methods for multiple linear systems", *J. Comput. Appl. Math.*, v. 196, n. 2, pp. 498–511, 2006.

[11] SHMELZER, M. H. G. . T., "The Block Grade of a Block Krylov Space", *Seminar for Applied Mathematics, Zurich, Switzerland*, July 2008.

[12] BOISVERT, R., *Matrix Market : A Web Resource for Test Matrix Collections*, Tech. rep., Townmeeting on Online Delivery of NIST Reference Data , NIST, Gaithersburg, MD, May 1997.

[13] LANCZOS, C., "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators", *Journal of Research of the National Bureau of Standards*, , n. 45, pp. 255–282, 1950.

[14] ARNOLDI, W., "The principle of minimized iteration in the solution of the matrix eigenvalue problem", *Quarterly of Applied Mathematics*, v. 9, pp. 17–25, 1951.

[15] SAAD, Y., SCHULTZ, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, v. 7, n. 3, pp. 856–869, 1986.

[16] EISENSTAT, S. C., ELMAN, H. C., SCHULTZ, M. H., "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations", *SIAM Journal on Numerical Analysis*, v. 20, n. 2, pp. 345–357, 1983.

[17] VAN DER VORST, H. A., "BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsymmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, v. 13, n. 2, pp. 631–644, 1992.

[18] FLETCHER, R., "Conjugate Gradient Methods for Indefinite Systems". In: *Proc. of the Dundee Biennial Conference on Numerical Analysis (1975)*, v. 506, *Lecture Notes in Mathematics*, pp. 73–89, 1975.

[19] SONNEVELD, P., "CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, v. 10, n. 1, pp. 36–52, 1989.

[20] O'LEARY, D., "The block conjugate gradient algorithm and related methods", *Linear Algebra and Its Applications*, v. 29, pp. 293–322, 1980.

[21] VITAL, B., *Etude de qualques methods de resolution de problemes lineaires de grande taille sur multiprocesseur*, Ph.D. Thesis, Universite de Rennes, 1990.

[22] I. S. DUFF, R. G. GRIMES, J. G. L., *User's Guide for the Harwell-Boeing Sparse Matrix Collection (ReleaseI)*, Tr/pa/92/86, CERFACS, October 1992.

[23] CULLUM, J., "Peaks, plateaus, numerical instabilities in a Galerkin minimal residual pair of methods for solving Ax=b", *Applied Numer. Math*, , n. 19, pp. 255–278, 1995.

[24] VAN DER VORST, H. A., *Iterative Krylov Methods for Large Linear Systems*. 2003.

[25] SAAD, Y., *Numerical Methods for Large Eigenvalue Problems*. 1992.

[26] SAAD, Y., *Iterative Methods for Sparse Linear Systems*. 2 ed. 2003.

[27] APOSTOL, T. M., *Linear Algebra, a First Course, With Applications to Differential Equations*. 1997.

[28] LANCZOS, C., "Solution of Systems of Linear Equations by Minimized Iterations", *Journal of Research of the National Bureau of Standards*, , n. 49, pp. 33–53, 1952.

[29] DAI, H., *Block Bidiagonalization Methods For Solving Nonsymmetric Linear Systems With Multiple Right-Hand Sides*, Technical Report TR/-PA/98/35, CERFACS, Toulouse, France, 1998.

[30] GUENNOUNI, A. E., JBILOU, K., SADOK, H., "The block Lanczos method for linear systems with multiple right-hand sides", *Applied Numerical Mathematics*, v. 51, n. 2-3, pp. 243–256, 2004.

[31] SIMONCINI, V., GALLOPOULOS, E., "Convergence Properties of Block GMRES and Matrix Polynomials", *Linear Algebra and its Applications*, v. 247, n. 1–3, pp. 97–119, 1996.

[32] DEMMEL, J. W., *Applied Numerical Linear Algebra*. 1997.

[33] BARRETT, R., BERRY, M., CHAN, T. F., etal., "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", 1994.

[34] BREZINSKI, C., ZAGLIA, M. R., "Block Projection Methods for Linear Systems", *Numerical Algorithms*, v. 29, n. 1, pp. 33–43, 2002.

[35] FREUND, R. W., NATCHTIGAL, N. M., "QMR: a Quasi-Minimal Residual Method for Non-Hermitian Linear Systems", *Numer. Math.*, v. 60, pp. 315–339, 1991.

[36] GOLUB, G. H., LOAN, C. F. V., *Matrix computations (3rd ed.)*. Baltimore, MD, USA, Johns Hopkins University Press, 1996.

[37] GOLUB, G. H., O'LEARY, D. P., "Some History of the Conjugate Gradient and Lanczos Algorithms: 1948–1976", *SIAM Review*, v. 31, n. 1, pp. 50–102, 1989.

[38] GREENBAUM, A., *Iterative Methods for Solving Linear Systems*. 1997.

[39] GREENBAUM, A., PTÁK, V., STRAKOUS, Z., "Any Nonincreasing Convergence Curve is Possible for GMRES", *SIAM J. Matrix Anal. Appl.*, v. 17, n. 3, pp. 465–469, 1996.

[40] HESTENES, M. R., STIEFEL, E., "Methods of Conjugate Gradients for Solving Linear Systems", *Journal of Research os the National Bureau of Standards*, v. 49, n. 6, pp. 409–436, December 1952.

[41] IPSEN, I. C. F., MEYER, C. D., "The Idea Behind Krylov Methods", *American Mathematical Monthly*, v. 105, n. 10, pp. 889–899, 1998.

[42] JBILOU, K., SADOK, H., "Oblique projection methods for linear systems with multiple right-hand sides ", *ETNA*, v. 20, pp. 119–138142, 2005.

[43] MORGAN, R. B., WILCOX, W., "Deflation of Eigenvalues for GMRES in Lattice QCD", *Nuclear Physics B - Proceedings Supplements*, v. 106, pp. 1067, 2002.

[44] MOTTA, V. S., CARVALHO, L. M., MACULAN FILHO, N., "Sistemas Lineares com Múltiplos Lados Direitos e Operadores de Projeção", *Anais do CMNE / CILAMCE 2007 , Congresso de Métodos Numéricos em Engenharia e Congresso Ibero Latino-Americano sobre Métodos Computacionais em Engenharia. Porto, Portugal*, 2007.

[45] PAIGE, C. C., SAUNDERS, M. A., "Solution of sparse indefinite systems of linear equations", *SIAM J. Numer. Anal.*, v. 12, pp. 617–629, 1975.

[46] SAAD, Y., "Practical Use of Some Krylov Subspace Methods for Solving Indefinite and Nonsymmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, v. 5, n. 1, pp. 203–228, 1984.

[47] SHEWCHUK, J. R., *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, Tech. rep., Pittsburgh, PA, USA, 1994.

[48] WIDLUND, O., "A Lanczos Method for a Class of Nonsymmetric Systems of Linear Equations", *SIAM Journal on Numerical Analysis*, v. 15, n. 4, pp. 801–812, 1978.

[49] YANG, U. M., GALLIVAN, K. A., "A new family of block methods", *Appl. Numer. Math.*, v. 30, n. 2-3, pp. 155–173, 1999.