COPPE/UFRJ

ESTUDOS SOBRE OS MÉTODOS ITERATIVOS DE KRYLOV PARA SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

Rafael Ferreira Lago

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

Orientadores: Nelson Maculan Filho
Luiz Mariano Paes de Carvalho Filho

Rio de Janeiro
Janeiro de 2010

ESTUDOS SOBRE OS MÉTODOS ITERATIVOS DE KRYLOV PARA SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

Rafael Ferreira Lago

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

Examinada por:

Prof. Nelson Maculan Filho, DSc.

Prof. Luiz Mariano Paes de Carvalho Filho, DSc.

Prof. Carlile Campos Lavor, DSc.

Prof. Alvaro Luiz Gayoso de Azevedo Coutinho, DSc.

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2010

Lago, Rafael Ferreira

Estudos Sobre os Métodos Iterativos de Krylov para Solução de Sistemas de Equações Lineares/Rafael Ferreira Lago. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIII, 189 p.: il.; 29, 7cm.

Orientadores: Nelson Maculan Filho

Luiz Mariano Paes de Carvalho Filho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 133 – 137.

1. Sistemas Lineares. 2. Método Iterativos. 3. Método de Krylov. 4. Subespaços de Krylov. I. Filho, Nelson Maculan *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

a Deus,
a Arão Ferreira de Almeida,
e a minha Aixa.

Agradecimentos

Gostaria de agradecer a todos os professores da COPPE pelos ensinamentos que se mostraram preciosos durate o decorrer do processo de produção desta dissertação.

Gostaria de agradecer em especial o professor Carlile Campos Lavor, com quem trabalhei por algum tempo. Gostaria de agradecer o professor Álvaro Luiz Gayoso de Azevedo Coutinho, por ter aceito participar da banca, e por seus conselhos a respeito deste trabalho. Também gostaria de agradecer os professores Luiz Mariano Paes de Caravlho Filho e Nelson Maculan Filho, pela sua orientação e grande contruibuição neste trabalho.

Também gostaria de agradecer os alunos com os quais estudei durante estes dois anos, em especial a Vinícius Leal do Forte e Michael de Souza, pelas suas caronas e incansáveis tardes de estudo na COPPE.

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

ESTUDOS SOBRE OS MÉTODOS ITERATIVOS DE KRYLOV PARA SOLUÇÃO DE SISTEMAS DE EQUAÇÕES LINEARES

Rafael Ferreira Lago

Janeiro/2010

Orientadores: Nelson Maculan Filho
Luiz Mariano Paes de Carvalho Filho

Programa: Engenharia de Sistemas e Computação

Nesta dissertação, discutimos vários aspectos de métodos iterativos que utilizam espaços de Kyrlov, desde interpretações geométricas a implantações algorítmicas. Explicamos e damos várias interpretações a respeito de recomeço, truncamento e subespaços de aumento. Desenvolvemos representações gráficas de vários destes esquemas. Propomos um novo método chamado OT, e apresentamos resultados numéricos comparando este método com alguns outros métodos já conhecidos na literatura.

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

ON ITERATIVE KRYLOV METHODS FOR THE SOLUTION OF LINEAR SYSTEM OF EQUATIONS

Rafael Ferreira Lago

January/2010

Advisors: Nelson Maculan Filho
Luiz Mariano Paes de Carvalho Filho

Department: Systems Engineering and Computer Science

In this dissertation we discuss a wide range of aspects of iterative methods using Krylov subspaces, from geometric interpretations to algorithm implantations. We explain and provide some interpretations on restart, truncation, and augmented subspaces. We develop graphical representation of some of these strategies. We propose a new method named OT, and we present numerical results comparing this method and other methods well-known in this field.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Nesta dissertação, estudaremos um espaço conhecido como *"espaço de Krylov"* e vários métodos iterativos que utilizam tal espaço para buscar solução de sistemas do tipo

$$Ax = b, \tag{1.0.1}$$

onde $A \in \mathbb{C}^{n\times n}$, $x, b \in \mathbb{C}^n$.

Sempre que mencionarmos tais sistemas aqui, consideraremos que trata-se de um sistema de grande porte[1], e desejavelmente esparso, ou que a matriz A não é conhecida (pode ser dada por uma função $f(x)$). Isto porque sistemas lineares de menor proporção podem facilmente resolvidos por métodos já conhecidos e estáveis, como a Eliminação Gaussiana, e a aplicação de métodos iterativos torna-se desnecessária, ou mesmo desaconselhável.

O trabalho está organizado da seguinte forma. No capítulo 2, introduziremos o conceito de espaço e subespaço de Krylov e apresentaremos uma propriedade chave a respeito de tal subespaço, que mostra em que casos é possível encontrar a solução de um determinado sistema linear neste subespaço.

No capítulo 3, discutiremos como encontrar uma base adequada para um subespaço de Krylov e, ao dispormos de tal, como encontrar a solução do sistema linear utilizando-a.

No extenso capítulo 4, discutiremos uma série de métodos para solução de sistemas lineares que estão relacionados a espaços de Krylov. Mostraremos algumas equivalências entre os mesmos ao considerar algumas propriedades especiais de A, mostrando que há uma íntima ligação entre estes métodos. Discutiremos também, brevemente, ao fim de cada método, as desvantagens do mesmo. Veremos que algumas destas desvantagens são tão sérias que podem inviabilizar tais métodos para alguns problemas específicos.

No capítulo 5, faremos uma discussão profunda a respeito das relações entre os

[1] onde n é muito grande, em alguns casos, da ordem de 10^9 ou superior

métodos explicitados no capítulo 4. Formalizaremos uma interpretação geométrica a respeito dos subespaços em questão e vetores gerados pelos métodos, e veremos que podemos classificar estes métodos em duas grandes famílias de métodos, RO e RM. Mostraremos algumas propriedades de cada família de método e algumas relações entre estas famílias. Em especial, na seção 5.7 deste capítulo, faremos uma breve discussão a respeito da convergência de métodos iterativos que utilizam espaços de Krylov.

No capítulo 6, veremos os efeitos da estratégia do recomeço nos métodos iterativos que utilizam espaço de Krylov, e veremos algumas interpretações alternativas a respeito desta estratégia, em especial a interpetação com relação à negligência de ortogonalização, na seção 6.3. Veremos que esta técnica pode gerar estagnação, o que a torna impraticável em alguns casos.

No capítulo 7, discutiremos uma estratégia para tentar solucionar o problema da estagnação e/ou acelerar métodos que utilizam subespaços de Krylov. Apresentaremos algumas estratégias de aumento do subespaço de Krylov e sua implantação. Alguns dos métodos descritos aqui são apresentados sob uma ótica que, até onde sabemos, é nova. A seção 7.7 é bastante relevante, e nela apresentaremos uma representação gráfica de algumas estratégias de aumento apresentadas.

No capítulo 8, veremos que mesmo os métodos apresentados no capítulo 7 continuam inviáveis. Propomos então diferentes tipos de truncamento, e mostramos o efeito do uso de tal técnica. Explicaremos qual forma de truncamento foi proposta por cada método e explicaremos a razão da escolha do truncamento de cada método. Novamente, na seção 8.3 veremos uma interpretação geométrica do truncamento ótimo proposto neste capítulo.

Em especial, no capítulo 8, mostraremos um método novo que propomos que chamamos de OT (*"Optimal Truncation"*, ou *"Truncamento Ótimo"*). Embora a estratégia tomada por este método já fosse conhecida, ela foi considerada inviável.

No capítulo 9, discutiremos os resultados numéricos da implantação em Matlab de alguns dos algoritmos discutidos. Em particular, discutiremos na seção 9.5 os resultados numéricos que mostram que o nosso método pode ser considerado viável, ou mesmo aconselhável quando aplicado a alguns tipos de problema.

Discutiremos na seção 9.6 deste mesmo capítulo a nossa implantação do método GCR na biblioteca PETSc, e explicaremos o funcionamento básico da mesma.

1.1 Notação

Antes de prosseguirmos, estabeleceremos algumas notações a serem utilizadas durante todo o trabalho.

Sempre que possível, denotaremos matrizes por caracteres itálicos maiúsculos,

e subespaços por caracteres cursivos maiúsculos. Por exemplo B e $\mathcal{B}$ denotarão uma matriz e um subespaço respectivamente. Consideraremos que todo e qualquer subespaço mencionado está contido num espaço de Hilbert. Denotaremos vetores por caracteres minúsculos, como b por exemplo, e polinômios por caracteres em negrito, como em $\mathbf{b}(t)$.

Denotaremos por $\text{img}(A)$ a imagem de uma matriz A e $\text{nul}(A)$ o espaço nulo de tal matriz. Também notaremos por

$$\text{span}\{u_1, u_2, \ldots, u_k\}, u_j \in \mathbb{C}^n, j = 1, 2, ..., n \tag{1.1.1}$$

o espaço gerado pelos vetores $u_1, u_2, \ldots, u_k$.

Seja $\mathcal{U}$ um subespaço de Hilbert qualquer. Consideraremos que $\mathcal{U}^\perp$ é o subespaço ortogonal à $\mathcal{U}$.

Caso as colunas de uma matriz U formem uma base para um subespaço $\mathcal{U}$, mencionaremos livremente U como sendo uma base para o subespaço U.

Seja M uma matriz qualquer. Denotaremos por M^H a Hermitiana (ou transposta conjugada) da matriz M, e por e M^* a conjugada da matriz M. Tomaremos o produto interno Euclidiano da seguinte forma:

$$\langle a, b\rangle = a^H b. \tag{1.1.2}$$

Qualquer produto interno em especial deverá se especificado. Consideremos um produto interno $\langle a, b\rangle_{\mathcal{Z}}$ já especificado. Então denotaremos $\|.\|_{\mathcal{Z}}$ a norma sujeita ao produto interno $\langle a, b\rangle_{\mathcal{Z}}$, ou seja $\|a\|_{\mathcal{Z}} = \sqrt{\langle a, a\rangle}_{\mathcal{Z}}$. No caso da norma sujeita ao produto interno Euclidiano denotaremos por $\|.\|_2$.

Em alguns casos utilizaremos também o delta de Kronecker, que é

$$\delta_{ij} = \begin{cases} 1, \text{se i=j} \\ 0, \text{se i}\neq\text{j} \end{cases} \tag{1.1.3}$$

Tomaremos a matriz $I_k \in \mathbb{C}^{k\times k}$ como sendo a identidade de dimensão k.

Porfim, consideraremos que dispomos de uma máquina com precisão infinita durante todo este trabalho, exceto quando explicitamente mencionado.

Capítulo 2

Espaço de Krylov

2.1 Propriedades Básicas

Como veremos mais a frente, o espaço de Krylov desempenha um papel importante no desenvolvimento teórico da maioria dos métodos que mostraremos aqui. Embora alguns destes métodos abandonem o espaço de Krylov durante a execução, ainda assim boa parte do entendimento do método estará sujeito às propriedades deste espaço. Nesta seção iremos apresentar o conceito do espaço de Krylov bem como algumas propriedades do mesmo. Começamos com a definição:

Definição 2.1.1. *Sejam $A \in \mathbb{C}^{n\times n}$ e $0 \neq b \in \mathbb{C}^n$. Chamaremos o espaço*

$$\mathcal{K}(A,b) = \text{span}\{b, Ab, A^2b, \dots\} \tag{2.1.1}$$

de espaço de Krylov associado à A e b e o espaço

$$\mathcal{K}_k(A,b) = \text{span}\{b, Ab, A^2b, \dots, A^{k-1}b\} \tag{2.1.2}$$

de subespaço de Krylov de ordem k, associado à A e b. Também notaremos $\mathcal{K}_k$ quando não houver ambiguidade.

Como $b \in \mathbb{C}^n$, então $\dim\left(\mathcal{K}_j\right)$ será menor ou igual a n, e portanto existirá um primeiro valor k tal que A^kb será linearmente dependente com os vetores anteriores da sequência de Krylov, ou seja

$$A^kb = \sum_{i=0}^{k-1} \alpha_i A^i b \tag{2.1.3}$$

Apresentamos mais uma definição retirada de [1], p.646:

Definição 2.1.2 (Polinômio Mínimo de Vetores)**.** *O polinômio mínimo de um vetor b com relação a matriz A, que notaremos por $\mathbf{v}(t)$, é o único polinômio mônico*

de menor grau tal que $\mathbf{v}(A)b = 0$.

Se tivermos o menor índice k tal que $A^k b$ é linearmente dependente, bem como os coeficientes de tal combinação linear, podemos calcular o polinômio mínimo de b com relação a A da seguinte forma:

$$A^k b = \sum_{i=0}^{k-1} \alpha_i A^i b \qquad \mathbf{v}(A)b = 0 \tag{2.1.4}$$

$$\left(A^k - \sum_{i=0}^{k-1} \alpha_i A^i \right) b = 0 \qquad \mathbf{v}(t) = t^k - \sum_{i=0}^{k-1} \alpha_i t^i \tag{2.1.5}$$

Com isto, formalizamos uma proposição que diz respeito aos subespaços de Krylov, que pode ser encontrada em [2, p.145]:

Proposição 2.1.3. *Seja k o grau do polinômio mínimo de b com relação a A. Então* $\dim\left(\mathcal{K}_j(A,b)\right) = \min(j,k)$.

Demonstração. Basta verificar que

$$A^{k+1}b = A\sum_{i=0}^{k-1} \alpha_i A^i b = A\left(\sum_{i=0}^{k-2} \alpha_i A^i b + \alpha_{k-1} A^{k-1} b \right) \tag{2.1.6}$$

$$= \sum_{i=0}^{k-2} \alpha_i A^{i+1} b + \alpha_{k-1} A^k b = \sum_{i=0}^{k-2} \alpha_i A^{i+1} b + \alpha_{k-1} \sum_{i=0}^{k-1} \alpha_i A^{i+1} b, \tag{2.1.7}$$

que é uma combinação linear dos primeiros k vetores da sequência de Krylov. Por indução temos que que para qualquer $l > k$, $A^l b$ poderá ser escrito como uma combinação linear dos k primeiros vetores da sequência de Krylov. □

Vamos agora apresentar a definição de polinômio mínimo, que também pode ser encontrada em [1], p.642, e que será muito útil nas próximas seções.

Definição 2.1.4 (Polinômio Mínimo de Matrizes)**.** *O polinômio mínimo de uma matriz A, que notaremos por $\mathbf{q}(t)$, é o único polinômio mônico de menor grau tal que $\mathbf{q}(A) = 0$.*

O polinômio mínimo $\mathbf{q}(t)$ de uma matriz A é dado por

$$\mathbf{q}(t) = \prod_{i=1}^{j} (t - \lambda_i)^{k_i} \tag{2.1.8}$$

onde j é o número de autovalores diferentes de A e λ_i é um autovalor de índice k_i da matriz A (todas estas definições e demonstrações podem ser encontradas em [1] com mais detalhes).

2.2 Caso Não-Singular

Vamos discutir uma propriedade especial do espaço de Krylov $\mathcal{K}_k(A, b)$ quando A é não-singular, que será muito útil na compreensão dos métodos de Krylov.

Se ℓ é o grau do polinômio mínimo da matriz A, então podemos escrevê-lo como

$$\mathbf{q}(t) = \sum_{i=0}^{\ell} \alpha_i t^i. \tag{2.2.1}$$

Como por definição $\mathbf{q}(A) = 0$, e como supomos que A é não-singular, então

$$\begin{aligned} A^{-1}\mathbf{q}(A) &= A^{-1}\left(\alpha_0 I + \alpha_1 A + \cdots + \alpha_\ell A^\ell\right) = 0 \\ \alpha_0 A^{-1} &= -\sum_{i=1}^{\ell} \alpha_i A^{i-1}. \end{aligned} \tag{2.2.2}$$

De (2.1.8) percebemos que α_0 é o produtório dos λ_i. Como supomos que A é não-singular, então todos os autovalores de A são diferentes de zero e portanto $\alpha_0 \neq 0$. Com isso

$$A^{-1} = -\frac{1}{\alpha_0}\sum_{i=1}^{\ell} \alpha_i A^{i-1}, \tag{2.2.3}$$

que é uma forma de representar A^{-1}. Voltando ao sistema $Ax = b$, temos

$$x = A^{-1}b = -\frac{1}{\alpha_0}\sum_{i=1}^{\ell} \alpha_i A^{i-1}b = -\frac{1}{\alpha_0}(\alpha_1 b + \alpha_2 Ab + \cdots + \alpha_\ell A^{\ell-1}b) \tag{2.2.4}$$

o que significa que $x \in \mathcal{K}_\ell(A, b)$, pois pode ser escrito como uma combinação linear dos vetores que geram tal espaço.

2.3 Caso Singular

Quando a matriz A é singular não podemos pré-multiplicar pela inversa em (2.2.2), e portanto não temos uma prova de que x pertença ao espaço $\mathcal{K}_\ell(A, b)$. O que veremos nesta seção é que, na verdade, nem sempre x pertencerá a tal espaço quando a matriz é singular, mesmo que exista uma solução. Estudaremos os casos em que isto ocorre.

Primeiramente, consideremos a seguinte definição:

Definição 2.3.1 (Índice de uma Matriz)**.** *Seja $A \in \mathbb{C}^{n\times n}$ uma matriz singular. O menor número inteiro positivo k tal que* $\mathrm{img}(A^k)$ *e* $\mathrm{nul}(A^k)$ *são subespaços complementares, ou seja*

$$\mathbb{C}^{n\times n} = \mathrm{img}(A^k) \oplus \mathrm{nul}(A^k) \tag{2.3.1}$$

é chamado de índice de A, notado por $\mathrm{ind}(A)$*. Caso A seja não-singular, então*

$\text{ind}(A) = 0$.

Tal índice existe para toda matriz singular, conforme é mostrado em [1] pela Decomposição da Imagem-Núcleo, p.394. Consideremos agora o seguinte teorema que resulta na decomposição cerne-nilpotente:

Teorema 2.3.2 (Teorema da Decomposição Cerne-Nilpotente)**.** *Se $A \in \mathbb{C}^{n\times n}$ é uma matriz singular tal que* $\text{ind}(A) = k$ *e posto*$(A^k) = r$*, então existe uma matriz não-singular Q tal que*

$$Q^{-1}AQ = \begin{bmatrix} C & 0 \\ 0 & N \end{bmatrix} \tag{2.3.2}$$

onde $C \in \mathbb{C}^{r\times r}$ é não-singular e N é nilpotente de índice k.

Demonstração. A prova deste teorema pode ser encontrada com detalhes em [1], p.397. □

Reparemos que caso A seja não-singular o teorema ainda é válido, pois $\text{ind}(A)$ seria zero, r seria igual a n e consequentemente $A = C$.

Voltemos a solução de $Ax = b$. De acordo com o teorema, podemos escrever esse sistema como

$$\begin{bmatrix} C & 0 \\ 0 & N \end{bmatrix} \tilde{x} = \tilde{b}, \tag{2.3.3}$$

onde $\tilde{x} = Q^{-1}x$ e $\tilde{b} = Q^{-1}b$. Além disto vamos particionar $\tilde{x}$ e $\tilde{b}$

$$\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}, \qquad \tilde{b} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{bmatrix} \tag{2.3.4}$$

de modo que

$$C\tilde{x}_1 = \tilde{b}_1 \tag{2.3.5}$$

$$N\tilde{x}_2 = \tilde{b}_2. \tag{2.3.6}$$

Vamos agora supor que $x \in \mathcal{K}_k(A, b)$ e analisar quais são as condições para que tal afirmação seja verdadeira. Se o grau do polinômio mínimo de C é ℓ, então

$$\tilde{x} = \sum_{i=0}^{\ell-1} \alpha_i \begin{bmatrix} C^i & 0 \\ 0 & N^i \end{bmatrix}, \tag{2.3.7}$$

$$\tilde{x}_1 = \sum_{i=0}^{\ell-1} \alpha_i C^i \tilde{b}_1, \tag{2.3.8}$$

$$\tilde{x}_2 = \sum_{i=0}^{\ell-1} \alpha_i N^i \tilde{b}_2, \tag{2.3.9}$$

para algum conjunto de α_i. Como C é não-singular, então $\tilde{x}_1 \in \mathcal{K}_\ell(C, \tilde{b}_1)$, logo (2.3.8) será igual a $\tilde{x}_1 = C^{-1}\tilde{b}_1$ supondo uma escolha adequada dos α_i, o que confere com (2.3.5).

Substituindo (2.3.9) em (2.3.6) temos que

$$\sum_{i=0}^{\ell-1} \alpha_i N^{i+1} \tilde{b}_2 = \tilde{b}_2 \tag{2.3.10}$$

$$\left(I - \sum_{i=0}^{\ell-1} \alpha_i N^{i+1} \right) \tilde{b}_2 = 0 \tag{2.3.11}$$

Entretanto, analisando os autovalores da matriz entre parênteses, temos que

$$\left(I - \sum_{i=0}^{\ell-1} \alpha_i N^{i+1} \right) v = \lambda v \tag{2.3.12}$$

$$\sum_{i=0}^{\ell-1} \alpha_i N^{i+1} v = (1-\lambda) v \tag{2.3.13}$$

$$\tag{2.3.14}$$

Em geral, o somatório de matrizes nilpotentes não será nilpotente, mas no nosso caso, elevando $\left(\sum_{i=0}^{\ell-1} \alpha_i N^{i+1} \right)$ a k o termo de menor grau do somatório sera $N^k = 0$ e portanto este somatório também será uma matriz nilpotente de índice k.

Como todos os autovalores de uma matriz nilpotente são iguais a zero, então $(1-\lambda) = 0$ o que nos leva à conclusão de todos os autovalores de $\left(I - \sum_{i=0}^{\ell-1} \alpha_i N^{i+1} \right)$ são iguais a 1, e portanto, tal matriz é inversível. Sabendo disto e tendo (2.3.11) em mente, vemos que uma condição necessária para que o sistema tenha solução em $\mathcal{K}_k(A, b)$ é que $\tilde{b}_2 = 0$.

Vamos agora abandonar a hipótese de que $x \in \mathcal{K}_k(A, b)$ e supor apenas que $\tilde{b}_2 = 0$. Vamos supor que $\tilde{x}_2 = 0$, o que satisfaz a equação (2.3.6), restando apenas analisar (2.3.5). Como C é não-singular, de (2.2.4) teremos $\tilde{x}_1 = \sum_{i=0}^{\ell-1} \alpha_i C^i \tilde{b}_1$, e com

isso

$$\begin{aligned} x =& Q\tilde{x} = Q \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = Q \begin{bmatrix} \sum_{i=0}^{\ell-1} \alpha_i C^i \tilde{b}_1 \\ 0 \end{bmatrix} \\ =& Q \begin{bmatrix} \sum_{i=0}^{\ell-1} \alpha_i C^i & 0 \\ 0 & \sum_{i=0}^{\ell-1} \alpha_i N^i \end{bmatrix} \begin{bmatrix} \tilde{b}_1 \\ 0 \end{bmatrix} = Q \sum_{i=0}^{\ell-1} \alpha_i \begin{bmatrix} C^i & 0 \\ 0 & N^i \end{bmatrix} Q^{-1} b \\ =& \sum_{i=0}^{\ell-1} \alpha_i A^i b, \end{aligned} \tag{2.3.15}$$

o que mostra que $x \in \mathcal{K}_\ell(A, b)$.

Para estabelecer a mesma notação encontrada em [3], reparemos que

$$A^k = Q \begin{bmatrix} C^k & 0 \\ 0 & 0 \end{bmatrix} Q^{-1} \tag{2.3.16}$$

e que caso $b \in \text{img}(A^k)$, então podemos escrever b como uma combinação linear das colunas de A^k, ou seja, o sistema $A^k y = b$ tem solução

$$\begin{bmatrix} C^k & 0 \\ 0 & 0 \end{bmatrix} Q^{-1} y = Q^{-1} b. \tag{2.3.17}$$

Particionando $Q^{-1}y = [\tilde{y}_1 \ \tilde{y}_2]^T$ obtemos os sistemas $C^k \tilde{y}_1 = \tilde{b}_1$, que sempre tem solução pois C é não-singular, e $0 = \tilde{b}_2$. Portanto, $b \in \text{img}(A^k)$ é equivalente a $\tilde{b}_2 = 0$.

No teorema que segue formalizamos então as condições necessárias e suficientes para que a solução do sistema $Ax = b$ se encontre no espaço de Kyrlov:

Teorema 2.3.3. *Seja ℓ o grau do polinômio mínimo de b com relação a A, e seja k o índice de A. O sistema linear $Ax = b$ possui solução no subespaço de Krylov $\mathcal{K}_\ell(A, b)$ se e somente se $b \in \text{img}(A^k)$.*

Demonstração. Primeiramente, consideremos que

$$\mathbf{v}(t) = \sum_{i=0}^{l} \beta_i t^i \tag{2.3.18}$$

é o polinômio mínimo de $\tilde{b}_1$ com relação à C, cujo grau é l. Aplicando A a este

polinômio e pós-multiplicando por b obtemos

$$\mathbf{v}(A)b = \sum_{i=0}^{l} \beta_i A^i b = Q \sum_{i=0}^{l} \beta_i \begin{bmatrix} C^i & 0 \\ 0 & N^i \end{bmatrix} \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{bmatrix} \tag{2.3.19}$$

Como supomos que $b \in \text{img}(A^k)$ e como vimos anterioremente que esta afirmação é equivalente a $\tilde{b}_2 = 0$, teremos

$$\mathbf{v}(A)b = Q \begin{bmatrix} \sum_{i=0}^{j} \beta_i C^i \tilde{b}_1 \\ 0 \end{bmatrix} = Q \begin{bmatrix} v(C)\tilde{b}_1 \\ 0 \end{bmatrix} = 0, \tag{2.3.20}$$

e portanto $\mathbf{v}(t)$ também é o polinômio mínimo de b com relação a A, e consequentemente $l = \ell$.

Seja h o grau do polinômio mínimo de C, obtido da decomposição (2.3.2). Reparemos que ℓ sempre será menor ou igual a h, pois no pior caso o polinômio minímo de $\tilde{b}$ com relação a C será o próprio polinômio mínimo de C.

Conforme foi visto nesta seção e na seção anterior, se $b \in \text{img}(A^k)$ então $x \in \mathcal{K}_h(A, b)$. Mas de acordo com a proposição 2.1.3 teremos $\dim(\mathcal{K}_h(A, b)) = \min(h, \ell)$, e como $\ell \leq h$, então na verdade a solução pertence ao subespaço $K_\ell(A, b)$. □

Reforçamos que este resultado é válido apenas considerando precisão infinita. Quando consideramos o erro de aproximação gerado pela aritmética de ponto flutuante, podemos em muitos casos necessitar de mais do que ℓ iterações para que a convergência do algoritmo ocorra.

Portanto, sempre que mencionarmos este resultado estaremos nos referindo ao caso de precisão infinita.

Capítulo 3

Bases

Conforme foi mostrado no teorema 2.3.3, é possível escrever a solução de $Ax = b$ como uma combinação linear dos vetores do espaço de Krylov associado à A e b. A vantagem de buscar um vetor x no espaço de Krylov é que este espaço pode ser muito menor que o espaço de busca completo. Entretanto, para efetuarmos a busca de x no espaço de Krylov, precisamos de uma base para este espaço. Neste capítulo veremos alguns métodos para encontrar uma base para o espaço de Krylov e suas particularidades.

3.1 Processo de Ortogonalização de Arnoldi

Seja $\mathcal{K}_k(A, b) = \text{span}\{v_1, v_2, \ldots, v_k\}$ e $V_k = [v_1\ v_2\ \ldots\ v_k]$. A princípio poderiamos considerar $v_1 = b$, $v_2 = Ab$, $\ldots$, $v_k = A^{k-1}b$ seguindo a definição dos subespaços de Krylov, mas esta sequência de vetores converge para o autovetor associado ao maior autovalor[1] da matriz A em módulo, e portanto, tende a tornar-se linearmente dependente quando utilizamos uma máquina com precisão finita, o que faz com que esta base seja mal-condicionada. Se por outro lado ortogonalizarmos cada vetor, teremos uma base com uma estabilidade melhor.

Apresentaremos agora o método que será o mais utilizado no nosso trabalho para a ortogonalização de bases para espaços de Krylov, o *processo de ortogonalização de Arnoldi*.

Tomando um vetor inicial v_1, o processo de ortogonalização de Arnoldi irá gerar cada v_k através da ortonormalização de Av_{k-1} com relação os vetores gerados previamente, utilizando o processo de Gram-Schmidt ou Gram-Schmidt Modificado[2].

[1] é exatamente neste princípio que se baseia o Método das Potências para encontrar o maior autovetor de uma matriz. Para mais informações, consulte [4], capítulo 7.3

[2] novamente, quando utilizamos uma máquina com precisão finita, mesmo o método de Gram-Schmidt ou Gram-Schmidt Modificado podem apresentar perda de ortogonalidade para alguns problemas. Recomendamos a leitura de [5] para mais informações a respeito da estabilidade do método de Gram-Schmidt

Como queremos que o espaço gerado pelos vetores de Arnoldi formem uma base para o subespaço de Krylov $\mathcal{K}(A, b)$, basta fazermos $v_1 = b/\|b\|_2$. O algoritmo 1 mostra este esquema. Mais a frente explicaremos o porque do critério de parada selecionado.

Algoritmo 1: Processo de ortogonalização de Arnoldi usando Gram-Schmidt Modificado

Input: A, b

$v_1 = \dfrac{b}{\|b\|_2}$;
for $j = 1...$ **do**
  $v_{j+1} = Av_j$;
  for $i = 1..j$ **do**
    $\eta_{(i,j)} = \left\langle v_i, v_{j+1} \right\rangle$;
    $v_{j+1} = v_{j+1} - \eta_{(i,j)} v_i$;
  end for
  $\eta_{(j+1,j)} = \left\| v_{j+1} \right\|_2$;
  if $\eta_{(j+1,j)} = 0$ **then** Stop;
  $v_{j+1} = \dfrac{v_{j+1}}{\eta_{(j+1,j)}}$;
end for

Seja $\tilde{H}_k \in \mathbb{C}^{k+1 \times k}$ a matriz contendo os coeficientes $\eta_{i,j}$, onde k é o número de iterações de Arnoldi aplicadas. Analisando o algoritmo, vemos que tal matriz terá a forma

$$\tilde{H}_k = \begin{bmatrix} * & * & * & \dots & * \\ * & * & * & \dots & * \\ & * & * & \dots & * \\ & & \ddots & \ddots & \vdots \\ & & & * & * \\ & & & & * \end{bmatrix}, \tag{3.1.1}$$

ou seja, será uma matriz de Hessenberg superior. A princípio tal matriz seria desnecessária, e poderíamos obter uma base ortonormal utilizando apenas uma variável α ao invés dos $\eta_{i,j}$, mas como veremos adiante, a informação obtida nesta matriz poderá será muito útil.

Uma simples indução mostra que este algoritmo atualiza v_{k+1} da seguinte forma:

$$\eta_{(k+1,k)} v_{k+1} = Av_k - \eta_{(1,k)} v_1 - \eta_{(2,k)} v_2 - \cdots - \eta_{(k,k)} v_k \tag{3.1.2}$$

$$Av_k = \sum_{i=0}^{k+1} \eta_{(i,k)} v_i = V_{k+1} h_k \tag{3.1.3}$$

onde h_k é a a k-ésima coluna de $\tilde{H}_k$. Generalizando para todos os vetores de V_k,

teremos a importante relação

$$AV_k = V_{k+1}\tilde{H}_k = V_k H_k + \eta_{(j+1,j)} v_{k+1} e_k^T \tag{3.1.4}$$

onde $H_k \in \mathbb{C}^{k\times k}$ é formado pelas k primeiras linhas de $\tilde{H}_k$.

Como foi afirmado anteriormente, a sequência de vetores que formam o espaço de Krylov convergem para um autovetor de A, sendo natural a idéia de que em algum momento, o vetor gerado pelo processo de Arnoldi seja uma combinação linear dos vetores gerados anterioremente, e portanto, ao ortogonalizarmos este novo vetor com os anteriores obteremos o vetor zero.

De acordo com a proposição 2.1.3, isto acontece precisamente na iteração ℓ, onde ℓ é o grau do polinômio mínimo de b com relação a A. Nesta iteração teremos os vetores $\{v_1, v_2, \ldots, v_\ell\}$ que constituem uma base para o espaço K_ℓ, e de acordo com o teorema 2.3.3, já podemos escrever uma solução com essa base. Como $\eta_{(j+1,j)} = \left\|v_{j+1}\right\|_2$ e $\|v_{\ell+1}\|_2 = 0$, fica claro que o critério de parada estabelecido no algoritmo 1 irá ser satisfeito apenas quando já tivermos um base para $\mathcal{K}_\ell(A, b)$. Outro fato interessante é que, de 3.1.4 temos

$$AV_\ell = V_\ell H_\ell, \tag{3.1.5}$$

ou seja, obtivemos um subespaço invariante de A na iteração ℓ.

3.2 Processo de Ortogonalização de Lanczos

Utilizando o resultado obtido em (3.1.5) vemos que caso A seja simétrica, H_ℓ também será simétrica, o que nos leva à conclusão de que H_k terá a forma

$$H_k = \begin{bmatrix} \eta_1 & \nu_1 & & & & & \\ \nu_1 & \eta_2 & \nu_2 & & & & \\ & \nu_2 & \eta_3 & \nu_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \nu_{k-2} & \eta_{k-1} & \nu_{k-1} & \\ & & & & \nu_{k-1} & \eta_k & \\ & & & & & \nu_k & \end{bmatrix}. \tag{3.2.1}$$

Isto significa que só precisamos ortogonalizar cada vetor com relação aos dois vetores anteriores, reduzindo significativamente o custo computacional associado à ortogonalização. O método que utiliza esta estratégia para gerar um base ortonormal para o espaço de Krylov associado a A e b é chamado de processo de ortogonalização de Lanczos, cujo pseudocódigo pode ser encontrado no algoritmo 2.

Algoritmo 2: Processo de ortogonalização de Lanczos

Input: A, b

$v_1 = \dfrac{b}{\|b\|_2}$;
$\nu_0 = 0$;
for $j = 1...$ **do**
  $v_{j+1} = Av_j$;
  $\eta_j = \left\langle v_j, v_{j+1} \right\rangle$;
  $v_{j+1} = v_{j+1} - \eta_j v_j - \nu_{j-1} v_{j-1}$;
  $\nu_j = \left\| v_{j+1} \right\|_2$;
  if $\nu_j = 0$ **then** Stop;
  $v_{j+1} = \dfrac{v_{j+1}}{\nu_j}$;
end for

Assim como o processo de ortogonalização de Arnoldi, este método pára quando encontra um vetor igual a zero, o que ocorre na iteração ℓ, onde ℓ é o grau do polinômio mínimo de b com relação a A, no que se baseia o critério de parada mostrado no algoritmo.

3.3 Encontrando uma Solução

Após construir uma base tal que (3.1.5) seja satisfeito, o sistema linear $Ax = b$ ainda não foi resolvido. Veremos agora como obter uma solução a partir da base encontrada.

Suponhamos que temos uma aproximação inicial x_0, e o resíduo associado à esta aproximação $r_0 = b - Ax_0$. Utilizaremos um dos métodos descritos de modo a encontrar uma base para o espaço $\mathcal{K}_\ell(A, r_0)$. Caso sejam satisfeitas as condições para existência de uma solução do problema $Au = r_0$, $u \in \mathcal{K}_\ell$, então u poderá ser escrito como $V_\ell y$, $y \in \mathbb{C}^\ell$ e

$$AV_\ell y = V_\ell H_\ell y = r_0 \tag{3.3.1}$$

$$H_\ell y = V_\ell^H r_0. \tag{3.3.2}$$

Se construirmos a base $\{v_1, v_2, \ldots, v_\ell\}$ começando com $v_1 = \frac{r_0}{\|r_0\|_2}$, então $r_0 = \|r_0\|_2 V_\ell e_1^\ell$, e consequentemente

$$H_\ell y = \|r_0\|_2 e_1^\ell. \tag{3.3.3}$$

Este problema é, em geral, muito mais simples de resolver que o problema $Ax = b$, devido à estrutura de H_ℓ, e pelo fato de que ℓ pode ser muito menor que n em alguns

casos. De posse de tal y, teremos

$$b - Ax_0 = r_0 = Au \tag{3.3.4}$$

$$A(x_0 + c) = b, \tag{3.3.5}$$

logo, resolver $Au = r_0$ é equivalente a $Ax = b$. Repare que se $x_0 = 0$, então $x = u$ e $r_0 = b$. Como veremos mais a frente, não necessitamos construir uma base para $\mathcal{K}_\ell$ pois em muitos casos, com $\mathcal{K}_k$, $k < \ell$ já teremos uma solução boa o suficiente.

3.4 Subespaço $\mathcal{W}_k$ e sua Base

Uma definição que será útil no futuro é

$$\mathcal{W}_k = A\mathcal{K}_k \tag{3.4.1}$$

onde $A\mathcal{K}_k$ é o subespaço de Krylov gerado por todos os vetores da sequência de Krylov multiplicados por A, ou seja $A\mathcal{K}_k = \text{span}\{Ab, A^2b, \ldots, A^kb\}$.

Se denotarmos uma matriz $\hat{W}_k$ cujas colunas formam uma base $\mathcal{W}_k$, de (3.1.4), temos

$$\hat{W}_k = V_{k+1}\tilde{H}_k. \tag{3.4.2}$$

Entretanto, vamos tentar encontrar uma base ortogonal para $\mathcal{W}_k$, utilizando a base ortogonal que já temos para $\mathcal{K}_k$. Para isso, consideremos o seguinte teorema

Teorema 3.4.1. *Seja $A = QR$ uma fatoração QR de uma matriz $A \in \mathbb{R}^{m\times n}$ de posto completo, onde $m \geq n$. Consideremos que podemos escrever $A = [a_1, \ldots, a_n]$ e $Q = [q_1, \ldots, q_m]$, onde a_j e q_j representam as colunas de A e Q respectivamente. Então*

$$\text{span}\{a_1, \ldots, a_k\} = \text{span}\{q_1, \ldots, q_k\}, \qquad 1 \leq k \leq n. \tag{3.4.3}$$

Em particular, se $Q_1 = Q(1:m, 1:n)$ e $Q_2 = Q(1:m, n+1:m)$, então

$$\text{img}(A) = \text{img}(Q_1) \tag{3.4.4}$$

$$\text{img}(A^\perp) = \text{img}(Q_2) \tag{3.4.5}$$

e $A = Q_1R_1$ onde $R_1 = R(1:n, 1:n)$

Demonstração. A prova deste teorema pode ser encontrada com detalhes em [4], p.229. □

No decorrer deste trabalho, quando mencionarmos $A = \bar{Q}\bar{R}$ estaremos nos referindo à *fatoração QR econômica*, onde $\bar{Q} \in \mathbb{C}^{m\times n}$ e $\bar{R} \in \mathbb{C}^{n\times n}$. Esta notação também pode ser encontrada em [4].

Com este teorema e esta notação, se obtivermos a fatoração QR econômica $\bar{Q}_k\bar{R}_k$ de $\tilde{H}_k$, teremos

$$\hat{W}_k = \left(V_{k+1}\bar{Q}_k\right)\bar{R}_k, \tag{3.4.6}$$

e como a matriz entre parênteses é uma matriz unitária, então temos a fatoração QR econômica de $\hat{W}_k$. Pelo próprio teorema 3.4.1

$$W_k = V_{k+1}\bar{Q}_k \tag{3.4.7}$$

é uma base ortogonal para $\mathcal{W}_k$.

3.5 Processo de Lanczos Não-Hermitiano

Na tentativa de reduzir o custo da ortogonalização do método de Arnoldi no caso em que A não é simétrica, uma estratégia pode ser a criação de duas bases biortonormais. Sejam as bases $\{v_1, v_2, \ldots, v_k\}$ e $\{z_1, z_2, \ldots, z_k\}$ biortonormais[3] que geram respectivamente $\mathcal{K}_k(A, b)$ e $\mathcal{K}_k(A^H, b)$. Partindo de v_1 e z_1, o que iremos fazer para obter tal base será

$$\tilde{v}_{k+1} = Av_k - \sum_{j=1}^{k} \nu_{(j,k)} v_j, \tag{3.5.1}$$

buscando coeficientes $\nu_{(i,j)}$ tais que $\tilde{v}_{k+1}$ seja ortogonal aos z_j que já temos. Faremos o mesmo para encontrar um $\tilde{z}_{k+1}$, armazenando os coeficientes $\zeta_{(i,j)}$, e por fim encontraremos constantes $\nu_{(k+1,k)}$ e $\zeta_{(k+1,k)}$ tais que

$$\begin{aligned} v_{k+1} &= \frac{\tilde{v}_{k+1}}{\nu_{(k+1,k)}} \\ z_{k+1} &= \frac{\tilde{z}_{k+1}}{\zeta_{(k+1,k)}} \\ \langle v_{k+1}, z_{k+1}\rangle &= 1 \end{aligned} \tag{3.5.2}$$

Com este processo teremos

$$\begin{aligned} AV_k &= V_{k+1}\tilde{H}_k \\ A^H Z_k &= Z_{k+1}\tilde{Y}_k \end{aligned} \tag{3.5.3}$$

assim como no processo de ortogonalização de Arnoldi, onde $\tilde{H}_k, \tilde{Y}_k \in \mathbb{C}^{k+1\times k}$ são matrizes de Hessenberg superior. Repare que o valor dos coeficientes de $\tilde{H}_k$ e $\tilde{Y}_k$ são diferentes aqui.

Como exigimos apenas que $\left\langle v_i, z_j \right\rangle = 0$ para $i < j$ e igual a um para $i = j$, teremos $Z_k^H V_k = L_k$, onde L_k será uma matriz quadrada, triangular inferior, cuja

[3] ou seja, $\langle v_i, z_j\rangle = \delta_{ij}$

diagonal principal é composta apenas por uns. Por outro lado, como exigimos o mesmo dos z_j, também teremos $V_k^H Z_k = J_k$, onde J_k é uma matriz quadrada, triangular inferior e cuja diagonal é composta apenas por uns. Como $(V_k^H Z_k)^H = Z_k^H V_k$ então $L_k = J_k = I_k$, e com isto teremos contruído duas bases biortonormais.

Pré-multiplicando a primeira igualdade de (3.5.3) por Z_k^H teremos

$$Z_k^H A V_k = Z_k^H V_{k+1} \tilde{H}_k = [I_k \quad 0]\tilde{H}_k = H_k \tag{3.5.4}$$

e pré-multiplicando a segunda igualdade por V_k^H teremos

$$V_k^H A^H Z_k = Y_k \tag{3.5.5}$$

$$(V_k^H A^H Z_k)^H = Y_k^H \tag{3.5.6}$$

$$Z_k^H A V_k = Y_k^H = H_k \tag{3.5.7}$$

Isto nos mostra que só precisamos calcular (e armazenar) uma das matrizes, e não ambas. Em segundo lugar, como Y_k e H_k são matrizes de Hessenberg superior, elas serão também tridiagonais. Para facilitar a notação, faremos $\nu_{(j+1,j)} = \nu_{j+1}$, $\zeta_{(j+1,j)} = \zeta_{j+1}$, e $\zeta^*_{(j,j)} = \nu_{(j,j)} = \eta_j$. Assim

$$H_k = \begin{bmatrix} \eta_1 & \zeta_2^* & & & \\ \nu_2 & \eta_2 & \zeta_3^* & & \\ & \ddots & \ddots & \ddots & \\ & & \nu_{k-1} & \eta_{k-1} & \zeta_k^* \\ & & & \nu_k & \eta_k \end{bmatrix} \tag{3.5.8}$$

Conforme foi dito anteriormente, os valores de ζ_j e ν_j devem apenas satisfazer a relação

$$\nu_j \zeta_j = \left\langle \tilde{v_j}, \tilde{z_j} \right\rangle \tag{3.5.9}$$

para que $\left\langle v_j, z_j \right\rangle = 1$, o que nos dá uma certa liberdade. Poderíamos por exemplo tomar $\nu_j = \zeta_j^* = \sqrt{\left\langle \tilde{v_j}, \tilde{z_j} \right\rangle}$ o que faria com que a matriz H_k além de tridiagonal fosse simétrica. Entretanto isso poderia gerar valores pertencente aos complexos, o que poderia ser indesejável quando estamos trabalhando apenas com reais. Por hora, consideraremos esta escolha em aberto, e trataremos apenas do caso geral.

Como já sabemos a estrutura de H_k, também sabemos que os cálculos dos v_j e z_j serão

$$\begin{aligned} \nu_{k+1} v_{k+1} &= A v_k - \eta_k v_k - \zeta_k^* v_{k-1} \\ \zeta_{k+1} z_{k+1} &= A^H z_k - \eta_k^* z_k - \nu_k^* z_{k-1}. \end{aligned} \tag{3.5.10}$$

Pré-multiplicando a primeira igualdade por z_k^H teremos

$$\nu_{k+1} z_k^H v_{k+1} = z_k^H A v_k - \eta_k z_k^H v_k - \zeta_k^* z_k^H v_{k-1} \tag{3.5.11}$$

$$0 = z_k^H A v_k - \eta_k \tag{3.5.12}$$

$$\eta_k = z_k^H A v_k \tag{3.5.13}$$

Basta agora definir v_1 e z_1. Como queremos que os espaços gerados por V_k e Z_k sejam $\mathcal{K}_k(A, b)$ e $\mathcal{K}_k(A^H, b)$ respectivamente, e que $\langle v_1, z_1 \rangle = 1$, uma idéia simples seria

$$v_1 = z_1 = \frac{b}{\|b\|_2} \tag{3.5.14}$$

Mostramos no algoritmo 3 um pseudocódigo para este método.

Algoritmo 3: Processo de BiOrtogonalização de Lanczos Não-Hermitiano

Input: A, b

1 $v_1 = z_1 = \dfrac{b}{\|b\|_2}$;
2 $\zeta_0 = \nu_0 = 0$;
3 **for** $j = 1...$ **do**
4 $\quad v_{j+1} = A v_j$;
5 $\quad z_{j+1} = A^H z_j$;
6 $\quad \eta_j = \left\langle z_j, v_{j+1} \right\rangle$;
7 $\quad v_{j+1} = v_{j+1} - \eta_j v_j - \zeta_j^* v_{j-1}$;
8 $\quad z_{j+1} = z_{j+1} - \eta_j^* z_j - \nu_j^* z_{j-1}$;
9 $\quad \nu_{j+1}\zeta_{j+1} = \left\langle v_{j+1}, z_{j+1} \right\rangle$;
10 $\quad$ **if** $\left(\nu_{j+1}\zeta_{j+1} = 0\right)$ **then** Stop;
11 $\quad v_{j+1} = \dfrac{v_{j+1}}{\nu_{j+1}}$;
12 $\quad z_{j+1} = \dfrac{z_{j+1}}{\zeta_{j+1}}$;
13 **end for**

Repare que não precisamos de fato armazenar Z_k para obter a base V_k, apenas os dois últimos z_j. Infelizmente a base gerada por este método não é ortonormal, e algumas propriedades interessantes vistas na seção 3.3 não são válidas para bases biortogonais. Ainda assim é um meio interessante de encontrar uma base para o espaço de Krylov, devido ao custo reduzido.

Algo importante a se notar é que, caso o produto interno na linha 9 seja igual a zero, a única solução possível é que $\nu_{j+1} = 0$ ou $\zeta_{j+1} = 0$. Com isso teríamos uma divisão por zero no algoritmo. Chamamos esse comportamento de *ruptura*. Existem alguns meios de evitar rupturas, como por exemplo uma técnica chamada "look-ahead". Em [6] podemos encontrar uma explicação melhor a respeito das rupturas,

como evitá-las e remediá-las. Também recomendamos o relatório técnico [7] para um estudo a respeito de rupturas no método de Lanczos.

Capítulo 4

Métodos para Solução de Sistemas Lineares

Neste capítulo utilizaremos os conceitos e resultados obtidos nos capítulos anteriores em busca de um método para solução do sistema linear $Ax = b$. Esta não é uma tarefa fácil pois embora alguns métodos encontrem uma solução num número mínimo de passos, eles apresentam outros tipos de dificuldades, como a necessidade de uma grande quantidade de memória disponível para o armazenamento de dados, ou custo crescente de iterações (a cada iteração o algoritmo fica mais complexo), o que os torna desaconselháveis na prática.

Como trabalharemos com métodos iterativos, então a cada iteração teremos uma aproximação nova para a solução, e esperamos que a cada iteração estejamos "melhorando" esta aproximação até o ponto em que ela seja uma aproximação aceitável para o nosso problema. A *aproximação da solução* da k-ésima iteração será denotada por x_k. Em muitos métodos atualizaremos esta aproximação fazendo $x_k = x_0 + c_k$, onde c_k será chamado de *correção*.

Podemos medir quão boa é uma aproximação utilizando o conceito de *resíduo* da iteração. O resíduo, que denotaremos por r_k nada mais é do que $b - Ax_k$. Entretanto, na maioria dos métodos encontraremos meios mais eficazes de calcular o resíduo (além disso, em alguns métodos nem mesmo seremos capazes de calcular x_k e portanto seremos obrigados a calcular o resíduo de uma forma alternativa). Devido à problemas de aproximação e de precisão, muitas vezes o resíduo r_k calculado será diferente de $b - Ax_k$. Por isso chamaremos $b - Ax_k$ de *resíduo real* quando se fizer necessário diferenciar.

Por motivos de simplificação, no desenvolver de todos estes métodos suporemos A não-singular.

4.1 FOM - Método da Ortogonalização Completa

Começamos apresentando o Método da Ortogonalização Completa (*Full Orthogonalization Method*, ou simplesmente FOM), aprensentado em [8] por Saad. Como veremos mais à frente, este método consiste basicamente do processo de ortogonalização de Arnoldi com a solução de um sistema linear simples, e por isso também é conhecido como *Método de Arnoldi*.

Neste método buscaremos a cada iteração uma correção $c_k \in \mathcal{K}_k(A, r_0)$, de modo que

$$b - Ax_k \perp \mathcal{K}_k, \tag{4.1.1}$$

o que é chamado de *condição de Galerkin*.

Podemos entender o objetivo desta condição da seguinte forma: se forçarmos o resíduo a ser sempre ortogonal, em algum momento ele será zero, e com isso $0 = b - Ax_k$ o que significa que encontramos a solução.

Aqui exigimos que $c_k \in \mathcal{K}_k$ pois como vimos na seção 3.3, se encontrarmos a solução de $Ac = r_0$ então teremos encontrado a solução do problema original. Para isso começamos com o vetor $v_1 = r_0/\|r_0\|_2$ e a cada iteração geramos um novo vetor v_{k+1} através de uma iteração do processo de Arnoldi descrito na seção 3.1, obtendo assim uma base para $\mathcal{K}_{k+1}(A, r_0)$.

Seguindo os resultados da seção 3.4, da condição de Galerkin (4.1.1) teremos

$$b - Ax_0 - Ac_k = r_0 - w_k \perp \mathcal{K}_k, \tag{4.1.2}$$

$w_k \in \mathcal{W}_k$ e portanto

$$w_k = \hat{W}_k y_k = V_{k+1}\tilde{H}_k y_k, \tag{4.1.3}$$

onde $y_k \in \mathbb{C}^k$. Conforme já foi mostrado na seção 3.3, como começamos com $v_1 = r_0/\|r_0\|_2$, então

$$r_0 = V_{k+1}\|r_0\|_2 \, e_1^{k+1} \tag{4.1.4}$$

e portanto

$$V_{k+1}(\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y_k) \perp \mathcal{K}_k. \tag{4.1.5}$$

Basta observarmos aqui que o vetor v_{k+1} é ortogonal ao espaço $\mathcal{K}_k$ por construção. Caso consigamos y_k tal que $\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y_k$ seja um múltiplo de e_{k+1}^{k+1} então teremos a condição de Galerkin satisfeita. Isto é

$$\|r_0\|_2 \begin{bmatrix} e_1^k \\ 0 \end{bmatrix} - \begin{bmatrix} H^k \\ \eta(e_k^k)^T \end{bmatrix} y_k = \alpha e_{k+1}^{k+1} \tag{4.1.6}$$

para algum $\alpha \in \mathbb{C}$, o que é equivalente a

$$\|r_0\|_2 e_1^k - H_k y_k = 0. \tag{4.1.7}$$

Infelizmente, nada garante a existência de uma solução $y_k = \|r_0\|_2 H_k^{-1} e_1^k$, o que pode levar este método a rupturas. No capítulo 5 veremos uma interpretação diferente deste método, e também estudaremos mais a fundo a existência ou não de uma solução para o sistema.

Supondo que não houve rupturas e que dispomos de y_k, obtemos $w_k = V_{k+1}\tilde{H}_k y_k = AV_k y_k$ e consequentemente $c_k = A^{-1}w_k = V_k y_k$.

A solução do sistema (4.1.7) faz com que o custo do algoritmo cresça a cada iteração. Por isso, foi proposto em [2] que este sistema fosse resolvido apenas após a convergência do algoritmo. Como não podemos calcular r_k nem x_k sem y_k, um critério de convergência seria $\eta_{k+1,k} = \|v_{k+1}\|_2 = 0$. Pela propriedade 2.1.3 isto ocorrerá na iteração ℓ onde ℓ é o grau do polinômio mínimo de r_0 com relação a A, e consequentemente já podemos obter a solução exata do problema.

A desvantagem deste critério de parada é que em muitos casos o algoritmo já encontra uma aproximação boa o suficiente antes da convergência exata. Além disso, os passos extras contribuem para o crescimento da matriz H_ℓ o que torna a solução do problema (4.1.7) mais complicada do que seria caso o algoritmo tivesse parado ao encontrar uma boa aproximação.

O algoritmo 4 mostra um pseudocódigo deste método conforme mostrado em [2]. Repare que o algoritmo trata-se basicamente do processo de Arnoldi (ver algoritmo 1) apenas com a adição da solução do sistema (4.1.7).

Algoritmo 4: FOM - Método da Ortogonalização Completa

Input: A, b, x_0
$r_0 = b - Ax_0$;
$v_1 = r_0/\|r_0\|_2$;
for $j = 1$ *to* n **do**
  $v_{j+1} = Av_j$;
  for $i = 1$ *to* j **do**
    $\eta_{i,j} = \langle v_i, v_{j+1} \rangle$;
    $v_{j+1} = v_{j+1} - \eta_{i,j} v_i$;
  end for
  $\eta_{j+1,j} = \|v_{j+1}\|_2$;
  $v_{j+1} = v_{j+1}/\eta_{j+1,j}$;
  if $\eta_{j+1,j} = 0$ **then** Goto 13
end for
$y = \|r_0\|_2 H_j^{-1} e_1^j$;
$x = V_j y$;

Além do critério de convergência, o FOM possui mais uma desvantagem. Suponhamos que a matriz A seja muito grande, mas que conta com a vantagem de ser esparsa, permitindo o seu armazenamento. Nada garante que a matriz V_k seja também esparsa, e esta cresce com o número de iterações, o que pode ser impraticável em termos de armazenamento e manipulação.

Além disso, como o FOM gera uma base para o espaço de Krylov usando o processo de ortogonalização de Arnoldi, precisaremos ortogonalizar cada v_{k+1} com todos os vetores anteriores, e com isso o custo da ortogonalização também aumenta com o número de iterações, tornando o FOM inviável em muitos casos. Além disso, em muitos casos a solução aproximada já possui um resíduo "aceitável", e portanto o algoritmo nem sempre precisa de ℓ passos para atingir a convergência. Como não calculamos o resíduo no FOM, não hão meios de saber se o algoritmo já pode parar.

Assim vemos que, embora o FOM garanta a convergência em um número finito de passos, nem sempre será um método praticável.

4.2 GMRES - Resíduos Mínimos Generalizados

Apresentamos agora um dos métodos mais populares atualmente, o método do Resíduo Mínimo Generalizado (Generalized Minimal Residual, de onde vem GMRES) criado por Saad e Schultz em [9].

Assim como no FOM, no GMRES iremos construir a cada iteração k um novo vetor v_{k+1} utilizando uma iteração do processo de ortogonalização de Arnoldi, produzindo assim uma base para o espaço $\mathcal{K}_{k+1}(A, r_0)$.

Então, a cada iteração iremos buscar uma correção c_k de modo que o resíduo seja minimizado, ou seja

$$r_k = b - Ax_k = b - Ax_0 - Ac_k = r_0 - Ac_k \tag{4.2.1}$$

$$c_k = \arg\min_{c \in \mathcal{K}_k} \|r_0 - Ac\|_2 , \tag{4.2.2}$$

e com a notação da seção 3.4 simplificamos o problema para

$$w_k = \arg\min_{w \in \mathcal{W}_k} \|r_0 - w\|_2 . \tag{4.2.3}$$

De forma similar ao que foi feito no FOM, da seção 3.3, teremos

$$\min_{w \in \mathcal{W}_k} \|r_0 - w_k\| = \min_{y \in \mathbb{C}^n} \left\| \|r_0\|_2 V_{k+1} e_1^{k+1} - V_{k+1} \tilde{H}_k y \right\| \tag{4.2.4}$$

$$= \min_{y \in \mathbb{C}^n} \left\| V_{k+1} \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right) \right\| \tag{4.2.5}$$

e como V_{k+1} é unitário

$$\min_{y\in\mathbb{C}^n}\left\|\|r_0\|_2\, e_1^{k+1} - \tilde{H}_k y\right\| \tag{4.2.6}$$

Com isso, a solução do problema $\min\|r_k\|$ se reduz à solução do problema de quadrados mínimos

$$\tilde{H}_k y_k = \|r_0\|_2\, e_1^{k+1}. \tag{4.2.7}$$

Supondo que temos a disposição a fatoração QR econômica de $\tilde{H}_k$, teremos

$$\bar{Q}_k\bar{R}_k y_k = \|r_0\|_2\, e_1^{k+1} \tag{4.2.8}$$

$$y_k = \bar{R}_k^{-1}\bar{Q}_k^H\|r_0\|_2\, e_1^{k+1}, \tag{4.2.9}$$

e de posse de y_k encontramos $w_k = AV_k y_k$ e $c_k = V_k y_k$. Repare que para obter y_k precisamos de $\bar{R}_k^{-1}$, ou seja, precisamos resolver um sistema triangular a cada iteração para obter x_k.

Mostraremos agora uma outra formulação do problema. Partindo de (4.2.3), o teorema abaixo mostra como encontrar w_k tal que a norma seja minimizada:

Teorema 4.2.1 (Teorema do ponto mais próximo)**.** *Seja $\mathcal{M}$ um subespaço de $\mathcal{H}$, e seja h um vetor em $\mathcal{H}$. O vetor em $\mathcal{M}$ mais próximo de h é a a projeção ortogonal de h em $\mathcal{M}$, que é única. Em outras palavras*

$$\min_{m\in\mathcal{M}}\|h-m\|_2 = \|h - P_{\mathcal{M}}h\|_2 \tag{4.2.10}$$

Demonstração. A prova deste teorema pode ser encontrada com detalhes em [1], p.435. □

Tendo este resultado, basta construir a matriz de projeção em $\mathcal{W}_k$ e teremos w_k. Considerando novamente que dispomos da fatoração QR econômica de $\tilde{H}_k$, de (3.4.7) teremos

$$w_k = W_k W_k^H r_0 = V_{k+1}\bar{Q}_k\bar{Q}_k^H V_{k+1}^H r_0. \tag{4.2.11}$$

Como $v_1 = r_0/\|r_0\|_2$, então $r_0 = \|r_0\|_2\, V_k e_1^k$, logo

$$w_k = V_{k+1}\bar{Q}_k\bar{Q}_k^H\|r_0\|_2\, e_1^k \tag{4.2.12}$$

Mas

$$AV_k = V_{k+1}\bar{Q}_k\bar{R}_k \tag{4.2.13}$$

$$AV_k\bar{R}_k^{-1} = V_{k+1}\bar{Q}_k, \tag{4.2.14}$$

e portanto

$$w_k = AV_k \bar{R}_k^{-1} \bar{Q}_k^H \|r_0\|_2 e_1^k = AV_k y_k, \tag{4.2.15}$$

ou seja, ambos as formulações levam à mesma solução.

Apesar de o sistema triangular a ser resolvido em cada iteração ter um tamanho modesto, ele cresce com o número de iterações e portanto, na maioria das implementações deste método, x_k só será calculado após a convergência do algoritmo. Para ter uma estimativa da convergência, podemos, dentre outras coisas, verificar a norma do resíduo r_k.

O cálculo de $r_k = r_0 - w_k$ pode ser significativamente simplificado pois necessita apenas da matriz V_{k+1}, que é calculada com o método de Arnoldi, e da matriz $\bar{Q}_k$ obtida da decomposição QR econômica de $\tilde{H}_k$. Embora a decomposição QR possa ser um procedimento caro, neste caso podemos fazê-lo com um custo muito baixo. Para isso, aplicaremos k rotações de Givens na matriz $\tilde{H}_k$ de modo a zerar os valores abaixo da diagonal principal, ou seja

$$G_1 G_2 \dots G_k \tilde{H}_k = R_k, \tag{4.2.16}$$

onde as matrizes G_j representam as matrizes de Givens. Como $Q_k^H = G_1 G_2 \dots G_k \in \mathbb{C}^{k+1 \times k+1}$ é unitário (pois é um produto de matrizes de Givens, que são unitárias) e $R_k \in \mathbb{C}^{k+1 \times k}$ é uma matriz triangular superior cuja última linha contém apenas zeros, podemos interpretar este resultado como uma fatoração QR completa de $\tilde{H}_k$. De fato podemos relacionar ambas fatorações QR obtendo

$$R_k = \begin{bmatrix} \bar{R}_k \\ 0 \end{bmatrix} \quad \text{e} \quad Q_k = \begin{bmatrix} \bar{Q}_k & q_{k+1} \end{bmatrix}, \tag{4.2.17}$$

onde $q_{k+1} \in \mathbb{C}^{k+1}$ é a $k+1$-ésima coluna de Q_k

Como Q_k é unitária, podemos simplificar a equação (4.2.6) para

$$\min_{y \in \mathbb{C}^n} \left\| Q_k^H (\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y) \right\| = \min_{y \in \mathbb{C}^n} \|g_k - R_k y\|, \tag{4.2.18}$$

para $g_k = \|r_0\|_2 Q_k^H e_1^{k+1}$, o que mostra que y_k é a solução do sistema $R_k y = g_k$. Ora, este problema é o mesmo que

$$\begin{bmatrix} \bar{R}_k \bar{y} \\ 0 \times \psi_{k+1} \end{bmatrix} = \begin{bmatrix} \bar{g}_k \\ \gamma_{k+1} \end{bmatrix} \tag{4.2.19}$$

onde ψ_{k+1} e γ_{k+1} são o último elemento de y e g_k respectivamente, e $\bar{y}, \bar{g}_k \in \mathbb{C}^k$. Podemos simplesmente definir $\bar{y} = \bar{R}_k^{-1} \bar{g}_k$, mas nunca poderemos encontrar $0 \times$

$\psi_{k+1} = \gamma_{k+1}$ a menos que $\gamma_{k+1} = 0$. Reparemos que a norma de $g_k - R_k y$ é na verdade a norma do resíduo, e como conseguimos sempre zerar os k primeiros elementos desse vetor, a norma será constituida apenas por γ_{k+1}.

Porfim, teremos também que

$$\tilde{H}_{k+1} = \begin{bmatrix} \tilde{H}_k & h_{k+1} \\ 0 & \eta_{(k+2,k+1)} \end{bmatrix} \in \mathbb{C}^{k+2 \times k+1} \tag{4.2.20}$$

onde $h_{k+1}^T = [\eta_{(1,k+1)}, \eta_{(2,k+1)}, \ldots, \eta_{(k+1,k+1)}]$. Portanto, a cada iteração só precisaremos zerar um elemento de $\tilde{H}_k$ para obter R_k o que custa a aplicação de apenas uma rotação de Givens por iteração.

Se utilizarmos tais rotações de Givens para atualizarmos o vetor g_k a cada iteração, portanto, teremos a norma do resíduo sem nenhuma operação adicional. Repare que não há a necessidade de calcularmos a aproximação da solução e que como dispomos da norma do resíduo, podemos parar o algoritmo quando esta for suficientemente pequena, e portanto não precisamos de fato executar ℓ passos.

O algoritmo 5 apresenta um pseudocódigo para o algoritmo do GMRES.

Algoritmo 5: GMRES - Resíduos Mínimos Generalizado

```
A, b, x_0;
r_0 = b − Ax_0;
β = ‖r_0‖_2;
v_1 = r_0/β;
j = 1; while ‖r_j‖_2 > tol do
    w = Av_j;
    for i = 1 to j do
        h_{i,j} = ⟨v_i, w_i⟩;
        w = w − h_{i,j} v_i;
    end for
    h_{j+1,j} = ‖w‖_2;
    v_{j+1} = w / h_{j+1,j};
    atualiza g_j = G_j g_{j−1};
end while
H̃_j y = β_2 e_1^{j+1};
x = x_0 + V_j y;
r = r_0 − V_{j+1} H̃_j y;
```

Apesar de ser um método comprovadamente estável (cf. [10] e [11]) e por evitar vários cálculos desnecessários, o GMRES ainda apresenta um problema. Por utilizar o processo de ortogonalização de Arnoldi, o GMRES compartilha os mesmos problemas do FOM, ou seja, V_k pode não ser esparso e cresce com o número de iterações, dificultando seu armazenamento e a ortogonalização nas iterações posteriores.

4.3 CG - Gradientes Conjugados

O metódo dos Gradientes Conjugados [12] foi criado por Hestenes e Stiefel em 1952 praticamente ao mesmo tempo em que Lanczos publicou o seu método em [13], que é considerado equivalente. Na verdade, em [6] é dito que *"O método dos Gradientes Conjugados não é nada mais que um caso particular do método de Lanczos"*. Devido à sua importância histórica, mostraremos aqui apenas o método dos Gradientes Conjugados (que notaremos por CG, de *Conjugate Gradients*). Seguiremos aqui o mesmo raciocínio exposto por Shewchuk em [14].

Este método foi criado para a solução de sistemas em que A é hermitiana e positiva definida além de não-singular. Durante toda a demonstração deste método suporemos A hermitiana definida positiva. Uma definição importante aqui será $e_k = x_k - A^{-1}b$, que chamamos de *erro*. Repare que $Ae_k = Ax_k - b = -r_k$, o que será uma relação importante para este método.

Suponha que dispomos de um conjunto de vetores $p_0, p_1, \ldots, p_{n-1}$ ortogonais (lembrando que $b \in \mathbb{C}^n$), onde $p_i \in \mathbb{C}^n$. Chamaremos estes vetores de *direções de busca*. A princípio seria possível escrever

$$e_0 = x_0 - A^{-1}b = -\sum_{i=0}^{n-1} \alpha_i p_i \tag{4.3.1}$$

Baseado nesta possibilidade e supondo que conhecemos os α_j, a cada iteração k daremos "um passo α_k" na direção de busca p_k, obtendo

$$x_{k+1} = x_0 + \sum_{i=0}^{k} \alpha_i p_i = x_k + \alpha_k p_k. \tag{4.3.2}$$

Vamos encontrar agora o coeficiente α_k, que define o tamanho do passo a ser dado na direção p_k. Primeiramente, de (4.3.1) e de (4.3.2) temos

$$e_k = x_k - A^{-1}b = \sum_{i=1}^{k-1} \alpha_i p_i + (x_0 - A^{-1}b) \tag{4.3.3}$$

$$= \sum_{i=1}^{k-1} \alpha_i p_i - \sum_{i=1}^{n-1} \alpha_i p_i = -\sum_{i=k}^{n-1} \alpha_i p_i, \tag{4.3.4}$$

o que mostra que $e_k \perp p_j$, para $j < k$. Para facilitar o cálculo de α_j vamos escrever o erro da seguinte forma:

$$e_{k+1} = e_0 + \sum_{i=1}^{k} \alpha_i p_i = e_k + \alpha_k p_k. \tag{4.3.5}$$

Forçarmos a condição de ortogonalidade $e_{k+1} \perp p_k$ teremos

$$0 = \langle p_k, e_{k+1} \rangle = \langle p_k, e_k \rangle + \alpha_k \langle p_k, p_k \rangle \tag{4.3.6}$$

$$\alpha_k = -\frac{\langle p_k, e_k \rangle}{\langle p_k, p_k \rangle} \tag{4.3.7}$$

A desvantagem desta idéia é óbvia: não dispomos de e_k, pois caso tivéssemos teríamos resolvido o problema fazendo $x_k - e_k = A^{-1}b$. Uma solução seria exigir uma *A-ortogonalidade* das direções de busca, ou seja,

$$\left\langle p_i, Ap_j \right\rangle = 0, \qquad i \neq j. \tag{4.3.8}$$

Isto faz com que o erro seja A-ortogonal às direções de busca, ao invés da ortogonalidade que mostramos em (4.3.4). Com isto

$$\alpha_k = -\frac{\langle p_k, Ae_k \rangle}{\langle p_k, Ap_k \rangle} = \frac{\langle p_k, r_k \rangle}{\langle p_k, Ap_k \rangle}, \tag{4.3.9}$$

o que pode ser calculado, pois podemos obter o resíduo fazendo

$$r_{k+1} = -Ae_{k+1} = -A(e_k + \alpha_k p_k) = r_k - \alpha_k A p_k \tag{4.3.10}$$

$$= r_0 - \sum_{i=0}^{k} \alpha_i A p_i \tag{4.3.11}$$

Repare entretanto que, caso A seja indefinida, Ap_k pode ser zero o que impediria o cálculo de α_k.

Vamos agora mostrar como encontrar o conjuto de direções de busca de modo que sejam mutuamente A-ortogonais. Uma idéia intuitiva seria, partir de um conjunto de vetores ortogonais de fácil obtenção, como a base canônica, definir $p_k = e_k^n$ e então A-ortogonalizar p_k com os vetores anteriores. Entretanto, se tomarmos os resíduos ao invés da base canônica, ou seja, se fizermos

$$p_k = r_k + \sum_{i=0}^{k-1} \beta_{(k,i)} p_i \tag{4.3.12}$$

para algum conjunto de $\beta_{(k,j)}$, começando com $p_0 = r_0$, teremos uma facilidade muito grande para o cálculo dos $\beta_{(k,j)}$, como mostraremos a seguir. Primeiramente, consideremos duas propriedades do resíduo.

Propriedade 4.3.1. *No CG o resíduo é ortogonal às direções de busca anteriores.*

Demonstração. Multiplicando (4.3.4) por $-A$ teremos uma expressão para r_k em

função das direções de busca p_i:

$$r_k = \sum_{i=k}^{n-1} \alpha_i A p_i \tag{4.3.13}$$

Portanto, fazendo o produto interno entre r_k e qualquer p_j onde $j < k$ teremos

$$\left\langle p_j, r_k \right\rangle = \left\langle p_j, \sum_{i=k}^{n-1} \alpha_i A p_i \right\rangle = \sum_{i=k}^{n-1} \alpha_i \left\langle p_j, A p_i \right\rangle = 0 \tag{4.3.14}$$

provando que o resíduo é ortogonal às direções de busca anteriores. □

Propriedade 4.3.2. *Os resíduos são mutuamente ortogonais*

Demonstração. Vamos fazer o produto interno entre p_j e r_r, que já sabemos que é zero para todo $j < k$

$$0 = \left\langle p_j, r_k \right\rangle = \left\langle r_j + \sum_{i=0}^{j-1} \beta_{j,i} p_i, r_k \right\rangle = \left\langle r_j, r_k \right\rangle + \sum_{i=0}^{j-1} \beta_{j,i} \left\langle p_i, r_k \right\rangle \tag{4.3.15}$$

Da propriedade 4.3.1 eliminaremos os $\langle p_i, r_k \rangle$, e ficaremos com

$$\left\langle r_j, r_k \right\rangle = 0. \tag{4.3.16}$$

Como $\left\langle r_j, r_k \right\rangle = \left\langle r_k, r_j \right\rangle = 0$, então temos que isto também será verdade para $j > k$, ou seja, $\left\langle r_k, r_j \right\rangle = 0$ para $j \neq k$. □

De posse dessas propriedades, vamos calcular os $\beta_{k,j}$. Faremos o produto interno entre p_k e Ap_j (com $j < k$). De (4.3.12) obtemos

$$\left\langle p_k, A p_j \right\rangle = \left\langle r_k, A p_j \right\rangle + \sum_{i=0}^{k-1} \beta_{(k,i)} \left\langle p_i, A p_j \right\rangle \tag{4.3.17}$$

$$0 = \left\langle r_k, A p_j \right\rangle + \beta_{(k,j)} \left\langle r_j, A p_j \right\rangle \tag{4.3.18}$$

$$\beta_{(k,j)} = -\frac{\left\langle r_k, A p_j \right\rangle}{\left\langle p_j, A p_j \right\rangle} \tag{4.3.19}$$

Agora vamos finalmente mostrar que o cálculo dos $\beta_{(k,j)}$ pode ser significativamente simplificado. De (4.3.10) temos

$$\left\langle r_k, r_{j+1} \right\rangle = \left\langle r_k, r_j \right\rangle - \alpha_j \left\langle r_k, A p_j \right\rangle \tag{4.3.20}$$

$$\alpha_j \left\langle r_k, A p_j \right\rangle = \left\langle r_k, r_j \right\rangle - \left\langle r_k, r_{j+1} \right\rangle \tag{4.3.21}$$

Pela ortogonalidade mostrada na propriedade 4.3.1 concluímos que, se $j = k$

$$\langle r_k, Ap_k \rangle = \frac{\langle r_k, r_k \rangle - \langle r_k, r_{k+1} \rangle}{\alpha_k} = \frac{\langle r_k, r_k \rangle}{\alpha_k}. \tag{4.3.22}$$

e que caso $j = k - 1$, teremos

$$\langle r_k, Ap_{k-1} \rangle = \frac{\langle r_k, r_{k-1} \rangle - \langle r_k, r_k \rangle}{\alpha_{k-1}} = -\frac{\langle r_k, r_k \rangle}{\alpha_{k-1}}. \tag{4.3.23}$$

e todos os demais $\left\langle r_k, Ap_j \right\rangle$ serão zero. Como o cálculo de $\beta_{(k,j)}$ só é efetuado para $j < k$, então $\beta_{(k,j)}$ só será diferente de zero para $j = k-1$. Para simplificar a notação faremos $\beta_{(k,k-1)} = \beta_k$. Assim,

$$p_k = r_k + \beta_k p_{k-1}. \tag{4.3.24}$$

Podemos simplificar ainda mais o cálculo de β_k:

$$\beta_k = -\frac{\langle r_k, Ap_{k-1} \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} = \frac{1}{\alpha_{k-1}} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} \tag{4.3.25}$$

$$= \frac{\langle p_{k-1}, Ap_{k-1} \rangle}{\langle p_{k-1}, r_{k-1} \rangle} \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, Ap_{k-1} \rangle} = \frac{\langle r_k, r_k \rangle}{\langle p_{k-1}, r_{k-1} \rangle} \tag{4.3.26}$$

Mas

$$\langle p_k, r_k \rangle = \langle r_k, r_k \rangle + \beta_k \langle p_{k-1}, r_k \rangle = \langle r_k, r_k \rangle \tag{4.3.27}$$

então

$$\beta_k = \frac{\langle r_k, r_k \rangle}{\langle r_{k-1}, r_{k-1} \rangle} = \frac{\|r_k\|_2^2}{\|r_{k-1}\|_2^2} \tag{4.3.28}$$

e

$$\alpha_k = \frac{\langle r_k, r_k \rangle}{\langle p_k, Ap_k \rangle} = \frac{\|r_k\|_2^2}{\langle p_k, Ap_k \rangle}. \tag{4.3.29}$$

Um pseudocódigo completo para o CG pode ser encontrado no algoritmo 6. Na prática só precisaremos de α, x, r, β e p, pois todos estes vetores podem ser sobrescritos, provando que trata-se de um método muito econômico.

4.4 Reinterpretando o CG

Na seção anterior descrevemos o CG mas não ficou muito clara a ligação entre este método e os demais métodos já apresentados. Isto porque o CG baseia-se em outros métodos como o método do Passo Descendente e o método das Direções Conjugadas,

Algoritmo 6: CG - Gradientes Conjugados

Input: A, b, x_0

$r_0 = b - Ax_0$;
$p_0 = r_0$;
while $\|r_i\|_2 > tol$ **do**
  $\alpha_i = \frac{\|r_i\|_2^2}{\langle p_i, Ap_i \rangle}$;
  $x_{i+1} = x_i + \alpha_i p_i$;
  $r_{i+1} = r_i - \alpha_i A p_i$;
  $\beta_{i+1} = \frac{\|r_{i+1}\|_2^2}{\|r_i\|_2^2}$;
  $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$;
end while

cuja estratégia era minimizar a forma quadrática

$$f(x) = \frac{1}{2} x^T A x - b^T x + c \tag{4.4.1}$$

para A simétrica e definida positiva, cuja única solução é dada por $Ax = b$. Estabeleceremos agora as ligações necessárias para mostrar como o CG está relacionado aos demais métodos. Para facilitar a leitura, iremos repetir aqui algumas das equações da seção anterior.

Primeiramente devemos lembrar que cada direção de busca é uma combinação linear dos k primeiros resíduos, ou seja

$$p_k = r_k + \beta_k p_{k-1} \tag{4.4.2}$$

$$p_k = r_k + \beta_k r_{k-1} + \beta_k \beta_{k-1} p_{k-2} \tag{4.4.3}$$

$$\vdots \tag{4.4.4}$$

$$p_k = r_k + \sum_{j=0}^{k-1} \left(\left(\prod_{i=j+1}^{k} \beta_i \right) r_j \right) \tag{4.4.5}$$

Para simplificar a notação, e consequentemente a visualização, substituiremos

$$\hat{\beta}_{(k,j)} = \prod_{i=j+1}^{k} \beta_i, \tag{4.4.6}$$

onde $\hat{\beta}_{(k,k)} = 1$, obtendo

$$p_k = \sum_{j=0}^{k} \hat{\beta}_{(k,j)} r_j. \tag{4.4.7}$$

Por outro lado, o resíduo será

$$r_{k+1} = r_k - \alpha_k A p_k = r_0 - \sum_{j=0}^{k} \alpha_j A p_j \tag{4.4.8}$$

$$= r_0 - \sum_{j=0}^{k} \left(\alpha_j A \sum_{i=0}^{j} \hat{\beta}_{(j,i)} r_i \right) \tag{4.4.9}$$

$$= r_0 - \sum_{j=0}^{k} \sum_{i=0}^{j} \alpha_j \hat{\beta}_{(j,i)} A r_i \tag{4.4.10}$$

Mais uma vez simplificando a notação, teremos

$$r_{k+1} = r_0 - \sum_{j=0}^{k} \hat{\alpha}_{(k,j)} A r_j \tag{4.4.11}$$

para algum conjunto $\hat{\alpha}_{(k,j)}$. Ainda temos que

$$r_{k+1} = r_0 - \sum_{j=0}^{k} \hat{\alpha}_{(k,j)} A \left(r_0 - \sum_{i=0}^{j-1} \hat{\alpha}_{(j,i)} A r_i \right) \tag{4.4.12}$$

$$= r_0 - \sum_{j=0}^{k} \hat{\alpha}_{(k,j)} A r_0 - \sum_{j=0}^{k} \hat{\alpha}_{(k,j)} A \sum_{i=0}^{j-1} \hat{\alpha}_{(j,i)} A r_i \tag{4.4.13}$$

$$= r_0 + \gamma_1 A r_0 - \sum_{j=0}^{k} \sum_{i=0}^{j-1} \hat{\alpha}_{(k,j)} \hat{\alpha}_{(j,i)} A^2 r_i \tag{4.4.14}$$

$$= r_0 + \gamma_1 A r_0 - \sum_{j=0}^{k} \sum_{i=0}^{j-1} \hat{\alpha}_{(k,j)} \hat{\alpha}_{(j,i)} A^2 r_i \tag{4.4.15}$$

Repetindo este processo, obteremos

$$r_{k+1} = r_0 + \gamma_1 A r_0 + \gamma_2 A^2 r_0 + \cdots + \gamma_k A^k r_0 = \sum_{j=0}^{k} \gamma_j A^j r_0 \tag{4.4.16}$$

onde $\gamma_0 = 1$.

Formalizamos este resultado na propriedade abaixo.

Propriedade 4.4.1. *No CG, os resíduos r_k , e consequentemente as direções de busca p_k, pertencem ao subespaço de Krylov $\mathcal{K}_{k+1}(A, r_0)$.*

Com esta propriedade e com a propriedade 4.3.2 verificamos que os r_j formam uma base ortogonal (não ortonormal) para $\mathcal{K}_k$. Se denotarmos $\mathbb{C}^{n\times k} \ni R_k = [r_0 \ \ldots \ r_{k-1}]$, então

$$V_k = R_k N_k^{-1} \tag{4.4.17}$$

$$N_k = \text{diag}\left(\|r_0\|_2 , \|r_1\|_2 , \ldots , \|r_{k-1}\|_2 \right) \tag{4.4.18}$$

onde $V_k \in \mathbb{C}^{n\times k}$ é uma matriz unitária cujas colunas formam uma base ortonormal para $\mathcal{K}_k$.

Seja $P_k \in \mathbb{C}^{n\times k}$ a matriz cujas colunas são as direções de busca $p_0, p_1, \ldots, p_{k-1}$. Da equação (4.4.7) temos que

$$P_k = R_k \begin{bmatrix} 1 & \hat{\beta}_{(1,0)} & \hat{\beta}_{(2,0)} & \cdots & \hat{\beta}_{(k-1,0)} \\ & 1 & \hat{\beta}_{(2,1)} & \cdots & \hat{\beta}_{(k-1,1)} \\ & & 1 & \cdots & \hat{\beta}_{(k-1,1)} \\ & & & \ddots & \vdots \\ & & & & 1 \end{bmatrix} = R_k U_k^{-1} \tag{4.4.19}$$

Embora a notação $U_k^{-1} \in \mathbb{C}^{k\times k}$ possa parecer estranha, ela faz sentido quando observamos que, tomando

$$U_k = \begin{bmatrix} 1 & -\beta_1 & & & \\ & 1 & -\beta_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & -\beta_{k-1} \\ & & & & 1 \end{bmatrix} \tag{4.4.20}$$

teremos $U_k U_k^{-1} = I_k$.

Continuando, de (4.4.8) teremos

$$Ap_{k-1} = \frac{r_{k-1} - r_k}{\alpha_{k-1}} = R_{k+1} \begin{bmatrix} 0 \\ \vdots \\ \frac{1}{\alpha_{k-1}} \\ -\frac{1}{\alpha_{k-1}} \end{bmatrix} \tag{4.4.21}$$

$$AP_k = R_{k+1} \begin{bmatrix} \frac{1}{\alpha_0} & & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} & & & \\ & -\frac{1}{\alpha_1} & \ddots & & \\ & & \ddots & \frac{1}{\alpha_{k-1}} \\ & & & -\frac{1}{\alpha_{k-1}} \end{bmatrix} \tag{4.4.22}$$

$$= R_{k+1} \left(\begin{bmatrix} \frac{1}{\alpha_0} & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} & & \\ & -\frac{1}{\alpha_1} & \ddots & \\ & & \ddots & \frac{1}{\alpha_{k-1}} \\ & & & 0 \end{bmatrix} - \frac{1}{\alpha_{k-1}} e_{k+1}^{k+1} (e_k^{k+1})^T \right) \tag{4.4.23}$$

$$= R_k \begin{bmatrix} \frac{1}{\alpha_0} & & & \\ -\frac{1}{\alpha_0} & \frac{1}{\alpha_1} & & \\ & -\frac{1}{\alpha_1} & \ddots & \\ & & \ddots & \frac{1}{\alpha_{k-1}} \end{bmatrix} - \frac{1}{\alpha_{k-1}} r_k (e_k^k)^T \tag{4.4.24}$$

$$AP_k = R_k L_k - \frac{1}{\alpha_{k-1}} r_k (e_k^k)^T, \tag{4.4.25}$$

onde $L_k \in \mathbb{C}^{k \times k}$. Juntando este resultado com (4.4.19), obtemos

$$AR_k U_k^{-1} = R_k L_k - \frac{1}{\alpha_{k-1}} r_k (e_k^k)^T \tag{4.4.26}$$

$$AR_k = R_k L_k U_k - \frac{1}{\alpha_{k-1}} r_k (e_k^k)^T U_k. \tag{4.4.27}$$

Como U_k é triangular superior com uns na diagonal, então $(e_k^k)^T U_k = (e_k^k)^T$, e

$$AV_k N_k = V_k N_k L_k U_k - \frac{1}{\alpha_{k-1}} r_k (e_k^k)^T \tag{4.4.28}$$

$$AV_k = V_k N_k L_k U_k N_k^{-1} - \frac{1}{\alpha_{k-1} \|r_{k-1}\|_2} r_k (e_k^k)^T \tag{4.4.29}$$

Com um pouco de álgebra, podemos decompor as matrizes da seguinte forma

$$N_k L_k = \begin{bmatrix} 1 & & & & \\ -\sqrt{\beta_1} & 1 & & & \\ & -\sqrt{\beta_2} & 1 & & \\ & & \ddots & \ddots & \\ & & & -\sqrt{\beta_{k-1}} & 1 \end{bmatrix} \begin{bmatrix} \frac{\|r_0\|_2}{\alpha_0} & & & \\ & \frac{\|r_1\|_2}{\alpha_1} & & \\ & & \ddots & \\ & & & \frac{\|r_{k-1}\|_2}{\alpha_{k-1}} \end{bmatrix} \tag{4.4.30}$$

$$U_k N_k^{-1} = \begin{bmatrix} \frac{1}{\|r_0\|_2} & & & \\ & \frac{1}{\|r_1\|_2} & & \\ & & \ddots & \\ & & & \frac{1}{\|r_{k-1}\|_2} \end{bmatrix} \begin{bmatrix} 1 & -\sqrt{\beta_1} & & \\ & 1 & -\sqrt{\beta_2} & \\ & & 1 & \ddots \\ & & & \ddots & -\sqrt{\beta_{k-1}} \\ & & & & 1 \end{bmatrix} \tag{4.4.31}$$

e portanto

$$N_k L_k U_k N_k^{-1} = \bar{L}_k \begin{bmatrix} \frac{1}{\alpha_0} & & & \\ & \frac{1}{\alpha_1} & & \\ & & \ddots & \\ & & & \frac{1}{\alpha_{k-1}} \end{bmatrix} \bar{L}_k^H = \bar{L}_k D_k^{-1} \bar{L}_k^H \tag{4.4.32}$$

Podemos então interpretar $\bar{L}_k D_k^{-1} \bar{L}_k^H$ como a decomposição de Cholesky de H_k em (3.1.4). Com isso fica ainda mais clara a necessidade de a matriz A ser hermitiana positiva definida para o funcionamento deste método, uma vez que a fatoração de Cholesky apenas existe para matrizes deste tipo.

Porfim, temos que

$$x_k = x_{k-1} + \alpha_{k-1} p_{k-1} = x_0 + \sum_{j=0}^{k-1} \alpha_j p_j = x_0 + P_k \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{k-1} \end{bmatrix} \tag{4.4.33}$$

$$= x_0 + P_k \begin{bmatrix} \alpha_0 & & & \\ \alpha_1 & \alpha_1 & & \\ \vdots & \vdots & \ddots & \\ \alpha_{k-1} & \alpha_{k-1} & \dots & \alpha_{k-1} \end{bmatrix} e_1^k = x_0 + P_k L_k^{-1} e_1^k \tag{4.4.34}$$

Não é por acaso que a matriz que aparece na equação acima é a inversa de L_k. Com

ajuda de um pequeno artifício matemático, teremos

$$x_k = x_0 + P_k L_k^{-1} N_k^{-1} N_k e_1^k = x_0 + R_k U_k^{-1} L_k^{-1} N_k^{-1} N_k e_1^k \tag{4.4.35}$$

$$= x_0 + V_k N_k U_k^{-1} L_k^{-1} N_k^{-1} \|r_0\|_2 \, e_1^k \tag{4.4.36}$$

$$= x_0 + V_k H_k^{-1} \|r_0\|_2 \, e_1^k = x_0 + V_k y_k \tag{4.4.37}$$

Com isso, vemos que o CG encontra a aproximação x_k resolvendo o sistema $H_k y_k = \|r_0\|_2 \, e_1^k$, exatamente como o método FOM.

Resumindo então o que foi visto nessa seção, podemos interpretar o CG como um método que tenta encontrar uma aproximação x_k de modo que $b - Ax_k \perp \mathcal{K}_k$, o que pode ser resolvido fazendo $x_k = V_k y_k$. Para resolver o sistema $H_k y_k = \|r_0\|_2 \, e_1^k$, o CG efetua, implicitamente, uma fatoração de Cholesky, valendo-se do fato de que a matriz A é hermitina positiva definida, e portanto, H_k também o será.

O CG resolve o problema do armazenamento de V_k existente no FOM e no GMRES, bem como o problema do custo da ortogonalização, utilizando o processo de ortogonalização de Lanczos ao invés do processo de ortogonalização de Arnoldi, pois a matriz A é simétrica. Por outro lado, é um método muito restrito funcionando apenas para matrizes simétricas definidas positivas. Uma solução seria, ao invés de resolver o problema para A, resolver para $A^T A$, que sempre será simétrica definida positiva caso A seja não-singular. A desvantagem é que em muitos casos $A^T A$ não será esparsa, mesmo que A o seja, sendo necessário o desenvolvimento de novos métodos para estes casos.

4.5 CR - Resíduos Conjugados

Vamos agora explicar o método dos Resíduos Conjugados [15] (do original, Conjugate Residual. Chamaremos simplesmente de CR) proposto por Stiefel em 1955. Antes de explicar o funcionamento do método, vamos entender sua motivação o porque de sua origem. Lembrando que na seção 4.3 definimos $e_k = x_k - A^{-1}b$, teremos

$$\|e_k\|_A^2 = \langle e_k, Ae_k \rangle = (x_k^T - b^T A^{-1}) A (x_k - A^{-1}b) \tag{4.5.1}$$

$$= x_k^T A x_k - b^T x_k - x_k^T b + b^t A^{-1} b = x_k^T A x_k - 2x_k^T b + c = 2f(x_k), \tag{4.5.2}$$

onde $f(x)$ representa a função quadrática dada em (4.4.1). Como já foi dito, minimizar tal função quadrática quando A é simétrica definida positiva, é o mesmo que encontrar a solução de $Ax = b$. Com a relação de equivalência que acabamos de mostrar, então minimizar a norma-A do erro também é o mesmo que resolver o

sistema. Além disso

$$\|e_k\|_A^2 = \langle e_k, Ae_k\rangle = \left\langle r_k, A^{-1}r_k\right\rangle = \|r_k\|_{A^{-1}}^2 , \tag{4.5.3}$$

ou seja, minimizar a norma-A^{-1} do resíduo é o mesmo que resolver o sistema $Ax = b$ quando A é simétrica definida positiva. Seria de mais fácil compreensão caso estivéssemos tentando minimizar a norma de euclidiana de r_k. Isto pode ser feito se ao invés de usarmos o produto interno euclidiano utilizarmos um produto interno subordinado à matriz A, ou seja

$$\langle a, b\rangle_A = b^T Aa, \tag{4.5.4}$$

para quaisquer vetores $a, b \in \mathbb{C}^n$. Com isso teríamos

$$\langle e_k, Ae_k\rangle_A = \left\langle e_k, A^2 e_k\right\rangle = \langle r_k, r_k\rangle = \|r_k\|_2^2 . \tag{4.5.5}$$

Os cálculos necessários para encontrar α_k e β_k são equivalentes ao cálculo utilizado no CG, com a diferença apenas no produto interno utilizado, e portanto não serão mostradas novamente. Na verdade, muitos dos cálculos efetuados nas duas sessões anteriores são tão similares que não os efetuaremos novamente, nos limitando a citá-los.

O Algoritmo 7 mostra um pseudocódigo para o CR. Repare que, como era de se esperar, trata-se de um algoritmo idêntico ao algoritmo do CG, exceto pelo produto interno.

Algoritmo 7: CR - Resíduos Conjugados

Input: A, b, x_0
$r_0 = b - Ax_0$;
$p_0 = r_0$;
while $\|r_i\|_2 > tol$ **do**
  $\alpha_i = \frac{\|r_i\|_A}{\langle p_i, Ap_i\rangle_A}$;
  $x_{i+1} = x_i + \alpha_i p_i$;
  $r_{i+1} = r_i - \alpha_i Ap_i$;
  $\beta_i = \frac{\left\|r_{i+1}\right\|_A^2}{\|r_i\|_A^2}$;
  $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$;
end while

Repare que a condição de que $\left\langle p_j, Ap_i\right\rangle = 0$ para $i \neq j$ torna-se $\left\langle p_j, Ap_i\right\rangle_A = 0$. Uma outra formulação do método é que este exige que os resíduos sejam A-ortogonais entre si, ao invés de exigi-lo das direções de busca como foi feito no CG. Mostramos que ambas as formulações são equivalentes:

Proposição 4.5.1. *Os resíduos gerados pelo CR serão A-ortogonais, o que é equivalente a dizer que os p_j serão A^2-ortogonais. Além disso, os resíduos r_k serão A-ortogonais aos p_j para $j < k$.*

Demonstração. A demonstração de que os resíduos serão A-ortogonais entre si e aos p_j anteriores quando $\left\langle p_j, Ap_k \right\rangle_A = \left\langle p_j, p_k \right\rangle_{A^2} = 0$, $j \neq k$ é idêntica à feita nas propriedades 4.3.2 e 4.3.1, respectivamente, bastando apenas a troca do produto interno.

Suponhamos agora que $\left\langle r_j, r_k \right\rangle_A = 0$ para $j \neq k$. De (4.4.7) obtemos

$$\left\langle r_k, p_j \right\rangle_A = \left\langle r_k, \sum_{i=0}^{j} \hat{\beta}_{(j,i)} r_i \right\rangle_A = \sum_{i=0}^{j} \hat{\beta}_{(j,i)} \left\langle r_k, r_i \right\rangle_A = 0 \tag{4.5.6}$$

para $j < k$, mostrando que o resíduo será A-ortogonal aos p_j anteriores. Fazendo então o produto interno subordinado à A entre r_k e p_j, para $k > j$, teremos

$$0 = \left\langle r_k, p_j \right\rangle_A = \left\langle r_k, r_j \right\rangle_A + \beta_j \left\langle r_k, p_{j-1} \right\rangle_A \tag{4.5.7}$$

$$= \beta_j \left(\left\langle r_{k-1}, p_{j-1} \right\rangle_A - \alpha_{k-1} \left\langle Ap_{k-1}, p_{j-1} \right\rangle_A \right) \tag{4.5.8}$$

$$= -\alpha_{k-1}\beta_j \left\langle Ap_{k-1}, p_{j-1} \right\rangle_A \tag{4.5.9}$$

ou seja

$$\left\langle Ap_k, p_j \right\rangle_A = \left\langle p_k, p_j \right\rangle_{A^2} = 0. \tag{4.5.10}$$

Novamente podemos usar a idéia de que $\left\langle p_k, p_j \right\rangle_{A^2} = \left\langle p_j, p_k \right\rangle_{A^2} = 0$ e portanto a igualdade também será válida para $k < j$, tornando necessário apenas que $k \neq j$. □

Assim como no CG, neste método também teremos $r_k, p_k \in \mathcal{K}_{k+1}(A, r_0)$, e a demonstração é idêntica (embora os α_j e β_j sejam diferentes, o que é irrelevante para esta demonstração).

Vamos então mostrar algumas semelhanças entre o CR e o GMRES. No segundo teremos

$$r_k = r_0 - w_k = (I - P_{\mathcal{W}_k}) r_0 \perp \mathcal{W}_k. \tag{4.5.11}$$

Como $\mathcal{W}_k = A\mathcal{K}_k(A, r_0)$, então $r_k \perp A\mathcal{K}_k$. Do CG temos

$$r_k = r_0 - \sum_{j=0}^{k-1} \hat{\alpha}_j A r_j \tag{4.5.12}$$

que pode ser encontrado na equação (4.4.11). Esta igualdade é válida também para o CR, como é fácil perceber pelo desenvolvimento na seção 4.4. Nesta igualdade podemos perceber que $\sum_{j=0}^{k-1} \hat{\alpha}_j A r_j \in \mathcal{W}_k$, o que estabelece mais uma semelhança entre o GMRES e o CR.

Reparando também que no CR o resíduo é A-ortogonal aos demais resíduos, teremos $r_k \perp Ar_j$ para $k \neq j$. Mais especificamente, isto será verdade para j de zero a $k-1$, e como $\text{span}\{r_0, r_1, \ldots, r_{k-1}\} = \mathcal{K}_k$, teremos $r_k \perp A\mathcal{K}_k$.

Podemos então interpretar o CR como um caso especial do método GMRES, no qual a matriz é simétrica definida positiva, o que simplifica os cálculos e reduz os recursos necessários para a execução do algoritmo.

De modo similar ao CG, o CR resolve o problema da ortogonalização e do armazenamento de V_k, mas ainda compartilha a devantagem de só ser aplicável a casos onde A é simétrica definida positiva.

Reparemos também que é possível escrever o algoritmo do CR utilizando apenas uma multiplicação matrix-vetor, conforme mostra o algoritmo 8. Entretanto se faz necessário o armazenamento de dois outros vetores: c_k e Ar_k.

Algoritmo 8: Resíduos Conjugados com apenas uma multiplicação Matrix-Vetor por iteração

Input: A, b, x_0
$r_0 = b - Ax_0$;
$p_0 = r_0$;
while $\|r_i\|_2 > tol$ **do**
  $\beta_i = \frac{\langle r_i, Ar_i \rangle}{\langle r_{i-1}, Ar_{i-1} \rangle}$;
  $p_i = r_i + \beta_i p_{i-1}$;
  $c_i = Ar_i + \beta_i c_{i-1}$;
  $\alpha_i = \frac{\langle r_i, Ar_i \rangle}{\langle c_i, c_i \rangle}$;
  $x_{i+1} = x_i + \alpha_i p_i$;
  $r_{i+1} = r_i - \alpha_i c_i$;
end while

4.6 QMR - Resíduos Quase-Mínimos

Vamos agora tratar do método dos Resíduos Quase-Mínimos [16] (Quasi-Minimal Residual, ou QMR), proposto por Freund e Nachtigal em 1991. O principal objetivo deste método é tentar tridiagonalizar a matriz $\tilde{H}_k$ a cada iteração, de modo a não precisarmos de todos os v_j e também simplificarmos a ortogonalização.

O que faremos aqui muito semelhante ao que foi feito no GMRES exceto que, ao invés do processo de ortogonalizçaão de Arnoldi, iremos utilizar o processo de Lanczos Não-Hermitiano para gerar duas bases biortogonais, conforme visto em 3.5. Não armazenaremos todos os z_j, mas somente os necessários para a biortogonalização.

Em [16] é proposto que, ao invés de exigirmos que $\langle v_j, z_j \rangle = 1$, passemos a exigir que $\langle v_j, z_j \rangle \neq 0$ e $\|v_j\| = \|z_j\| = 1$ durante o processo de Lanczos Não-Hermitiano.

Com isso teríamos $Z_k^H V_k = D_k$ onde $D_k \in \mathbb{C}^{k\times k}$ é uma matriz diagonal. Isto mudaria ligeiramente os cálculos vistos em 3.5, e teríamos

$$Z_k^H A V_k = Z_k^H V_{k+1} \tilde{H}_k = [D_k \quad 0]\tilde{H}_k = D_k H_k \tag{4.6.1}$$

$$V_k^H A^H Z_k = V_k^H Z_{k+1} \tilde{Y}_k = [D_k^* \quad 0]\tilde{Y}_k = D_k^* Y_k \tag{4.6.2}$$

$$Y_k = D_k^{-*} H_k^H D_k^* \tag{4.6.3}$$

ao invés de $Y_k = H_k^H$. Os devidos ajustes serão feitos no algortimo para que tenhamos a versão equivalente.

Assim como no GMRES, tentaremos aqui minimizar a norma do resíduo, ou seja

$$\min_{y\in\mathbb{C}^n} \left\| V_{k+1} \left(\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y \right) \right\|_2 \tag{4.6.4}$$

O problema é que não podemos retirar V_{k+1} da norma aqui pois no QMR ela não será de uma matriz unitária. O que faremos será negligenciar este fato, e resolver o problema

$$\min_{y\in\mathbb{C}^n} \left\| \|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y \right\|_2 \tag{4.6.5}$$

exatamente como foi feito no GMRES, aplicando rotações de Givens a cada iteração para obter a decomposição QR econômica de $\tilde{H}_k$.

Seguindo o outro ponto de vista estabelecido no desenvolvimento do GMRES, mesmo fazendo a fatoração QR de $\tilde{H}_k$, teríamos

$$A V_k = (V_{k+1}\bar{Q}_k)\bar{R}_k. \tag{4.6.6}$$

O problema é que neste caso a matriz entre parênteses não é unitária, e portanto não obtemos uma base ortonormal para $\mathcal{W}_k$ e logo não podemos construir a projeção de r_0 em $\mathcal{W}_k$ sem um custo adicional significativo. O algoritmo 9 mostra um pseudocódigo para o QMR.

Notemos que, por utilizar o método de Lanczos Não-Hermitiano, o QMR também pode apresentar rupturas. Não entraremos em detalhes a respeito de como evitar rupturas no QMR, mas em [16] há uma discussão a este respeito.

Vamos mostrar agora o fato de ignorarmos V_{k+1} em (4.6.4) sob um outro ponto de vista. Consideremos

$$v, w \in \mathcal{K}_\ell, \; v = V_\ell x \quad \text{e} \quad w = V_\ell y. \tag{4.6.7}$$

Definiremos o produto interno em $\mathcal{K}_\ell$

$$\langle v, w \rangle_{\mathcal{Z}} = \langle V_\ell x, V_\ell y \rangle_{\mathcal{Z}} := \langle x, y \rangle . \tag{4.6.8}$$

Algoritmo 9: QMR - Resíduos Quase-Mínimos

$r_0 = b - Ax_0$;
$\beta = \|r_0\|_2$;
$v_1 = z_1 = \dfrac{r_0}{\beta}$;
$\zeta_0 = \nu_0 = 0$;
$j = 1$;
while $\left\|r_j\right\|_2 > tol$ **do**
  $v_{j+1} = Av_j$;
  $z_{j+1} = A^H z_j$;
  $\eta_j = \left\langle z_j, v_{j+1} \right\rangle$;
  $\delta_j = \left\langle v_j, z_j \right\rangle$;
  $v_{j+1} = v_{j+1} - \eta_j v_j - \zeta_j^* v_{j-1}$;
  $z_{j+1} = \dfrac{z_{j+1}}{\delta_j} - \dfrac{\eta_j^*}{\delta_j} z_j - \dfrac{\nu_j^*}{\delta_{j-1}} z_{j-1}$;
  $\nu_{j+1} = \left\|v_{j+1}\right\|_2$;
  $\zeta_j = \left\|z_{j+1}\right\|_2$;
  $v_{j+1} = \dfrac{v_{j+1}}{\nu_{j+1}}$;
  $z_{j+1} = \dfrac{z_{j+1}}{\zeta_{j+1}}$;
  $\tilde{H}_j y = \beta_2 e_1^{j+1}$;
  $x_{j+1} = x_j + V_j y$;
  $r_{j+1} = r_j - V_{j+1}\tilde{H}_j y$;
  $j = j + 1$;
end while

Um exemplo de produto interno em $\mathcal{K}_\ell$ satisfazendo (4.6.8) seria

$$\langle a, b \rangle_{\mathcal{Z}} = b^H Z_\ell D_\ell^{-*} D_\ell^{-1} Z_\ell^H a. \tag{4.6.9}$$

com $a, b \in \mathcal{K}_\ell$. Com este produto interno em consideração, a matriz V_k gerada pelo algoritmo 9 será uma base $\mathcal{Z}$-ortogonal em $\mathcal{K}_\ell$ pois

$$\left\langle v_i, v_j \right\rangle_{\mathcal{Z}} = v_j^H Z_\ell D_\ell^{-*} D_\ell^{-1} Z_\ell^H v_i \tag{4.6.10}$$

$$= \left\langle v_j, z_j \right\rangle (e_j^\ell)^H D_\ell^{-*} D_\ell^{-1} \left\langle z_i, v_i \right\rangle e_i^\ell, \tag{4.6.11}$$

ou seja, $\left\langle v_i, v_j \right\rangle_{\mathcal{Z}} = \delta_{i,j}$. Além disso, a norma na equação (4.6.4) será

$$\begin{aligned}
&\left\| V_{k+1} \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right) \right\|_{\mathcal{Z}} \\
=&\sqrt{\left\langle V_{k+1} \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right), V_{k+1} \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right) \right\rangle_{\mathcal{Z}}} \\
=&\sqrt{\left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right)^H V_{k+1}^H Z_\ell D_\ell^{-*} D_\ell^{-1} Z_\ell^H V_{k+1} \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right)} \\
=&\sqrt{\left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right)^H \left(\|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right)} \\
=&\left\| \|r_0\|_2 e_1^{k+1} - \tilde{H}_k y \right\|_2
\end{aligned} \tag{4.6.12}$$

ou seja, a mudança do produto interno justifica o fato de ignorarmos V_k na norma em (4.6.4). É importante notar que não precisamos de fato da matriz Z_ℓ, apenas a consideramos para o entendimento teórico. Reparamos também que a matriz $Z_\ell D_\ell^{-*} D_\ell^{-1} Z_\ell^H$ poderá não ser definida positiva em todo o espaço, e assim, o produto interno $\mathcal{Z}$ só será um produto interno em $\mathcal{K}_\ell$. Além disso, poderá não ser possível encontrar um produto interno em $\mathcal{H}$ satisfazendo (4.6.8), embora possamos assegurar que isto será sempre possível em $\mathcal{K}_\ell$.

Como não estamos de fato minimizando o resíduo com relação à um produto interno em $\mathcal{H}$, este algortimo não garante que a norma euclidiana do resíduo seja zero na iteração em que o algoritmo termina, o que significa que podemos não ter alcançado a convergência, sendo esta a desvantagem deste método.

4.7 BiCG - Gradientes BiConjugados

Apresentamos agora o método dos Gradientes BiConjugados [13] (BiConjugate Gradients ou BiCG), criado por Lanczos em 1952. Assim como o CG, este método teve uma formulação bastante diferente da dos métodos apresentados aqui, sendo notável o trabalho de Fletcher [17] ao expor este método como um método de Krylov.

Assim como no QMR, construiremos duas bases biortogonais V_k e Z_k, mas desta vez seguiremos os mesmos passos descritos em 3.5, ou seja $\left\langle v_j, z_j \right\rangle = \delta_{i,j}$. Efetuaremos (implicitamente) a fatoração

$$H_k = N_k L_k U_k N_k^{-1} \tag{4.7.1}$$

onde $L_k \in \mathbb{C}^{k \times k}$ e $U_k \in \mathbb{C}^{k \times k}$ são as mesmas matrizes encontradas no CG(ver seção 4.4), respectivamente triangular inferior e superior. Repare que aqui não podemos interpretar esta decomposição como uma decomposição de Cholesky de H_k, pois nada garante que H_k seja simétrica, e nos limitaremos à decomposição LU.

Fazendo também $P_k = R_k U_k^{-1}$, onde $R_k = V_k N_k$, podemos escrever

$$x_k = x_0 + V_k H_k^{-1} \|r_0\|_2 \, e_1^k = x_0 + V_k N_k U_k^{-1} L_k^{-1} N_k^{-1} \|r_0\|_2 \, e_1^k \tag{4.7.2}$$

$$= x_0 + P_k L_k^{-1} e_1^k \tag{4.7.3}$$

O algoritmo 10 mostra um pseudocódigo para o BiCG. Como era de se esperar existe uma semelhança muito grande entre este algoritmo e o CG (ver algoritmo 6). A diferença é que incluímos o cálculo dos t_j e q_j para que seja então possível calcular os α_j e β_j de acordo com a nova estrutura de H_k. Também vemos que o cálculo de x_{i+1} neste algoritmo está de acordo com a equação (4.7.2), como é possível mostrar com uma simples indução.

Algoritmo 10: BiCG - Gradientes BiConjugados

Input: A, b, x_0

$r_0 = b - Ax_0$;
$t_0 = p_0 = r_0$;
while $\|r_i\|_2 > tol$ **do**
  $\alpha_i = \dfrac{\langle t_i, r_i \rangle}{\langle q_i, Ap_i \rangle}$;
  $x_{i+1} = x_i + \alpha_i p_i$;
  $r_{i+1} = r_i - \alpha_i A p_i$;
  $t_{i+1} = t_i - \alpha_i A^H q_i$;
  $\beta_{i+1} = \dfrac{\langle t_{i+1}, r_{i+1} \rangle}{\langle t_i, r_i \rangle}$;
  $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$;
  $q_{i+1} = t_{i+1} + \beta_{i+1} q_i$;
end while

A seguinte propriedade pode ser encontrada em [2], p.211

Propriedade 4.7.1. *Os vetores produzidos pelo BiCG satisfazem as propriedades*

$$\left\langle t_i, r_j \right\rangle = 0, \quad e \quad \left\langle q_i, Ap_j \right\rangle = 0 \tag{4.7.4}$$

para $i \neq j$.

Demonstração. Assim como escrevemos $V_k = R_k N_k^{-1}$, teremos $Z_k = T_k M_k^{-1}$ para uma matriz diagonal $M_k \in \mathbb{C}^{k \times k}$. Portanto

$$V_k^H Z_k = I_k \tag{4.7.5}$$

$$N_k^{-H} R_k^H T_k M_k^{-1} = I_k \tag{4.7.6}$$

$$R_k^H T_k = N_k^H M_k \tag{4.7.7}$$

que é uma matriz diagonal, provando que $\left\langle t_i, r_j \right\rangle = 0$ para todo $i \neq j$. Seja Q_k a matriz cujas colunas são os q_j. Da mesma forma que obtivemos (4.4.7) na seção 4.4, obteremos aqui

$$p_k = \sum_{i=0}^{k} \hat{\beta}_{(k,i)} r_i, \quad e \quad q_k = \sum_{i=0}^{k} \hat{\beta}_{(k,i)} t_i \tag{4.7.8}$$

o que nos leva à

$$t_j^H p_k = \sum_{i=0}^{k} \hat{\beta}_{(k,i)} t_j^H r_i, \qquad r_j^H q_k = \sum_{i=0}^{k} \hat{\beta}_{(k,i)} r_j^H t_i \tag{4.7.9}$$

$$t_j^H p_k = 0 \qquad r_j^H q_k = 0 \tag{4.7.10}$$

para $j < k$. Com isso, multiplicando o cálculo de r_{k+1} por q_j^H com $j < k$ teremos

$$q_j^H r_{k+1} = q_j^H r_k - \alpha_k q_j^H A p_k \tag{4.7.11}$$

$$q_j^H A p_k = 0 \tag{4.7.12}$$

e de modo similar, multiplicando o cálculo de t_{k+1} por p_j^H com $j < k$

$$p_j^H A^H q_k = 0, \tag{4.7.13}$$

mostrando que $\left\langle q_i, A p_j \right\rangle = 0$ para $i \neq j$. □

Também podemos verificar, da mesma forma que foi verificado na seção 4.4, que os r_j formam uma base para o subespaço de Krylov $\mathcal{K}_k(A, r_0)$ e que os t_j formam uma base para o subespaço de Krylov $\mathcal{K}_k(A^H, r_0)$, embora estas bases não sejam ortonormais.

Podemos então interpretar o BiCG como um método que tenta a cada iteração encontrar r_k tal que $r_k \perp \mathcal{K}_k(A^H, r_0)$. O ideal seria resolver o problema

$$V_{k+1} \left(\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y_k \right) \perp \mathcal{K}_k \tag{4.7.14}$$

mas isso teria um grande custo pois V_{k+1} não é $\mathcal{K}_k$ ortogonal. Se, por outro lado trocarmos o produto interno euclidiano por um produto interno $\mathcal{Z}$ em $\mathcal{K}_\ell$ satisfazendo (4.6.8), na perpendicularidade estabelecida em (4.7.14) teremos

$$V_{k+1} \left(\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y_k \right) \perp_{\mathcal{Z}} \mathcal{K}_k \tag{4.7.15}$$

$$\|r_0\|_2 \, e_1^{k+1} - \tilde{H}_k y_k \perp \mathcal{K}_k. \tag{4.7.16}$$

Uma solução para este problem é dada por $y_k = H_k^{-1} \|r_0\| \, e_1^k$. Um exemplo de

produto interno satisfazendo (4.6.8) neste caso seria

$$\langle a, b\rangle_{\mathcal{Z}} = a^H Z_\ell Z_\ell^H b. \tag{4.7.17}$$

Assim vimos que o BiCG pode ser comparado ao FOM mediante à uma troca de produto interno em $\mathcal{K}_\ell$, e com isso, geramos uma grande economia no processo de geração da base para $\mathcal{K}_k$. Assim como no QMR a desvantagem está no fato de que esta base não é ortonormal e portanto não há garantia de convergência.

4.8 BiCGStab - Gradientes BiConjugados Estabilizado

Como já sabemos, o resíduo gerado pelo BiCG pertence ao espaço de Krylov e portanto pode ser escrito como

$$r_k = \sum_{i=0}^{k-1} \gamma_{(k,i)} A^i r_0 = \mathbf{r}(A)_k r_0 \tag{4.8.1}$$

onde $\mathbf{r}(t)_k$ é um polinômio de grau no máximo igual a k. Já mostramos também que as direções de busca podem ser escritas como uma combinação linear dos resíduos anteriores, e portanto

$$p_k = \sum_{i=0}^{k-1} \xi_{(k,i)} A^i r_0 = \mathbf{p}(A)_k r_0. \tag{4.8.2}$$

Repare que o polinômio para t_k e q_k são os mesmos, ou seja

$$q_k = \mathbf{p}(A^H)_k t_0 \quad \text{e} \quad t_k = \mathbf{r}(A^H)_k t_0 \tag{4.8.3}$$

Como já foi dito, no BiCG utilizamos os q_k e t_k apenas para calcular os coeficientes β_k e α_k. Entretanto, repare que

$$\langle t_k, r_k\rangle = \left\langle \mathbf{r}(A^H)t_0, \mathbf{r}(A)r_0 \right\rangle = \left\langle t_0, \mathbf{r}(A)^2 r_0 \right\rangle \tag{4.8.4}$$

$$\langle q_k, Ap_k\rangle = \left\langle \mathbf{p}(A^H)t_0, A\mathbf{p}(A)r_0 \right\rangle = \left\langle t_0, A\mathbf{p}(A)^2 r_0 \right\rangle \tag{4.8.5}$$

Se calcularmos

$$r_k = \mathbf{r}(A)_k^2 r_0 \quad \text{e} \quad p_k = \mathbf{p}(A)_k^2 r_0, \tag{4.8.6}$$

então precisaremos apenas de t_0 para calcular os α_k e β_k. Esta observação levou Sonneveld a criar o CGS [18] (Conjugate Gradients Squared) em 1989. Entretanto, este método se mostrou instável como é discutido por van der Vorst em [19]. Para resolver o problema da instabilidade, o próprio van der Vorst propôs uma versão

estável do CGS que é chamada BiCGStab [20] (Biconjugate Gradients Stabilized, ou Gradientes BiConjugados Estabilizados) na qual ao invés de fazermos (4.8.6) faremos

$$r_k = \mathbf{q}(A)_k \mathbf{r}(A)_k r_0 \quad \text{e} \quad p_k = \mathbf{q}(A)_k \mathbf{p}(A)_k r_0 \tag{4.8.7}$$

para algum polinômio $\mathbf{q}(t)_k$ de grau k ou menor. Para obter uma fácil recursão, faremos

$$\mathbf{q}(t)_{k+1} = (1 - \omega_k t)\mathbf{q}(t)_k \tag{4.8.8}$$

e deixaremos a escolha dos ω_k em aberto por enquanto. Da definição do BiCG, temos

$$\mathbf{r}(t)_{k+1} = \mathbf{r}(t)_k - \alpha_k A\mathbf{p}(t)_k \quad \text{e} \quad \mathbf{p}(t)_{k+1} = \mathbf{r}(t)_{k+1} + \beta_{k+1}\mathbf{p}(t)_k \tag{4.8.9}$$

e portanto

$$\mathbf{q}(t)_{k+1}\mathbf{r}(t)_{k+1} = (1 - \omega_k t)\left(\mathbf{q}(t)_k \mathbf{r}(t)_k - \alpha_k A\mathbf{q}(t)_k \mathbf{p}(t)_k\right) \tag{4.8.10}$$

$$\mathbf{q}(t)_k \mathbf{p}(t)_k = \mathbf{q}(t)_k \mathbf{r}(t)_k + \beta_k \mathbf{q}(t)_k \mathbf{p}(t)_{k-1} \tag{4.8.11}$$

$$= \mathbf{q}(t)_k \mathbf{r}(t)_k + \beta_k \left(1 - \omega_{k-1} t\right) \mathbf{q}(t)_{k-1} \mathbf{p}(t)_{k-1} \tag{4.8.12}$$

Com isso podemos finalmente encontrar uma fórmula para r_k e p_k:

$$r_{k+1} = \mathbf{q}(A)_{k+1}\mathbf{r}(A)_{k+1} r_0 = (1 - \omega_k A)\left(r_k - \alpha_k A p_k\right) \tag{4.8.13}$$

$$p_{k+1} = \mathbf{q}(A)_{k+1}\mathbf{p}(A)_{k+1} r_0 = r_k + \beta_k \left(1 - \omega_{k-1} A\right) p_{k-1} \tag{4.8.14}$$

Vamos agora calcular as constantes. Seja

$$\tilde{\rho}_k = \langle r_k, r_0 \rangle = \left\langle \mathbf{r}(A)_k r_0, \mathbf{q}(A^H)_k t_0 \right\rangle. \tag{4.8.15}$$

Vamos tentar encontrar uma relação entre este $\tilde{\rho}_k$ e a constante $\rho_k = \left\langle \mathbf{r}(A)_k r_0, \mathbf{r}(A^H)_k t_0 \right\rangle$ e com isto calcularemos β_k definido no algoritmo do BiCG. Escrevendo o polinômio da forma expandida

$$\mathbf{q}(t)_k = \sum_{i=0}^{k} \vartheta_{(k,i)} t^i \tag{4.8.16}$$

teremos claramente que $\mathbf{q}(A^H)_k r_0$ será uma combinação linear dos t_j para $j \leq k$, e pela própria definição do BiCG, o vetor $\mathbf{r}(A)_{k+1} r_0$ é ortogonal à todo t_j para $k \neq j$.

Com isso teremos

$$\tilde{\rho}_k = \left\langle \mathbf{r}(A)_k r_0, \mathbf{q}(A^H)_k t_0 \right\rangle = \sum_{i=0}^{k} \left\langle \mathbf{r}(A)_k r_0, \vartheta_{(k,i)} (A^H)^i t_0 \right\rangle \tag{4.8.17}$$

$$= \left\langle \mathbf{r}(A)_k r_0, \vartheta_{(k,k)} (A^H)^k t_0 \right\rangle = \vartheta_{(k,k)} \left\langle \mathbf{r}(A)_k r_0, (A^H)^k t_0 \right\rangle \tag{4.8.18}$$

Teremos um caso semelhante para ρ_k

$$\rho_k = \left\langle \mathbf{r}(A)_k r_0, \mathbf{r}(A^H)_k t_0 \right\rangle = \left\langle \mathbf{r}(A)_k r_0, \gamma_{(k,k)} (A^H)^k t_0 \right\rangle \tag{4.8.19}$$

$$= \gamma_{(k,k)} \left\langle \mathbf{r}(A)_k r_0, (A^H)^k t_0 \right\rangle, \tag{4.8.20}$$

o que nos dá a relação

$$\tilde{\rho}_k = \frac{\vartheta_{(k,k)}}{\gamma_{(k,k)}} \rho_k. \tag{4.8.21}$$

Observando a definição do polinômio em (4.8.8) é fácil notar que, supondo que conhecemos o coeficiente do termo de maior grau de $\mathbf{q}(t)_k$, o coeficiente do termo de maior grau de $\mathbf{q}(t)_{k+1}$ será $-\omega_k \vartheta_k^{(k)}$. De similar forma conhecemos o coeficiente do termo de maior grau de $\mathbf{r}(t)_{k+1}$, e obtemos

$$\vartheta_{(k+1,k+1)} = -\omega_k \vartheta_{(k,k)} \quad \text{e} \quad \gamma_{(k+1,k+1)} = -\alpha_k \gamma_{(k,k)} \tag{4.8.22}$$

e com isso

$$\tilde{\rho}_k = \frac{\omega_{k-1} \vartheta_{(k-1,k-1)}}{\alpha_{k-1} \gamma_{(k-1,k-1)}} \rho_k = \frac{\omega_{k-1}}{\alpha_{k-1}} \frac{\tilde{\rho}_{k-1}}{\rho_{k-1}} \rho_k \quad \text{e portanto} \quad \frac{\rho_k}{\rho_{k-1}} = \frac{\tilde{\rho}_k}{\tilde{\rho}_{k-1}} \frac{\omega_{k-1}}{\alpha_{k-1}} \tag{4.8.23}$$

o que nos leva à

$$\beta_{k+1} = \frac{\rho_{k+1}}{\rho_k} = \frac{\langle r_{k+1}, t_0 \rangle}{\langle r_k, t_0 \rangle} \frac{\omega_k}{\alpha_k} \tag{4.8.24}$$

Para α_{k+1} teremos um caso muito similar ao cálculo de β_k, graças ao fato de Ap_k ser ortogonal a q_k:

$$\begin{aligned} \left\langle A\mathbf{p}(A)_k r_0, \mathbf{q}(A^H)_k t_0 \right\rangle &= \vartheta_{(k,k)} \left\langle A\mathbf{p}(A)_k r_0, (A^H)^k t_0 \right\rangle \\ \left\langle A\mathbf{p}(A)_k r_0, \mathbf{p}(A^H)_k t_0 \right\rangle &= \xi_{(k,k)} \left\langle A\mathbf{p}(A)_k r_0, (A^H)^k t_0 \right\rangle \\ &= \frac{\xi_{(k,k)}}{\vartheta_{(k,k)}} \left\langle A\mathbf{p}(A)_k r_0, \mathbf{q}(A^H)_k t_0 \right\rangle \end{aligned} \tag{4.8.25}$$

Disto e de (4.8.21) teremos

$$\alpha_k = \frac{\rho_k}{\langle A\mathbf{p}(A)_k r_0, \mathbf{p}(A^H)_k t_0 \rangle} = \frac{\gamma_{(k,k)}}{\vartheta_{(k,k)}} \frac{\vartheta_{(k,k)}}{\xi_{(k,k)}} \frac{\tilde{\rho}_k}{\langle A\mathbf{p}(A)_k r_0, \mathbf{q}(A^H)_k t_0 \rangle} \tag{4.8.26}$$

Pela própria definição dos polinômios (ver equação (4.8.9)) vemos que o termo de

maior grau de $\mathbf{p}(t)_k$ será o termo de maior grau de $\mathbf{r}(t)_k$, e com isso $\gamma_{(k,k)} = \xi_{(k,k)}$, obtendo

$$\alpha_k = \frac{\tilde{\rho}_k}{\langle Ap_k, t_0 \rangle} \tag{4.8.27}$$

Vamos agora definir os ω_k. A princípio este poderia ser um parâmetro livre, mas uma boa idéia seria defini-lo de modo a minimizar a norma euclidiana de r_k. Para simplificar os cálculos, faremos

$$s_k = r_k - \alpha_k A p_k \tag{4.8.28}$$

Assim, o resíduo será

$$r_{k+1} = (1 - \omega_k A)\, s_k = s_k - \omega_k A s_k \tag{4.8.29}$$

Se definirmos

$$\omega_k = \frac{\langle As_k, s_k \rangle}{\langle As_k, As_k \rangle} \tag{4.8.30}$$

então $\omega_k A s_k$ será a projeção de s_k em As_k, e portanto, teremos o resíduo minimizado.

Para encontrar uma expressão para x_j, lembremos que durante o desenvolvimento do CG an seção 4.3 decidimos escrever o erro e_k como uma combinação linear de um conjunto de vetores linearmente independentes, e a cada passo encontraríamos um novo vetor e seu respectivo coeficiente. No caso dos gradientes conjugados estes vetores eram os p_k e os coeficientes eram $-\alpha_k$. No BiCGStab teremos

$$r_{k+1} = s_k - \omega_k A s_k = r_k - \alpha_k A p_k - \omega_k A s_k = r_k - \alpha_k A \left(p_k - \frac{\omega_k}{\alpha_k} s_k \right) \tag{4.8.31}$$

e portanto, ao invés dos p_j teremos o vetor entre parêntesis na equação acima. Efetuando a substituição de p_k por este vetor no cálculo de x_k, teremos

$$x_{k+1} = x_k + \alpha_k \left(p_k - \frac{\omega_k}{\alpha_k} s_k \right) = x_k + \alpha_k p_k + \omega_k s_k. \tag{4.8.32}$$

O algoritmo 11 mostra um pseudocódigo para o BiCGStab. Repare que como podemos definir $t_0 = r_0$, não precisaremos desta variável no algoritmo.

4.9 GCR - Resíduos Conjugados Generalizado

Este método foi proposto por Eisenstat, Elman e Schultz em [21] como um a generalização do CR para sistemas não-Hermitianos. Em [21] é estabelecido um algoritmo geral para métodos de resíduos conjugados, que pode ser visto no algoritmo 12.

Como foi dito na sessão 4.5, uma das formulações do CR é que este método exige

Algoritmo 11: BiCGStab - Gradientes BiConjugados Estabilizado

Input: A, b, x_0

1 $r_0 = b - Ax_0$;
2 $p_0 = r_0$;
3 **while** $\|r_i\|_2 > tol$ **do**
4 $\quad \alpha_i = \dfrac{\langle r_i, r_0 \rangle}{\langle Ap_i, r_0 \rangle}$;
5 $\quad s_i = r_i - \alpha_i A p_i$;
6 $\quad w_i = \dfrac{\langle As_i, s_i \rangle}{\langle As_i, As_i \rangle}$;
7 $\quad x_{i+1} = x_i + \alpha_i p_i + w_i s_i$;
8 $\quad r_{i+1} = s_i - \omega_i A s_i$;
9 $\quad t_{i+1} = t_i - \alpha_i A^H q_i$;
10 $\quad \beta_{i+1} = \dfrac{\langle r_{i+1}, r_0 \rangle}{\langle r_i, r_0 \rangle} \times \dfrac{\alpha_i}{\omega_i}$;
11 $\quad p_{i+1} = r_{i+1} + \beta_{i+1}(p_i - \omega_i A p_i)$;
12 **end while**

Algoritmo 12: Família dos Resíduos Conjugados

1 $r_0 = b - Ax_0$;
2 $p_0 = r_0$;
3 **for** $i = 0$ *até convergir* **do**
4 $\quad \alpha_i = \dfrac{\langle r_i, Ap_i \rangle}{\langle Ap_i, Ap_i \rangle}$;
5 $\quad x_{i+1} = x_i + \alpha_i p_i$;
6 $\quad r_{i+1} = r_i - \alpha_i A p_i$;
7 $\quad$ Calcula p_{i+1};
8 **end for**

que os resíduos sejam mutuamente A-ortogonais, o que faz com que as direções p_k sejam A^2-ortogonais. Isto é assegurado pelo cálculo de p_{i+1} no algoritmo 7 pois a matriz A é simétrica definida positiva, mas quando a matriz A não possui nenhuma propriedade especial, é necessário fazer uma ortogonalização com relação à todas as direções previamente geradas.

Se tomarmos p_{i+1} na linha 7 do algoritmo 12 como

$$\begin{aligned} \beta_{j,i} &= \frac{\left\langle Ar_{i+1}, Ap_j \right\rangle}{\left\langle Ap_j, Ap_j \right\rangle} \\ p_{i+1} &= r_{i+1} - \sum_{j=0}^{i} \beta_j p_j \end{aligned} \tag{4.9.1}$$

então estaremos na verdade $A^H A$-ortogonalizando p_{i+1} com todos os p_k já gerados, de um modo muito semelhante ao processo de ortogonalização de Gram-Schmidt.

Repare que, caso A seja simétrica, então as direções serão A^2-ortogonais. O método que utiliza o cálculo de p_{i+1} dado em (4.9.1) é chamado de GCR, Generalized Conjugate Residual.

O teorema abaixo, cuja prova pode ser encontrada em [21], mostra uma série de propriedades que são encontradas no algoritmo do GCR:

Teorema 4.9.1. *Se $\{x\}_{j=1}^{k}$, $\{x\}_{j=1}^{k}$ e $\{x\}_{j=1}^{k}$ são gerados pelas iterações do método GCR, então as seguintes relações são satisfeitas:*

$$\langle Ap_i, Ap_i\rangle = 0, i \neq j, \tag{4.9.2}$$

$$\langle r_i, Ap_i\rangle = 0, i > j, \tag{4.9.3}$$

$$\langle r_i, Ap_i\rangle = \langle r_i, Ar_i\rangle, \tag{4.9.4}$$

$$\langle r_i, Ar_i\rangle = 0, i > j, \tag{4.9.5}$$

$$\langle r_i, Ap_i\rangle = \langle r_0, Ap_i\rangle, i \geq j, \tag{4.9.6}$$

$$\text{span}\{p_0, p_1, \ldots, p_i\} = \text{span}\{p_0, Ap_0, \ldots, A^i p_0\} = \text{span}\{r_0, r_1, \ldots, r_i\} \tag{4.9.7}$$

$$\text{se } r_i \neq 0, \text{ então } p_i \neq 0 \tag{4.9.8}$$

$$x_{i+1} \text{ minimiza } \|b - Ax\| \text{ no espaço } \text{span}\{x_0, p_0, p_1, \ldots, p_i\} \tag{4.9.9}$$

Assim como no CR, os resíduos $r_0, r_1, \ldots, r_k$ formam uma base para o espaço $\mathcal{K}_k$, e portanto, com as equações (4.9.7) e (4.9.9) temos que x_{i+1} minimiza a norma do resíduo no espaço $\mathcal{K}_k$, como no GMRES.

Reparamos que efetuar a $A^H A$ ortogonalização exigiria um esforço computacional muito grande e um excesso de multiplicações matriz-matriz. Para evitar isto poderíamos guardar uma matriz $\bar{A} = A^H A$ mas isto poderia ser proibitivo, pois A pode ser muito grande e $A^H A$ pode ser densa mesmo que A seja esparsa. Além disso, o número de multiplicações matriz-vetor ainda poderia ser proibitivo durante o processo de ortogonalização.

Para economizar recursos, vamos considerar uma versão modificada do algoritmo. Fazendo $c_k = Ap_k$, de acordo com (4.9.1), calcularemos c_k da seguinte forma

$$c_k = Ap_k = Ar_k - \sum_{j=0}^{k-1} \frac{\left\langle Ar_k, Ap_j \right\rangle}{\left\| Ap_j \right\|_2^2} Ap_j \tag{4.9.10}$$

$$= Ar_k - \sum_{j=0}^{k-1} \frac{1}{\left\| c_j \right\|_2^2} c_j c_j^H Ar_k \tag{4.9.11}$$

$$= (I - D_k^{-2} C_k C_k^H) Ar_k, \tag{4.9.12}$$

onde $P_k = [p_0, p_1, \ldots, p_{k-1}]$, $C_k = [c_0, c_1, \ldots, c_{k-1}]$ e $D_k = \text{diag}\left(\|c_0\|_2, \|c_1\|_2, ..., \|c_{k-1}\|_2\right)$. Entretanto, se armazenarmos os $c_0, c_1, ...c_k$

normalizados, a matriz diagonal D_k não será mais necessária. Redefiniremos então

$$\begin{aligned} \hat{c}_k &= (I - C_k C_k^H) A r_k \\ c_k &= \frac{\hat{c}_k}{\|\hat{c}_k\|_2} \end{aligned} \tag{4.9.13}$$

Com isso, podemos calcular o resíduo r_k utilizando apenas os c_k, embora o mesmo não possa ser feito com relação aos x_k. Isto tornaria o algoritmo caro pois, além da ortogonalização mostrada em (4.9.13), deveríamos fazer a $A^H A$-ortogonalização dos p_k, processo que queremos evitar. O algoritmo 13 mostra um meio iterativo de calcular c_k de acordo com (4.9.13).

Algoritmo 13: Cálculo iterativo de c_k

```
1 c_k = A r_k;
2 for i = 0 até k − 1 do
3 |   β_i = ⟨c_k, c_i⟩;
4 |   c_k = c_k − β_i c_i;
5 end for
```

Observando a linha 4 do algoritmo 13, vemos que

$$c_k = c_k - \beta_i c_i = A p_k - \beta_i A p_i \tag{4.9.14}$$

e portanto

$$p_k = A^{-1}(c_k - \beta_i c_i) = p_k - \beta_i p_i. \tag{4.9.15}$$

Basta então descobrirmos qual será o valor de p_k para $i = 0$. De acordo com a linha 1 do algoritmo 13, teremos $p_k = r_k$, e portanto, obtemos um método iterativo para o cálculo de p_k sem demais multiplicações matriz-vetor. O algoritmo 14 mostra o pseudocódigo completo do GCR de acordo com esta formulação.

Reparemos que os p_k são $A^H A$-ortogonais entre si, mas os c_k formam uma base ortonormal. Na verdade $\text{img}(C_k) = \mathcal{W}_k$. Esta característica do GCR será cruscial para o desenvolvimento de algoritmos como o GMRESR e o GCRO, que serão vistos mais a frente.

Algoritmo 14: GCR - Resíduos Conjugados Generalizado

$r_0 = b - Ax_0$;
$k = 0$;
while $\|r_k\|_2 > tol$ **do**
    $k = k + 1$;
    $p_k = r_{k-1}$;
    $c_k = Ap_k$;
    for $i = 0$ *até* $k - 1$ **do**
        $\beta = \langle c_k, c_i \rangle$;
        $c_k = c_k - \beta c_i$;
        $p_k = p_k - \beta p_i$;
    end for
    $c_k = \dfrac{c_k}{\|c_k\|_2}$;
    $p_k = \dfrac{p_k}{\|c_k\|_2}$;
    $\alpha = \langle r_{k-1}, c_k \rangle$;
    $x_k = x_{k-1} + \alpha p_k$;
    $r_k = r_{k-1} - \alpha c_k$;
end while

Capítulo 5

Interpretação Geométrica dos Métodos

5.1 Famílias de Métodos

No capítulo anterior vimos que a maioria dos métodos pode ser descrita como uma variação do GMRES ou do FOM. O que veremos nesta seção é que este conceito pode ser extendido e formalizado de modo que tenhamos duas famílias de métodos que englobam a grande maioria dos métodos de Krylov existentes.

Vamos redefinir o espaço $\mathcal{W}_k$ de modo que este não esteja relacionado ao espaço de Krylov, de acordo com a definição a seguir:

Definição 5.1.1 (Redefinição dos Subespaços)**.** *Consideremos* $\{\mathcal{W}_j\}_{j=0}^{\ell}$ *como uma sequência qualquer de subespaços aninhados, ou seja*

$$\mathcal{W}_0 \subset \mathcal{W}_1 \subset \ldots \mathcal{W}_\ell, \tag{5.1.1}$$

onde $\dim(\mathcal{W}_k) = k$*. Consideraremos também a sequência de subespaços aninhados* $\{\mathcal{V}_j\}_{j=0}^{\ell}$*, onde* $\mathcal{V}_k = \text{span}\{r_0\} + \mathcal{W}_{k-1}$*. Chamaremos os subespaços* $\mathcal{W}_k$ *de* subespaços de aproximação *e os subespaços* $\mathcal{V}_k$ *de* subespaços residuais*.*

Também consideraremos ℓ *como a iteração na qual o algoritmo converge.*

Utilizaremos esta redefinição de subespaços daqui em diantes. Consideremos agora o seguinte teorema, cuja prova pode ser encontrada em [22]:

Teorema 5.1.2. *Suponha que* $\{h_j\}_{j=0}^{\ell}$ *seja uma sequência de aproximações de* r_0 *de modo que* $h_k \in \mathcal{W}_k$ *e que* $h_\ell = r_0$*. Então existe um produto interno* $\langle ., . \rangle_{\mathcal{Z}}$ *em* $\mathcal{V}_\ell$ *tal que*

$$\left\| r_0 - h_j \right\|_{\mathcal{Z}} = \min \| r_0 - w \|_{\mathcal{Z}}, \qquad j = 1, 2, \ldots, \ell - 1, \tag{5.1.2}$$

se e somente se $h_k \in \mathcal{W}_{k-1}$ *implica em* $h_k = h_{k-1}$*, ou em outras palavras, se e somente se*[1]

$$h_j \in \mathcal{W}_j \backslash \mathcal{W}_{j-1} \quad ou \quad h_j = h_{j-1} \quad para \quad j = 1, 2, \ldots, \ell - 1 \tag{5.1.3}$$

Este teorema mostra que todo método que gera uma sequência de aproximações h_j de r_0 satisfazendo (5.1.3) poderá ser interpretado como um método que minimiza a norma do resíduo, para algum produto interno em $\mathcal{V}_\ell$. Novamente ressaltamos que $\langle ., . \rangle_{\mathcal{Z}}$ não necessariamente será um produto interno para todo o espaço, só garantimos que $\langle a, b \rangle_{\mathcal{Z}}$ satisfaz todas as propriedades do produto interno quando $a, b \in \mathcal{V}_\ell$.

Definição 5.1.3 (Métodos da Família RM)**.** *Um método de Resíduos Mínimos (RM) é um método que gera aproximações* $\{w_j^{RM}\}_{j=0}^{\ell}$*,* $w_k^{RM} \in \mathcal{W}_k$*, de modo que*

$$w_j^{RM} = \arg \min_{w \in \mathcal{W}_j} \|r_0 - w\|_{\mathcal{Z}}, \qquad j = 1, 2, \ldots, \ell - 1, \tag{5.1.4}$$

onde $w_\ell^{RM} = r_0$*, para algum produto interno* $\langle ., . \rangle_{\mathcal{Z}}$ *em* $\mathcal{V}_\ell$*.*

Com esta definição e com o teorema 4.2.1, o teorema do ponto mais próximo, teremos

$$w_k^{RM} = P_{\mathcal{W}_k} r_0 \tag{5.1.5}$$

onde $P_{\mathcal{W}_k}$ representa o operador de projeçao $\mathcal{Z}$-ortogonal em $\mathcal{W}_k$.

Como supomos na definição 5.1.1 que $\mathcal{W}_k \subset \mathcal{W}_{k+1}$, teremos

$$\mathcal{W}_{k+1} = \mathcal{W}_k \oplus \text{span}\{u\} \tag{5.1.6}$$

para algum vetor u. A soma direta se deve ao fato de termos suposto que $\dim(\mathcal{W}_{k+1}) = k + 1$ na definição 5.1.1. Deste modo vemos que $P_{\mathcal{W}_k} r_0 = P_{\mathcal{W}_{k+1}} r_0$ se e somente se o vetor u for $\mathcal{Z}$-ortogonal a r_0, ou seja, se na iteração $k + 1$ o espaço de aproximação crescer numa direção $\mathcal{Z}$-ortogonal a r_0. Reparemos que, como toda projeção é única, não hão meios de que $w_k^{RM} \in \mathcal{W}_{k+1}$ a menos que u seja $\mathcal{Z}$-ortogonal a r_0.

Definindo o resíduo

$$r_k^{RM} = r_0 - w_k^{RM} = (I - P_{\mathcal{W}_k}) r_0, \tag{5.1.7}$$

observamos que r_k^{MR} é $\mathcal{Z}$-ortogonal a $\mathcal{W}_k$. Como r_k^{RM} é uma combinação linear de r_0 e de w_k^{RM}, então $r_k^{RM} \in \{\text{span}\{r_0\} + \mathcal{W}_k\} = \mathcal{V}_{k+1}$, justificando o nome "subespaço residual" dado à $\mathcal{V}_k$.

[1] a notação $v \in \mathcal{A} \backslash \mathcal{B}$ significa que $v \in \mathcal{A}$ e $v \notin \mathcal{B}$

Com estas informações já é possível formular uma representação gráfica válida para todos os métodos pertencentes à família RM, que pode ser vista na figura 5.1.

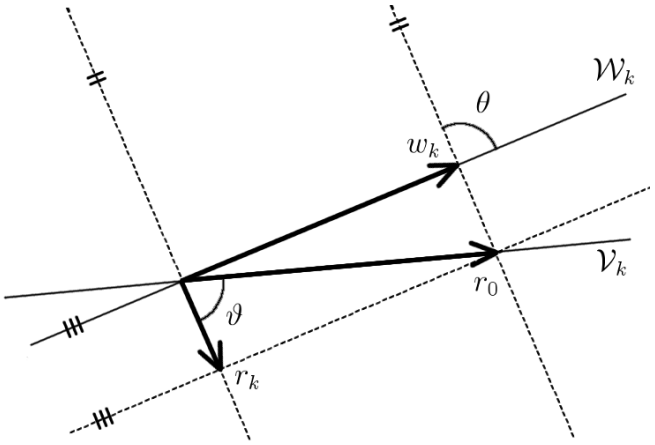

Figura 5.1: Representação gráfica da k-ésima iteração de um método da família RM

Nesta figura, o ângulo θ e ϑ dependem do produto interno em questão. Para o produto interno euclidiano teremos $\theta = \pi/2$, mas nada podemos afirmar sobre o ângulo ϑ.

Interpretando os métodos da família RM como métodos que se valem de projeções ortogonais, verificamos ques estes não possuem rupturas, pois sempre é possível projetar ortogonalmente. Reparemos que, no caso em que $\mathcal{W}_k$ é $\mathcal{Z}$-ortogonal a r_0, a aproximação w_k resultante será zero. Embora ainda não tenha sido discutido como o espaço $\mathcal{W}_k$ é expandido a cada iteração, pelo teorema 5.1.2 e pela definição 5.1.3 sabemos que o objetivo é encontrar $w_\ell = r_0$, e portanto, evitaremos expandir o espaço $\mathcal{W}_k$ numa direção ortogonal a r_0.

Uma segunda família de métodos pode ser definida, de modo semelhante à anterior, graças ao seguinte teorema:

Teorema 5.1.4. *Suponha que $\{h_j\}_{j=0}^{\ell}$ é uma sequência de aproximações de r_0 de modo que $h_k \in \mathcal{W}_k$ e que $h_\ell = r_0$. Então existe um produto interno $\langle ., . \rangle_{\mathcal{Z}}$ em $\mathcal{V}_\ell$ tal que*

$$r_0 - h_j \perp_{\mathcal{Z}} \mathcal{V}_j, \qquad j = 1, 2, \ldots, \ell - 1, \tag{5.1.8}$$

se e somente se $h_j \in \mathcal{W}_j \backslash \mathcal{W}_{j-1}$ para $j = 1, 2, \ldots, \ell - 1$.

Definição 5.1.5 (Métodos da Família RO)**.** *Um método de Resíduos Ortogonais (RO) é um método que gera aproximações $\{w_j^{RM}\}_{j=0}^{\ell}$, $w_k^{ROs} \in \mathcal{W}_k$, de modo que*

$$r_0 - w_k^{RO} \perp_{\mathcal{Z}} \mathcal{V}_k, \tag{5.1.9}$$

onde $w_\ell^{RO} = r_0$, para algum produto interno $\langle ., . \rangle_{\mathcal{Z}}$ em $\mathcal{V}_\ell$.

De modo similar ao que foi feito com os métodos da família RM, podemos interpretar a aproximação RO como

$$w_k^{RO} = P_{\mathcal{W}_k}^{\mathcal{V}_k} r_0 \tag{5.1.10}$$

onde $P_{\mathcal{W}_k}^{\mathcal{V}_k}$ representa o operador de projeção em $\mathcal{W}_k$ que é $\mathcal{Z}$-ortogonal a $\mathcal{V}_k$. Tal projetor garante que w_k^{RO} pertença a $\mathcal{W}_k$ ao mesmo tempo que garante que o mesmo seja $\mathcal{Z}$-ortogonal a $\mathcal{V}_k$.

Vamos definir o resíduo

$$r_k^{RO} = r_0 - w_k^{RO} = (I - P_{\mathcal{W}_k}^{\mathcal{V}_k}) r_0. \tag{5.1.11}$$

A figura 5.2 mostra uma representação gráfica deste processo.

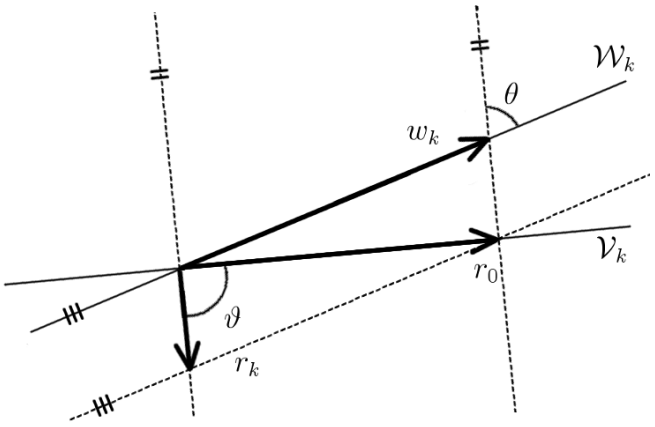

Figura 5.2: Representação gráfica da k-ésima iteração de um método da família RO

Caso o produto interno utilizado seja o produto interno euclidiano, o ângulo ϑ será $\pi/2$, mas nada podemos afirmar do ângulo θ.

Ao contrário da família RM, este método falhará quando $\mathcal{W}_k$ for $\mathcal{Z}$-ortogonal a $\mathcal{V}_k$, pois o operador de projeção $P_{\mathcal{W}_k}^{\mathcal{V}_k}$ não está definido neste caso. Este é exatamente o mesmo caso no qual a aproximação RM será zero - na verdade veremos que estes fatos estão intimamente ligados nas próximas seções.

Porfim, podemos classificar os métodos já descritos como membros das famílias apresentadas: GMRES, MINRES, CR, GCR e QMR pertencem à família dos métodos RM, pois conforme descritos no capítulo anterior, todos estes métodos resolvem o problema (5.1.4) para algum produto interno $\mathcal{Z}$ em $\mathcal{V}_k$.

Similarmente, FOM, CG, SYMMLQ, BiCG e BiCGStab são métodos pertencentes à família de métodos OR, pois conforme mostrado, ressolvem o problema (5.1.8) para algum produto interno $\mathcal{Z}$ em $\mathcal{V}_k$.

5.2 Ângulos Entre Vetores e Subespaços

Nas próximas seções serão necessárias algumas definições a respeito de ângulos entre vetores e subespaços. Todas estas definições podem ser encontradas no estudo feito por Eiermann e Ernst em [22]. Por motivos de simplificação, consideraremos apenas o produto interno euclidiano daqui para frente.

Definição 5.2.1 (Ângulo entre Vetores)**.** *O ângulo entre dois vetores $x, y \in \mathcal{H}$ é dado pela relação*

$$\cos \measuredangle(x, y) := \frac{|\langle x, y\rangle|}{\|x\|\|y\|} \tag{5.2.1}$$

que determina unicamente $\measuredangle(x, y) \in [0, \pi/2]$.

Esta definição de ângulo entre vetores será mais interessante para o nosso estudo, pois é mais apropriada para o estudo de ângulos entre subespaços, que faremos adiante.

Definição 5.2.2 (Ângulo entre Vetor e Subespaço)**.** *Seja $x \in \mathcal{H}$ um vetor diferente de zero, e $\mathcal{U} \subset \mathcal{H}$ diferente de vazio. Então definimos*

$$\measuredangle(x, \mathcal{U}) := \inf_{0 \neq u \in \mathcal{U}} \measuredangle(x, u) \tag{5.2.2}$$

$$\cos \measuredangle(x, \mathcal{U}) := \sup_{0 \neq u \in \mathcal{U}} \cos \measuredangle(x, u) \tag{5.2.3}$$

como o ângulo entre o vetor x e o subespaço $\mathcal{U}$. Com isto definimos também

$$\operatorname{sen}\measuredangle(x, \mathcal{U}) := \sqrt{1 - \cos^2 \measuredangle(x, \mathcal{U})} \tag{5.2.4}$$

O seguinte teorema traz mais alguns resultados que serão úteis no futuro, e cuja prova pode ser encontrada, por exemplo, em [23].

Teorema 5.2.3 (Relações do Ângulo entre Vetor e Subespaço)**.** *Seja $x \in \mathcal{H}$ um vetor diferente de zero, e $\mathcal{U} \subset \mathcal{H}$ diferente de vazio e $P_{\mathcal{U}}$ o operador de projeção ortogonal em $\mathcal{U}$. Então teremos*

$$\measuredangle(x, \mathcal{U}) = \measuredangle(x, P_{\mathcal{U}}) \tag{5.2.5}$$

e por consequência

$$\|P_{\mathcal{U}} x\| = \|x\| \cos \measuredangle(x, \mathcal{U}), \tag{5.2.6}$$

$$\|(I - P_{\mathcal{U}})x\| = \|x\| \operatorname{sen}\measuredangle(x, \mathcal{U}), \tag{5.2.7}$$

Definição 5.2.4 (Ângulos Canônicos)**.** *Chamaremos os ângulos $\{\theta\}_{j=1}^{k}$ de* ângulos canônicos *ou* ângulos principais *entre dois subespaços $\mathcal{V}$ e $\mathcal{W}$ os ângulos satisfazendo*

$$\cos\theta_j := \max_{0\neq v\mathcal{V}} \max_{0\neq w\mathcal{W}} \frac{|\langle v, w\rangle|}{\|v\|\|w\|} := \frac{|\langle v, w\rangle|}{\|v\|\|w\|} \tag{5.2.8}$$

sujeito a $v \perp v_1 \perp \ldots \perp v_{j-1}$ *e* $w \perp w_1 \perp \ldots \perp w_{j-1}$.

Finalmente podemos definir o ângulo entre dois subespaços:

Definição 5.2.5 (Ângulo entre Subespaços)**.** *Definimos o ângulo entre dois subespaços* $\mathcal{V}$ *e* $\mathcal{W}$ *não vazios como*

$$\measuredangle(\mathcal{V}, \mathcal{W}) := \theta_k \tag{5.2.9}$$

onde θ_k *é o maior ângulo canônico entre* $\mathcal{V}$ *e* $\mathcal{W}$.

5.3 Considerações Geométricas na Família RM

Nesta seção, utilizaremos os conceitos de ângulos entre subespaços vistos na seção 5.2 para definir algumas relações entre aproximações e resíduos gerados por métodos da família RM.

Durante toda a seção consideraremos que dispomos de uma base $\{w\}_{j=1}^{k}$ ascendente ortonormal para os subespaços $\mathcal{W}_k$, isto é, os vetores $\{w_j\}_{j=1}^{m}$ formam uma base ortonormal para $\mathcal{W}_m$, $m = 1, 2, \ldots, k$.

Com isto já podemos apresentar a seguinte proposição:

Proposição 5.3.1. *Seja* $\{w\}_{j=1}^{k}$ *uma base ascendente ortonormal para* $\mathcal{W}_k$. *Teremos*

$$w_k^{RM} = w_{k-1}^{RM} + \langle r_0, w_k\rangle\, w_k \tag{5.3.1}$$

$$\left\|r_k^{RM}\right\|_2^2 = \left\|r_{k-1}^{RM}\right\|_2^2 - \|r_0\|_2^2 \cos^2\measuredangle(r_0, w_k) \tag{5.3.2}$$

Demonstração. Uma forma de representar o operador de projeção ortogonal em $\mathcal{W}_k$ é, denotando a matriz $W_k = [w_1, w_2, \ldots, w_k]$,

$$P_{\mathcal{W}_k} = W_k W_k^H. \tag{5.3.3}$$

Com esta consideração podemos escrever

$$w_k^{RM} = W_k W_k^H r_0 = \begin{bmatrix} W_{k-1} & w_k \end{bmatrix} \begin{bmatrix} W_{k-1}^H \\ w_k^H \end{bmatrix} r_0 \tag{5.3.4}$$

$$= W_{k-1} W_{k-1}^H r_0 + w_k w_k^H r_0 \tag{5.3.5}$$

$$= w_{k-1}^{RM} + \langle r_0, w_k\rangle\, w_k, \tag{5.3.6}$$

provando (5.3.1).

Utilizando este resultado, teremos

$$\left\|r_k^{RM}\right\|_2^2 = \left\|r_0 - w_k^{RM}\right\|_2^2 = \left\|r_0 - w_{k-1}^{RM} - \langle r_0, w_k\rangle\, w_k\right\|_2^2 \tag{5.3.7}$$

$$= \left\|r_{k-1}^{RM} - \langle r_0, w_k\rangle\, w_k\right\|_2^2 \tag{5.3.8}$$

$$= \left\|r_{k-1}^{RM}\right\|_2^2 - 2\left|\left\langle r_{k-1}^{RM}, \langle r_0, w_k\rangle\, w_k\right\rangle\right| + \|\langle r_0, w_k\rangle\, w_k\|_2^2 \tag{5.3.9}$$

$$= \left\|r_{k-1}^{RM}\right\|_2^2 - 2\left|\langle r_0, w_k\rangle\left\langle r_0 - w_{k-1}^{RM}, w_k\right\rangle\right| + |\langle r_0, w_k\rangle|^2 \overbrace{\|w_k\|_2^2}^{=1} \tag{5.3.10}$$

$$= \left\|r_{k-1}^{RM}\right\|_2^2 - 2\left|\langle r_0, w_k\rangle^2 - \langle r_0, w_k\rangle\left\langle w_{k-1}^{RM}, w_k\right\rangle\right| + |\langle r_0, w_k\rangle|^2 \tag{5.3.11}$$

Como $w_{k-1}^{RM} \in \mathcal{W}_{k-1}$ e $\{w\}_{j=1}^{k}$ forma uma base ortonormal para $\mathcal{W}_k$, teremos $w_k \perp w$ para qualquer $w \in W_m$, $m = 1, 2, \ldots, k-1$, e consequentemente

$$\left\|r_k^{RM}\right\|_2^2 = \left\|r_{k-1}^{RM}\right\|_2^2 - |\langle r_0, w_k\rangle|^2 \tag{5.3.12}$$

e pela definição (5.2.1)

$$\left\|r_k^{RM}\right\|_2^2 = \left\|r_{k-1}^{RM}\right\|_2^2 - \|r_0\|_2^2 \cos^2 \measuredangle(r_0, w_k), \tag{5.3.13}$$

completando a prova. □

Reparamos que sempre que $w_k \perp r_0$, teremos $\langle r_0, w_k\rangle = 0$ e consequentemente a aproximação RM não mudará da iteração $k-1$ para k, ou seja, $w_k^{RM} = w_{k-1}^{RM}$. Como os subespaços $\mathcal{W}_j$ são aninhados, podemos então afirmar que

$$\mathcal{W}_k = \text{span}\{w_k\} \oplus \mathcal{W}_{k-1}. \tag{5.3.14}$$

A soma na relação acima é direta porque w_k por definição pertence à $\mathcal{W}_{k-1}^{\perp}$, uma vez que a base $\{w_j\}_{j=1}^{m}$ é ortonormal. Formalizamos este resultado na proposição que se segue.

Proposição 5.3.2. *Caso o subespaço $\mathcal{W}_{k-1}$ seja expandido numa direção w_k tal que $w_k \in \mathcal{W}_{k-1}^{\perp}$ e $w_k \perp r_0$, então $w_k^{RM} = w_{k-1}^{RM}$ e $r_k^{RM} = r_{k-1}^{RM}$.*

Reparemos que a proposição acima cobre também o caso em que $\mathcal{W}_{k-1}^{\perp} = 0$, pois nesse caso $w_k = 0$, e por definição, o vetor 0 é ortogonal a todos os vetores.

Conforme observado anteriormente, o objetivo dos métodos tanto da família RM quanto RO é encontrar um $w_\ell \in \mathcal{W}_\ell$ de modo que $w_\ell = r_0$, e portanto, seria interessante evitar que o subespaço $\mathcal{W}_{k-1}$ seja expandido numa direção ortogonal a r_0. As figuras 5.4 e 5.3 mostram uma representação gráfica destes dois casos.

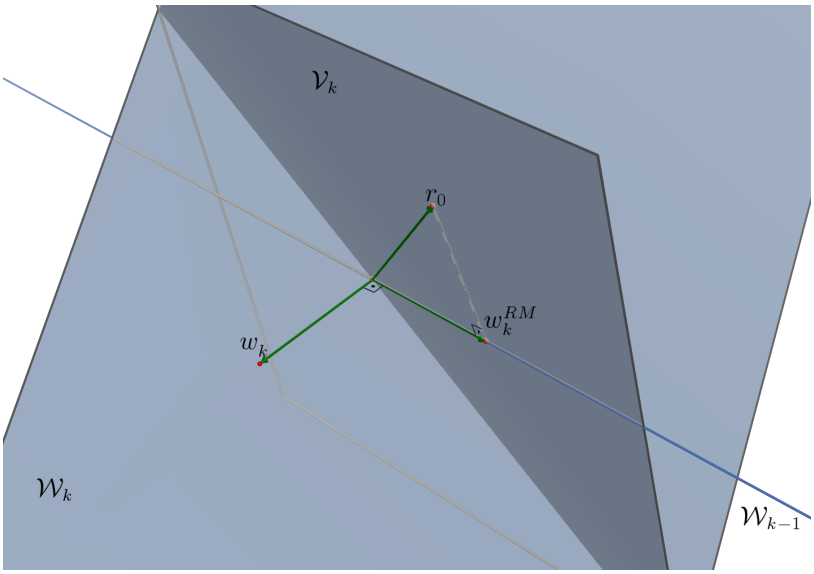

Figura 5.3: Representação gráfica dos subespaços quando w_k forma um ângulo $\theta = \pi/2$ com r_0

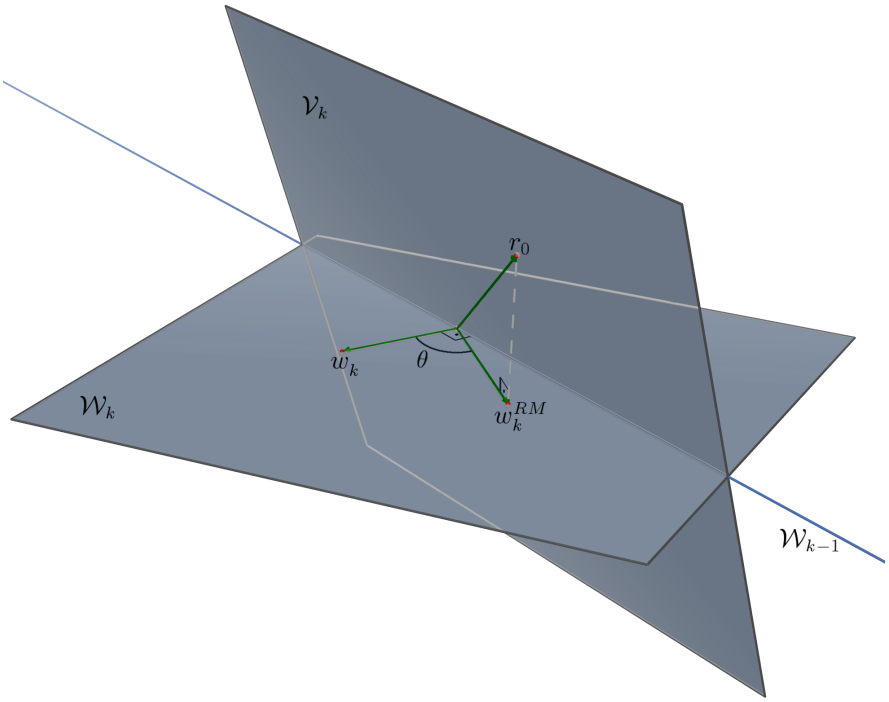

Figura 5.4: Representação gráfica dos subespaços quando w_k forma um ângulo $\theta \neq \pi/2$ com r_0

Em ambas figuras o vetor w_k é ortogonal à $\mathcal{W}_{k-1}$. Na figura 5.3 o vetor w_k também é ortogonal a r_0, e como podemos verificar, a aproximação resultante está completamente contida em $\mathcal{W}_{k-1}$. Como trata-se de uma projeção e projeções são sempre únicas, então $w_k^{RM} = w_{k-1}^{RM}$. Notemos que caso $w_k \notin \mathcal{W}_{k-1}^{\perp}$, então w_k^{RM} poderá ser diferente de w_{k-1}^{RM}, mesmo que $w_k \perp r_0$.

Ainda na proposição (5.3.1), a equação (5.3.2) mostra que a evolução dos resíduos depende unicamente do ângulo entre r_0 e w_k, e portanto, quanto melhor a escolha de w_k melhor será o aprimoramento do resíduo.

Além disso, como definimos o ângulo entre dois vetores entre $[0, \pi/2]$ em 5.2.1, (5.3.2) mostra que o resíduo nunca cresce para métodos da família RM, e que no pior caso, não haverá redução do resíduo. Este é justamente o caso em que r_0 é ortogonal a w_k, o que reforça o resultado já mostrado na proposição 5.3.2. O colorário a seguir formaliza esta idéia.

Corolário 5.3.3. *O resíduo gerado por métodos da família RM obedece à*

$$\left\|r_k^{RM}\right\| \leq \left\|r_{k-1}^{RM}\right\| \tag{5.3.15}$$

com a igualdade se e somente se $\measuredangle(r_0, w_k) = \pi/2$.

O seguinte teorema pode ser encontrado em [22, p.263]:

Teorema 5.3.4. *Dado um subespaço* $\mathcal{W}_k$ *e* $\mathcal{V}_k = \text{span}\{r_0\} + \mathcal{W}_{k-1}$, *então*

$$\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = \measuredangle(r_{k-1}^{RM}, \mathcal{W}_k) \tag{5.3.16}$$

Com este teorema, apresentaremos mais uma caracterização dos resíduos RM:

Proposição 5.3.5. *O resíduo gerado por métodos da família RM obedece à*

$$\left\|r_k^{RM}\right\| = s_k \left\|r_{k-1}^{RM}\right\| \tag{5.3.17}$$

onde

$$s_k = \text{sen}\measuredangle(\mathcal{V}_k, \mathcal{W}_k) \tag{5.3.18}$$

Demonstração. Notemos que, como $w_{k-1}^{RM} \in \mathcal{W}_k$, $(I - P_{\mathcal{W}_k})w_{k-1}^{RM} = 0$. Com isso

$$\begin{aligned} r_k^{RM} &= (I - P_{\mathcal{W}_k})r_0 - (I - P_{\mathcal{W}_k})w_{k-1}^{RM} && (5.3.19)\\ &= (I - P_{\mathcal{W}_k})(r_0 - w_{k-1}^{RM}) && (5.3.20)\\ &= (I - P_{\mathcal{W}_k})r_{k-1}^{RM} && (5.3.21) \end{aligned}$$

Tomando a norma e utilizando o mesmo princípio utilizado no teorema 5.2.3, obtemos

$$\begin{aligned} \left\|r_k^{RM}\right\| &= \left\|(I - P_{\mathcal{W}_k})r_{k-1}^{RM}\right\| && (5.3.22)\\ &= s_k \left\|r_{k-1}^{RM}\right\| && (5.3.23) \end{aligned}$$

onde $s_k = \text{sen}\measuredangle(r_{k-1}^{RM}, \mathcal{W}_k)$. Pelo teorema 5.3.4, teremos $s_k = \text{sen}\measuredangle(\mathcal{V}_k, \mathcal{W}_k)$, completando a prova. □

Vemos que o resíduo só será zero quando $\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = 0$, o que significa que $\mathcal{V}_k = \mathcal{W}_k$, o que é equivalente a dizer que $r_0 \in \mathcal{W}_k$. Se ℓ é a iteração em que isto ocorre, então $w_\ell = r_0$, que é, pela definição 5.1.3, a última iteração does métodos da família RM.

Como os resíduos são sempre menores ou iguais aos resíduos anteriores, podemos interpretar os métodos da famíla RM como métodos que aproximam os subespaços $\mathcal{V}_k$ e $\mathcal{W}_k$ gradativamente, até que coincidam.

Ainda nesta proposição concluímos que não haverá redução no resíduo em iterações consecutivas quando $\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = \pi/2$. Mas, como

$$\mathcal{V}_k = \text{span}\{r_0\} + \mathcal{W}_{k-1} \tag{5.3.24}$$

$$\mathcal{W}_k = \text{span}\{w_k\} \oplus \mathcal{W}_{k-1}, \tag{5.3.25}$$

pela própria definição de ângulo entre subespaços, os $(k-1)$-primeiros cossenos entre os ângulos canônicos serão um, e o k-ésimo ângulo canônico será exatamente $\cos\measuredangle(r_0, w_k)$, ou seja, $\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = \pi/2$ é equivalente a $\measuredangle(r_0, w_k) = \pi/2$, o que já era esperado.

5.4 Considerações Geométricas na Família RO

Similar ao que foi feito na seção anterior, faremos agora algumas considerações a respeito de métodos da família RO.

Corolário 5.4.1. *Os resíduos da família RO obdecem a seguinte relação*

$$\left\|r_k^{RO}\right\| \leq \frac{\|r_0\|}{\cos\measuredangle(\mathcal{V}_k, \mathcal{W}_k)} \tag{5.4.1}$$

Demonstração. Pelo teorema 2.9 encontrado em [22, p.260], temos

$$\left\|I - P_{\mathcal{W}}^{\mathcal{V}}\right\| = \frac{1}{\cos\measuredangle(\mathcal{V}, \mathcal{W})} \tag{5.4.2}$$

para quaisquer subespaços $\mathcal{V}, \mathcal{W} \subset \mathcal{H}$. Como

$$r_k^{RO} = (I - P_{\mathcal{W}_k}^{\mathcal{V}_k})r_0 \tag{5.4.3}$$

temos a prova completa. □

Repare que este corolário é válido para quaisquer subespaços $\mathcal{V}_k$ e $\mathcal{W}_k$. Ele nos mostra mais uma vez que a aproximação RO nem sempre existirá, pois haverá uma ruptura quando $\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = \pi/2$. De fato existe uma ligação entre o avanço dos métodos RM e a ruptura dos métodos RO, pois o método RO apresentará uma

ruptura na iteração em que o método RM não apresentar melhoras na redução do resíduo.

Corolário 5.4.2. *A aproximação RO de um $r_0 \in \mathcal{H}$ arbitrário com respeito aos subespaços $\mathcal{V}_k$ e $\mathcal{W}_k$ dados conforme a definição 5.1.1 é unicamente definida se e somente se $\measuredangle(\mathcal{V}_k, \mathcal{W}_k) \neq \pi/2$, ou equivalentemente, se e somente se*

$$\left\|r_k^{RM}\right\| \leq \left\|r_{k-1}^{RM}\right\| \tag{5.4.4}$$

Em [22] pode ser encontrada a prova não só deste corolário, como também várias outras conclusões que levam a esse mesmo corolário.

5.5 Relações Entre RO e RM

Começaremos esta seção com uma observação simples mas que resulta numa interessante relação entre a norma dos resíduos RO e RM.

Observemos que, como $w_k^{RM} \in \mathcal{W}_k$, então $(I - P_{\mathcal{W}_k}^{\mathcal{V}_k})w_k^{RM} = 0$. Com isto e utilizando o truque de somar zero, teremos

$$\begin{aligned} r_k^{RO} &= (I - P_{\mathcal{W}_k}^{\mathcal{V}_k})r_0 - (I - P_{\mathcal{W}_k}^{\mathcal{V}_k})w_k^{RM} && (5.5.1)\\ &= (I - P_{\mathcal{W}_k}^{\mathcal{V}_k})(r_0 - w_k^{RM}) && (5.5.2)\\ &= (I - P_{\mathcal{W}_k}^{\mathcal{V}_k})r_k^{RM}. && (5.5.3) \end{aligned}$$

Tomando a norma euclidiana

$$\begin{aligned} \left\|r_k^{RO}\right\|_2 &= \left\|(I - P_{\mathcal{W}_k}^{\mathcal{V}_k})r_k^{RM}\right\|_2 && (5.5.4)\\ \left\|r_k^{RO}\right\|_2 &\leq \left\|I - P_{\mathcal{W}_k}^{\mathcal{V}_k}\right\|\left\|r_k^{RM}\right\|_2 && (5.5.5) \end{aligned}$$

e utilizando novamente o teorema 2.9 em [22, p.260], obtemos

$$\cos\measuredangle(\mathcal{V}_k, \mathcal{W}_k)\left\|r_k^{RO}\right\|_2 \leq \left\|r_k^{RM}\right\|_2 \tag{5.5.6}$$

Formalizamos este resultado no seguinte corolário

Corolário 5.5.1. *O resíduo RO e RM obedecem a seguinte relação*

$$\cos\measuredangle(\mathcal{V}_k, \mathcal{W}_k)\left\|r_k^{RO}\right\|_2 \leq \left\|r_k^{RM}\right\|_2 \leq \left\|r_k^{RO}\right\|_2 \tag{5.5.7}$$

A desigualdade a esquerda é satisfeita sempre, pois pela própria definição, o resíduo RM é mínimo. Repare que este corolário é válido para subespaços $\mathcal{V}_k$ e $\mathcal{W}_k$ quaisquer, e não só os que seguem a definição 5.1.1.

O seguinte teorema nos dá várias relações entre métodos pertencentes a ambas famílias.

Teorema 5.5.2. *Sejam dados $r_0 \in \mathcal{H}$, uma sequência aninhada de subespaços $\mathcal{W}_k \in \mathcal{H}$ de dimensão k e sua correspondente sequência de subespaços residuais $\mathcal{V}_k = \text{span}\{r_0\} + \mathcal{W}_{k-1}$. Então, as aproximações RM e RO de r_0 com relação aos subespaços $\mathcal{W}_k$ e $\mathcal{V}_k$ satisfazem*

$$\left\|r_k^{RM}\right\|_2 = s_k \left\|r_{k-1}^{RM}\right\|_2, \tag{5.5.8}$$

$$\left\|r_k^{RM}\right\|_2 = s_1, s_2, \ldots, s_k \|r_0\|_2, \tag{5.5.9}$$

$$\left\|r_k^{RO}\right\|_2 = s_k \left\|r_{k-1}^{RO}\right\|_2, \tag{5.5.10}$$

$$\left\|r_k^{RO}\right\|_2 = s_1, s_2, \ldots, s_k \|r_0\|_2 / c_k \tag{5.5.11}$$

onde $s_k = \text{sen}\measuredangle(\mathcal{V}_k, \mathcal{W}_k)$ e $c_k = \cos\measuredangle(\mathcal{V}_k, \mathcal{W}_k)$.

Demonstração. Este teorema é o idêntico ao teorema 3.4 em [22, p.265], exceto pela definição de s_k e c_k. Entretanto, pelo terorema 5.3.4, temos que ambas definições são equivalentes e portanto, os teoremas são equivalentes. □

Este teorema reforça várias conclusões que já foram mostrada previamente, como a relação entre a redução do resíduo e o valor s_k e a existência da aproximação RO e o valor c_k. É notável que este teorema estabelece em (5.5.10) uma relação entre os resíduos RM e RO mais forte que a estabelecida no corolário (5.5.1). Um fato interessante que pode ser observado é que, quando o $s_k = 0$, e portanto o método RM convergiu, teremos $\left\|r_k^{RO}\right\|_2 = 0$. Com isto vemos que se o método RM converge na iteração ℓ, então o método RO converge na mesma iteração ℓ.

Em (5.5.9) podemos estabelecer uma relação entre os ângulos e a convergência do algoritmo: se o produto do seno dos ângulos entre $\mathcal{V}_j$ e $\mathcal{W}_j$ para $j = 1, 2, \ldots, k$ tende a zero, então o algoritmo terá convergência superlinear.

Teorema 5.5.3. *Assumindo o teorema 5.5.2 satisfeito, as aproximações RM e*

RO satisfazem

$$w_k^{RM} = s_k^2 w_{k-1}^{RM} + c_k^2 w_k^{RO} \tag{5.5.12}$$

$$r_k^{RM} = s_k^2 r_{k-1}^{RM} + c_k^2 r_k^{RO} \tag{5.5.13}$$

$$\frac{w_k^{RM}}{\left\|r_k^{RM}\right\|_2^2} = \sum_{j=0}^{k} \frac{w_j^{RO}}{\left\|r_j^{RO}\right\|_2^2} \tag{5.5.14}$$

$$\frac{r_k^{RM}}{\left\|r_k^{RM}\right\|_2^2} = \sum_{j=0}^{k} \frac{r_j^{RO}}{\left\|r_j^{RO}\right\|_2^2} \tag{5.5.15}$$

$$\frac{1}{\left\|r_k^{RM}\right\|_2^2} = \sum_{j=0}^{k} \frac{1}{\left\|r_j^{RO}\right\|_2^2} = \frac{1}{\left\|r_{k-1}^{RM}\right\|_2^2} + \frac{1}{\left\|r_k^{RO}\right\|_2^2} \tag{5.5.16}$$

Este teorema, cuja prova pode ser encontrada em [22, p.266], nos mostra mais algumas relações entre as aproximações e resíduos RM e RO.

Para mais algumas comparações entre métodos RO e RM, recomendamos a leitura de [24].

5.6 Subespaço de Correção

Durante todo este capítulo consideramos espaços de aproximação genéricos, apenas com a exigência de que fossem aninhados. Similar ao que foi feito na seção 3.4 do capítulo 3 vamos estabelecer mais uma definição.

Definição 5.6.1. *Consideremos o subespaço*

$$\mathcal{W}_k = A\mathcal{C}_k \tag{5.6.1}$$

onde A é uma matriz não-singular. O subespaço $\mathcal{C}_k$ será chamado de subespaço de correção.

Como supomos A não-singular, para um dado subespaço $\mathcal{W}_k$ teremos um único subespaço de correção. Além disso, para que $\mathcal{W}_k$ seja um subespaço aninhado, basta então que o subespaço $\mathcal{C}_k$ seja aninhado.

Consideremos agora um método qualquer pertencen à família RM e um método qualquer pertecente à família RO, apenas com a restrição de que ambos os métodos constroem o mesmo subespaço de aproximação $\mathcal{W}_k$ a cada iteração k. Com isso, tanto a aproximação RM quanto RO podem ser escritas como

$$w_k^{RM} = Ac_k^{RM} \quad \text{e} \quad w_k^{RO} = Ac_k^{RO}, \tag{5.6.2}$$

para algum $c_k^{RM}, c_k^{RO} \in \mathcal{C}_k$. Considerando a $\ell-$ésima iteração, teremos

$$Ac_\ell^{RM} = w_\ell^{RM} = r_0 = b - Ax_0 \tag{5.6.3}$$
$$A(x_0 + c_\ell^{RM}) = b \tag{5.6.4}$$

e analogamente

$$A(x_0 + c_\ell^{RO}) = b. \tag{5.6.5}$$

Justificamos o nome "subespaço de correção" dado aos subespaços $\mathcal{C}_k$ pois os c_k^{RM} e c_k^{RO} podem ser considerados como uma correção para a aproximação inicial x_0.

Também vemos que as correções $c_\ell^{RM}, c_\ell^{RO} \in \mathcal{C}_\ell$ são soluções para o sistema $Ac = r_0$. Como consideramos A não-singular, então $c_\ell^{RM} = c_\ell^{RO}$.

Corolário 5.6.2. *Seja ℓ o índice no qual a os método RM e RO em questão convergem. Então*

$$c_\ell^{RM} = c_\ell^{RO} \quad e \quad w_\ell^{RM} = w_\ell^{RO} \tag{5.6.6}$$
$$c_k^{RM} = s_k^2 c_{k-1}^{RM} + c_k^2 c_{k-1}^{RO} \tag{5.6.7}$$
$$x_k^{RM} = s_k^2 x_{k-1}^{RM} + c_k^2 x_{k-1}^{RO} \tag{5.6.8}$$

Demonstração. A demonstração da primeira igualdade é trivial, uma vez que ambos os métodos terminam na mesma iteração ℓ e que $w_\ell^{RO} = w_\ell^{RM} = r_0$. A segunda é obtida imediatamente da primeira igualdade do teorema 5.5.3, em conjunto com a não-singularidade de A.

Para a última igualdade temos

$$c_k^{RM} = s_k^2 c_{k-1}^{RM} + c_k^2 c_{k-1}^{RO} \tag{5.6.9}$$
$$x_k^{RM} - x_0 = s_k^2 (x_{k-1}^{RM} - x_0) + c_k^2 (x_{k-1}^{RO} - x_0) \tag{5.6.10}$$
$$x_k^{RM} - x_0 = s_k^2 x_{k-1}^{RM} + c_k^2 x_{k-1}^{RO} - \underbrace{(s_k^2 + c_k^2)}_{=1} x_0 \tag{5.6.11}$$
$$x_k^{RM} = s_k^2 x_{k-1}^{RM} + c_k^2 x_{k-1}^{RO} \tag{5.6.12}$$

□

Conforme já mencionado, teremos que o $c_\ell^{RM} = c_\ell^{RO}$ é uma solução para o problema $Ac = r_0$; portanto, para garantir que o método convirja, devemos assegurar que o subespaço de correção seja selecionado de modo que isto seja possível para um número finito de passos.

Conforme mostrado na seção 2.3 pelo teorema 2.3.3, caso tomemos $\mathcal{C}_k = \mathcal{K}_k(A, r_0)$, quando k for o grau do polinômio mínimo de r_0 com relação a A, te-

remos uma solução para $Ac = r_0$ no espaço de correção. Além disso, $\mathcal{K}_k(A, r_0)$ é um subespaço aninhado por definição, e como A é não-singular, $\mathcal{W}_k$ também será aninhado, tornando válida toda a discussão feita no capítulo 3, em especial as conclusões das seções 3.3 e 3.4. Teremos

$$\mathcal{W}_k = \text{span}\{Ar_0, A^1 r_0, \ldots, A^k r_0\} \tag{5.6.13}$$

$$\mathcal{V}_k = r_0 + \text{span}\{W_{k-1}\} = \text{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\} = \mathcal{K}_k \tag{5.6.14}$$

Com isto torna-se mais fácil a representação do método, pois o espaço residual é igual ao espaço de correção.

Outra característica interessante é que os ângulo podem ser reescritos como

$$\measuredangle(\mathcal{V}_k, \mathcal{W}_k) = \measuredangle(\mathcal{K}_k, A\mathcal{K}_k) \tag{5.6.15}$$

e o algoritmo, tanto da família RO como da família RM terminará quando

$$\mathcal{V}_k = \mathcal{W}_k \quad \text{ou seja} \quad \mathcal{K}_k = A\mathcal{K}_k, \tag{5.6.16}$$

o que acontece apenas quando k for igual ao grau do polinômio mínimo de r_0 com relação a A, conforme discutido no capítulo 2.

Uma análise mais cuidadosa mostra que, métodos que utilizam um subespaço de Krylov como subespaço de correção tem a sua convergência relacionada a estrutura da matriz A. Isto porque, conforme foi mostrado no teorema 5.5.2, a convergência dos algoritmos da família RM e RO estão diretamente relacionadas ao seno dos ângulos entre $\mathcal{V}_k$ e $\mathcal{W}_k$. Tomando $\mathcal{C}_k$ como um subespaço de Krylov, teremos que a convergência do algoritmo está relacionada apenas ao ângulo entre $A\mathcal{K}_k$ e $\mathcal{K}_k$. Como os espaços de Krylov dependem basicamente da estrutura de A, temos uma importante relação entre tal estrutura e a evolução do algoritmo.

Além das vantages mencionadas anteriormente, o espaço de Krylov é um espaço de fácil construção, onde a cada iteração um novo vetor da base de Krylov pode ser obtido através de uma apenas uma multiplicação matriz-vetor (embora, conforme discutido anteriormente, seja aconselhável a ortogonalização dos vetores utilizando, por exemplo, o método de Arnoldi). Por isso, conforme foi mostrado no capítulo 4, a maioria dos métodos utilizam o espaço de Krylov ou uma versão truncada[2] do mesmo, ou se esforçam para fazê-lo.

[2]isto é, descarta-se alguns vetores da base. Veremos alguns detalhes a respeito de truncamento no capítulo 8

5.7 Convergência de Métodos de Krylov

Vamos agora estabelecer algumas relações entre a convergência de métodos de Krylov e a estrutura da matriz de coeficientes A. Não nos prenderemos às demonstrações aqui nos limitando a mencioná-las.

Um resultado muito conhecido é que os resíduos gerados por um método RM satisfazem

$$\frac{\left\|r_k^{RM}\right\|_2}{\|r_0\|_2} \leq \min_{\substack{\mathbf{q}(t)\in\mathcal{P}_k \\ \mathbf{q}(0)=1}} \|\mathbf{q}(A)\|_2 \,, \tag{5.7.1}$$

onde $\mathbf{q}(t)$ denota o polinômio que gera os resíduos do método RM. Este resultado pode ser encontrado, por exemplo, em [25], [26], [27] e [16].

Muitas relações podem ser derivadas desta relação básica. Por exemplo, no caso da variante RM do CG (cf. [28, p.1154]), quando aplicado a matrizes simétricas e positiva definida, temos a seguinte relação

$$\frac{\left\|r_k^{CGRM}\right\|_2}{\|r_0\|_2} \leq 2\left(1 - \frac{2}{\sqrt{\kappa}+1}\right)^{i+1}, \tag{5.7.2}$$

onde $\kappa \geq 1$ é conhecido como *"número de condicionamento"* da matriz A. Esta relação mostra que se κ for próximo de 1, então a convergência será rápida.

O número de condicionamento por outro lado, é dado por

$$\kappa = \|A\|_2 \left\|A^{-1}\right\|_2 \quad \text{ou} \quad \kappa = \left|\frac{\lambda_{max}}{\lambda_{min}}\right| \tag{5.7.3}$$

onde λ_{max} e λ_{min} são, respectivamente, o maior e o menor autovalor de A em valor absoluto. Com isto, vemos que no caso especificado, se a matriz apresentar autovalores muito pequenos ou muito grandes, a convergência será ruim, sendo o caso ideal aquele no qual os autovalores estão próximos. Além disso, ainda para o CG no caso onde A é simétrica e definida positiva, citando Morgan em [28, p.1155], *"A distribuição dos autovalores também influencia a convergência. Agrupamento de autovalores é favorável"*.

Uma relação mais genérica pode ser encontrada, supondo apenas que A seja diagonalizável[3]:

$$\frac{\left\|r_k^{RM}\right\|_2}{\|r_0\|_2} \leq \min_{\substack{\mathbf{q}(t)\in\mathcal{P}_k \\ \mathbf{q}(0)=1}} \max_{i=1,\ldots,n} \|\mathbf{q}(\lambda)\|_2 \|U\|_2 \left\|U^{-1}\right\|_2 , \tag{5.7.4}$$

[3]em [29] existe uma caracterização para matrizes não diagonalizáveis

onde $A = U\Lambda U^{-1}$ é a decomposição espectral de A, $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$ e os λ_i são os autovalores de A (cf. [1, p.517]). Embora este limitante relacione a razão entre as normas dos resíduos e os autovalores de A, ele se mostrou muito ruim, pois o valor de $\|U\|_2 \left\|U^{-1}\right\|_2$ pode ser muito grande em alguns casos, e assim diminuindo a importância dos autovalores na relação.

Consideremos agora que A seja uma matriz normal, isto é

$$A^H A = A A^H. \tag{5.7.5}$$

Para matrizes normais, teremos que $\|U\|_2 \left\|U^{-1}\right\|_2 = 1$, e portanto, diretamente de (5.7.4) temos que

$$\frac{\left\|r_k^{RM}\right\|_2}{\|r_0\|_2} \leq \min_{\substack{\mathbf{q}(t)\in\mathcal{P}_k \\ \mathbf{q}(0)=1}} \max_{i=1,\ldots,n} \|\mathbf{q}(\lambda)\|_2 . \tag{5.7.6}$$

relacionando diretamente os autovalores da matriz com a convergência de métodos da família RM.

Embora as relações (5.7.6) e (5.7.4) sejam similares, conforme dito anteriormente, a relação (5.7.4) diz pouco a respeito da convergência do método, e não pode-se afirmar que nos casos de matrizes A não normais a convergência do método está relacionada ao espectro de A. O seguinte teorema, cuja prova pode ser encontrada em [30], deixa clara esta idéia.

Teorema 5.7.1. *Dada uma sequência positiva $s_0 \geq s_1 \geq \ldots \geq s_{n-1} > 0$ é possível mostrar que existe uma matriz $A \in \mathbb{C}^{n\times n}$ e vetores $x_0, b \in \mathbb{C}^n$, com $r_0 = b - Ax_0$ e $\|r_0\| = s_0$, tal que $\|r_k\| = s_k$ para $k = 1, ..., n-1$, onde r_k é o resíduo RM gerado no k-ésimo passo do método GMRES para a solução do sistema $Ax = b$. Além disto, a matriz A pode ser escolhida de modo que possua qualquer conjunto de autovalores.*

Embora este teorema especifique o método GMRES, isto já é o suficiente para mostrar que é possível escrever um exemplo onde os autovalores não estão relacionados à convergência do método.

Muitos autores também mencionam que a relação entre o espectro da matriz A e a convergência de métodos RM estão relacionadas quando a matriz é *"próxima da normalidade"*, isto é, quando

$$\left\|AA^H - A^H A\right\|_2 \tag{5.7.7}$$

é pequeno. Chamamos este valor de *"distância à normalidade"*. Em [31] há uma demonstração deste resultado.

Muitas outras relações foram propostas para matrizes não normais, mas para

todas é possível criar um exemplo numérico onde a relação (5.7.6) diz muito pouco ou nada a respeito da convergência dos métodos de Krylov, e portanto um limite razoável para uma matriz A genérica é, ainda, um problema em aberto.

Capítulo 6

Métodos com Recomeço

No capítulo 4, vimos basicamente dois tipos de métodos: os métodos completos, que garantem convergência mas requerem uma quantidade crescente de recursos, como o GMRES e o FOM, e os métodos incompletos, que utilizamum produto interno particular em $\mathcal{V}_\ell$ mas que não garantem a convergência, como o BiCGStab e o QMR.

Os métodos completos podem ser muito caros, ou mesmo impossíveis de serem realizados na prática. Estudaremos neste capítulo uma técnica muito utilizada para viabilizar a implantação destes métodos conhecida como *"recomeço"*. Embora estas estratégias possam ser empregadas tanto em métodos RO como RM, trataremos aqui apenas de métodos RM. Quando nos referirmos ao método completo, o denotaremos por *método RMC* (Resíduo Mínimo Completo).

É importante lembrar que caso A seja hermitiana, já foram apresentados métodos que garantem a convergência e cujos recursos necessários não crescem a cada iteração. É o caso, por exemplo do MINRES, ou do CG, caso além de hermitiana, A seja definida positiva. Portanto, os métodos completos só são de fato empregados para problemas não hermitianos.

Denotaremos daqui pra frente

$$r^{(0)} = r_0 \quad \text{e} \quad x^{(0)} = x_0 \tag{6.0.1}$$

6.1 Múltiplos Subespaços de Correção

Consideraremos que seja possível *"dividir"* o subespaço $\mathcal{C}$ em subespaços de dimensão menor segundo a seguinte definição

Definição 6.1.1 (Múltiplos Subespaços de Correção)**.** *Seja $\mathcal{C} \subset \mathcal{H}$, um subespaço*

qualquer. Consideremos que seja possível escrever

$$\mathcal{C} = \mathcal{C}^{(1)} \oplus \mathcal{C}^{(2)} \oplus \dots \tag{6.1.1}$$

e que a matriz A seja não-singular. Com isso, definimos

$$\mathcal{W}^{(j)} = A\mathcal{C}^{(j)} \quad e \quad \mathcal{W} = A\mathcal{C} = \mathcal{W}^{(1)} \oplus \mathcal{W}^{(2)} \oplus \dots \tag{6.1.2}$$

Por motivos de simplificação, consideraremos que cada subespaço $\mathcal{C}^{(j)}$ possui dimensão $\rho < \dim(\mathcal{C})$. Para simplificar a leitura, denotaremos

$$C_S^{(i)} = \mathcal{C}^{(1)} \oplus \mathcal{C}^{(2)} \oplus \dots \mathcal{C}^{(i)} \tag{6.1.3}$$

e

$$W_S^{(i)} = \mathcal{W}^{(1)} \oplus \mathcal{W}^{(2)} \oplus \dots \mathcal{W}^{(i)} \tag{6.1.4}$$

Em cada *ciclo*[1] i de um método com recomeço, geraremos uma nova "parte" $\mathcal{C}^{(i)}$ do subespaço de correção $\mathcal{C}$ e o seu respectivo subespaço de aproximação $\mathcal{W}^{(i)}$. Calculamos então uma nova aproximação da solução de $Ax = b$, que denotaremos por $x^{(i)}$, e descartamos toda a informação relativa a $C^{(i)}$ e $\mathcal{W}^{(i)}$, mantendo apenas $x^{(i)}$.

A vantagem dos métodos com recomeço é que podemos escolher a dimensão ρ dos $\mathcal{C}^{(j)}$ a priori, de modo que o armazenamento destes subespaços seja viável. Entretanto, como descartamos informação a cada ciclo, não será possível garantir uma solução MR com relação ao subespaço $\mathcal{C}_S^{(i)}$.

Suponha que durante o ciclo $i-1$ obtivemos uma correção $\tilde{c}^{(i-1)} \in \mathcal{U}$, não necessariamente RM, e consequentemente, um vetor $A\tilde{c}^{(i-1)} = \tilde{w}^{(i-1)} \in A\mathcal{U}$. Então, no ciclo i nós estaremos interessados apenas em calcular o "erro" na correção anterior, isto é

$$c^{(i)} = x^* - (x^{(0)} + \tilde{c}^{(i-1)}). \tag{6.1.5}$$

Entretanto, como x^* é desconhecido, tentaremos determinar $c^{(i)}$ de outra forma:

$$Ac^{(i)} = Ax^* - A(x^{(0)} + \tilde{c}^{(i-1)}) \tag{6.1.6}$$

$$Ac^{(i)} = r^{(i-1)}. \tag{6.1.7}$$

[1] um ciclo pode conter várias iterações de um método. Por exemplo, caso $\mathcal{C}$ seja um subespaço de Krylov de dimensão ρ, serão necessárias ρ iterações do método como o método de Arnoldi para a geração de tal subespaço

Como nem sempre será possível encontrar a solução exata, consideraremos aproximar este resultado, exigindo uma norma mínima, ou seja

$$Ac^{(i)} = w^{(i)} = \arg\min_{c \in \mathcal{C}^{(i)}} \left\| r^{(i-1)} - Ac \right\|_2 \tag{6.1.8}$$

$$= \arg\min_{w \in \mathcal{W}^{(i)}} \left\| r^{(i-1)} - w \right\|_2 . \tag{6.1.9}$$

Definimos então

$$\tilde{c}^{(i)} = \tilde{c}^{(i-1)} + c^{(i)} \quad \text{e} \quad \tilde{w}^{(i)} = \tilde{w}^{(i-1)} + w^{(i)}. \tag{6.1.10}$$

Podemos perceber que se $c^{(i)}$ é a solução exata do problema acima, então $x^{(0)} + \tilde{c}^{(i)}$ é a solução exata de $Ax = b$.

O algortimo 15 mostra um pseudocódigo de um método RMR (Resíduo Mínimo com Recomeço) genérico.

Algoritmo 15: Pseudocódigo de um método RMR genérico

```
   Input: A, b, x^(0)
1  i = 1;
2  while Não Convergiu do
3  |  if i > 1 then
4  |  |  Descarta todas as informações dos ciclos anteriores, exceto x^(i-1);
5  |  end if
6  |  r^(i-1) = b − Ax^(i-1);
7  |  Gerar os subespaços C^(i) e W^(i);
8  |  w^(i) = P_{W^(i)} r^(i-1);
9  |  c^(i) = A^{-1} w^(i);
10 |  x^(i) = x^(i-1) + c^(i);
11 end while
```

Ressaltamos que os algoritmos na prática não calculam $c^{(i)}$ conforme mostrado na linha 9 do algorítmo 15, mas de um modo implícito. Muitos dos algoritmos também calculam $w^{(i)}$ implicitamente.

O algorítmo 15 também mostra que nós não precisamos guardar os $\tilde{w}^{(i-1)}$ e $\tilde{c}^{(i-1)}$ explicitamente, apenas os $x^{(i-1)}$. Uma simples indução mostra que

$$x^{(i)} = x^{(0)} + \sum_{j=1}^{i-1} c^{(j)}, \tag{6.1.11}$$

e que

$$r^{(i-1)} = r^{(0)} - \sum_{j=1}^{i-1} w^{(j)}, \tag{6.1.12}$$

e consequentemente

$$\tilde{w}^{(i)} = \sum_{j=1}^{i} w^{(j)} \quad \text{e} \quad \tilde{c}^{(i)} = \sum_{j=1}^{i} c^{(j)}. \tag{6.1.13}$$

Notemos também que

$$\tilde{w}^{(i)} \in W_S^{(i)} \quad \text{e} \quad \tilde{c}^{(i)} \in \mathcal{C}_S^{(i)}, \tag{6.1.14}$$

então estamos de fato buscando uma solução em todo o espaço gerado, embora de um modo geral não possamos garantir que esta seja a solução de RM. Também é de comum uso o armazenamento de $r^{(i-1)}$ para evitar recalculá-lo na linha 6 do algoritmo 15.

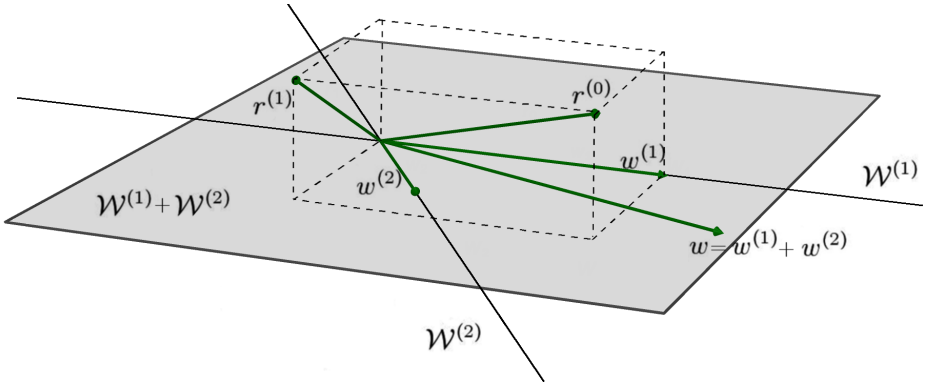

Figura 6.1: Representação gráfica dos dois primeiros ciclo de um método RMR

A figura 6.1 mostra uma representação gráfica das duas primeiras iterações de um método RMR. Repare que $\tilde{w}^{(2)} = w^{(1)} + w^{(2)}$ não é a aproximação RM de $r^{(0)}$ no subespaço $\mathcal{W}^{(1)} + \mathcal{W}^{(2)}$, o que seria desejável.

De agora em diante, denotaremos métodos com recomeço por seu nome usual seguido de (ρ)[2], onde ρ é a dimensão máxima de cada $\mathcal{C}^{(i)}$. Por exemplo, o método GMRES com recomeço será chamado de GMRES(ρ).

6.2 Subespaços de Krylov e a Estagnação

Na seção anterior consideramos apenas um caso teórico, para espaços de correção genéricos. Porém, como explicamos anteriormente, geralmente é desejável que $\mathcal{C} = \mathcal{K}_\ell(A, r^{(0)})$.

Vamos considerar o primeiro ciclo de um método RMR. Uma idéia natural é fazer $\mathcal{C}^{(1)} = \mathcal{K}_\rho(A, r^{(0)})$, já que a dimensão de $\mathcal{C}^{(1)}$ é limitada a no máximo ρ. Notemos

[2]por exemplo, GMRES(ρ)

que

$$c^{(1)} = \alpha_1 r^{(0)} + \alpha_2 A r^{(0)} + \ldots + \alpha_m A^{m-1} r^{(0)} \tag{6.2.1}$$

para algum $\{\alpha_j\}_{j=1}^{m}$. Com isto,

$$\begin{aligned} r^{(1)} &= r^{(0)} - w^{(1)} && (6.2.2)\\ &= r^{(0)} - \alpha_1 A r^{(0)} - \alpha_2 A^2 r^{(0)} - \ldots - \alpha_m A^m r^{(0)} && (6.2.3)\\ &= \mathbf{q}_1(A) r_0, && (6.2.4) \end{aligned}$$

onde $\mathbf{q}_1(t) \in \mathcal{P}_\rho$, e $\mathcal{P}_\rho$ denotam os subespaços de todos os polinômios complexos de grau até ρ.

Se tomarmos $\mathcal{C}^{(2)} = \mathcal{K}_\rho(A, r^{(1)})$, nós teremos que qualquer vetor u que pertença a $\mathcal{C}^{(2)}$ pode ser escrito como

$$\begin{aligned} u &= \beta_1 r^{(1)} + \beta_2 A r^{(1)} + \ldots + \beta_m A^{m-1} r^{(1)} && (6.2.5)\\ &= (\beta_1 I + \beta_2 A + \ldots + \beta_m A^{m-1}) r^{(1)} && (6.2.6)\\ &= (\beta_1 I + \beta_2 A + \ldots + \beta_m A^{m-1}) \mathbf{q}_1(A) r_0 && (6.2.7) \end{aligned}$$

que é um polinômio em A, no máximo de grau $2\rho - 1$. Contudo, podemos garantir que $\dim\left(\mathcal{C}^{(2)}\right) = \rho$ se e somente se $\rho \leq \ell_2$, onde ℓ_2 é o grau do polinômio mínimo de $r^{(1)}$ com relação a A. O grau deste polinômio também depende dos coeficientes $\{\alpha_j\}_{j=1}^{\rho}$, sendo que alguns dos α_j que podem ser zero. Nós formalizamos este resultado com a seguinte proposição.

Proposição 6.2.1. *Considere que $r^{(i+1)} = \mathbf{q}_{i+1}(A) r^{(i)}$ é o resíduo gerado no ciclo $i+1$ do algoritmo 15, e que cada $\mathcal{C}^{(i)} = \mathcal{K}_\rho(A, r^{(i-1)})$. Seja μ_i o grau do polinômio $\mathbf{q}_i(t)$ e $\nu_i = \min(\ell_i, \rho)$, onde ℓ_i é o grau do polinômio mínimo de $r^{(i-1)}$ com relação a A. Neste caso teremos*

$$\mathcal{C}^{(i)} \oplus \mathcal{C}^{(i+1)} = \mathcal{K}_{(\nu_i + \mu_{(i+1)})}(A, r^{(i-1)}) \tag{6.2.8}$$

Esta proposição é muito semelhante à proposição 4.4 em [25, pag.284]. Ressaltamos que caso μ_{i+1} seja menor que ρ, então a informação oriunda dos últimoss $\rho - \mu_{i+1}$ vetores de Krylov do subespaço $\mathcal{C}^{(i)}$ não foram usados durante o cálculo de $x^{(i)}$, e consequentemente $r^{(i)}$. Como descartamos toda a informção a respeito dos subespaços $\mathcal{C}^{(i)}$ durante o ciclo $i+1$, um método RMR irá gerar novamente o subespaço gerado por esses últimos vetores de Krylov cuja informação não foi aproveitada, antes de proceder para novos vetores de Krylov. Graças a esta "duplicata de informação", métodos RMR tendem a necessitar de mais iterações que métodos

RMC para atingir a convergência, mesmo considerando uma máquina com precisão infinita.

Em alguns casos, pode ser que nenhuma informação do ciclo i seja utilizada para calcular $x^{(i)}$, isto é, $x^{(i)} = x^{(i-1)}$. Isto significa que o ciclo $i+1$ irá gerar o mesmo espaço de correção gerado no ciclo i, ou seja, $\mathcal{C}^{(i+1)} = \mathcal{C}^{(i)}$. Este fenômeno é conhecido como *estagnação*.

Um caso muito simples onde isto ocorre é quando a solução exata do problema é

$$x^* = x^{(0)} + A^{\ell-1}r^{(0)}. \tag{6.2.9}$$

Neste caso $c = A^{\ell-1}r^{(0)}$. Para todo $\rho < \ell$ nós teremos $c^{(j)} = 0$, porque o vetor $A^{\ell-1}r^{(0)}$ não pertence a $\mathcal{C}^{(j)}$. Ressaltamos que $A^{\ell-1}r^{(0)}$ não pode ser escrito como uma combinação linear dos vetores de Krylov predecessores, uma propriedade que discutimos na seção 2.1.

Enfatizamos que, em aritmética de ponto flutuante, alguns outros fenômenos podem causar estagnação. Além disso, em aritmética de ponto flutuante, um método poderia simplesmente "sair da estagnação" pelo fato de o produto interno não ser exatamente zero, como assumimos aqui, e assim, mesmo quando nenhuma correção deveria ser feita, uma correção é realizada, mesmo que muito pequena. Não nos aprofundaremos aqui neste tipo de análise.

6.3 Negligência da Ortogonalização

Nesta seção mostraremos brevemente uma interpretação interessante a respeito de métodos RMR.

Métodos como o GCR [21], que já foi mostrado na seção 4.9, calculam a projeção de $r^{(0)}$ no subespaço de aproximação $\mathcal{W}$ de uma forma implícita. Podemos escrever

$$w = WW^H r^{(0)} = \sum_{j=1}^{\dim(\mathcal{W})} w_j w_j^H r^{(0)}, \tag{6.3.1}$$

onde w_j é a j-ésima coluna de W. Se formos capazes de gerar cada w_j de modo iterativo, poderemos então "atualizar" a projeção a cada passo utilizando apenas o w_j atual. O algoritmo 16 mostra um pseudocódigo para este procedimento

Reparemos que ainda precisamos armazenar todos os vetores anteriores tão somente para efeitos de ortogonalização.

A proposição 6.2.1 mostra que, em métodos RMR, num caso ótimo nós geraremos cada w_j apenas uma vez, onde $\{w_j\}_{j=1}^{\dim(\mathcal{W})}$ é uma base ortonormal para o subespaço de aproximação completo $\mathcal{W}$. Contudo, ao descartar periodicamente os w_j, podemos interpretar os métodos RMR como métodos que periodicamente "negligenciam parte

Algoritmo 16: Cálculo Iterativo de uma aproximação RM

1 $w = 0$;
2 **for** $j = 1$ *até* $\dim(\mathcal{W})$ **do**
3 | Gera w_j;
4 | Ortonormaliza w_j com relação aos vetores w_t, para todo $t < j$;
5 | $w = w + w_j w_j^H r^{(0)}$;
6 **end for**

da ortogonalização" no passo 4 do algoritmo 16.

Capítulo 7

Métodos Aumentados

Uma idéia que pode ser usada em conjunto com um método RMR é a dos *"Subespaços de Aumento"*. Esta estratégia consiste em guardar "partes" dos subespaços anteriores em um subespaço adicional, de modo que alguma informação dos ciclos anteriores sejam utilizadas nos ciclos posteriores. A forma como este subespaço adicional é construído e mantido varia em cada método.

Explicaremos algumas destas estratégias a frente. Recomendamos [25] e [26] para mais detalhes a respeito de métodos com subespaço de aumento e das várias estratégias já publicadas.

7.1 Subespaço de Aumento

Consideremos a definição a seguir.

Definição 7.1.1 (Subespaços de Aumento)**.** *Denotaremos os subespaços* $\mathcal{G}^{(i-1)}$ *e* $\mathcal{Z}^{(i-1)}$ *que dispomos no início do ciclo* i *por* "subespaço de aumento de correção" *e o subespaço por* "subespaço de aumento de aproximação"*, respectivamente. Utilizaremos livremente o nome "subespaço de aumento" quando não houver ambiguidade.*

A princípio consideraremos que a dimensão de $\mathcal{G}^{(i-1)}$ *aumentará a cada ciclo, pois estaremos adicionando informações a cada ciclo. Chamaremos os espaços*

$$\mathcal{C}_+^{(i)} = \mathcal{C}^{(i)} + \mathcal{G}^{(i-1)} \quad e \quad \mathcal{W}_+^{(i)} = \mathcal{W}^{(i)} + \mathcal{Z}^{(i-1)}. \tag{7.1.1}$$

de "subespaço de correção aumentado" *e* "subespaço de aproximação aumentado"*, respectivamente. Chamaremos simplesmente de* "subespaço aumentado" *quando não houver ambiguidade.*

Como queremos utilizar a informação tanto de $\mathcal{G}^{(i-1)}$ e de $\mathcal{C}^{(i)}$ durante o ciclo i,

consideraremos um problema diferente do proposto na seção anterior:

$$
\begin{aligned}
w_+^{(i)} &= \arg\min_{h \in \mathcal{C}_+^{(i)}} \left\| r^{(i-1)} - Ah \right\|_2 \\
&\text{ou} \\
w_+^{(i)} &= \arg\min_{\substack{c \in \mathcal{C}^{(i)} \\ g \in \mathcal{G}^{(i-1)}}} \left\| r^{(i-1)} - Ac - Ag \right\|_2
\end{aligned}
\tag{7.1.2}
$$

De uma forma geral, a solução ótima seria tomarmos $P_{\mathcal{W}_+^{(i)}}$ como aproximação. Supondo que dispomos de uma base $\tilde{W}^{(i)}$ qualquer para $\mathcal{W}^{(i)}$ e uma base $\tilde{Z}^{(i-1)}$ qualquer para $\mathcal{Z}^{(i-1)}$. Seja

$$
\tilde{W}_+^{(i)} = \begin{bmatrix} \tilde{W}^{(i)} & \tilde{Z}^{(i-1)} \end{bmatrix} \tag{7.1.3}
$$

uma base para $\mathcal{W}_+^{(i)}$ e

$$
\hat{Q}R = \tilde{W}_+^{(i)} \tag{7.1.4}
$$

uma decomposição QR econômica de $\tilde{W}_+^{(i)}$. Nada garante que $\hat{Q}$ seja uma base ortonormal para o subespaço $\mathcal{W}_+^{(i)}$ pois não podemos garantir que $\tilde{W}_+^{(i)}$ tenha posto completo[1]. Entretanto existe uma matriz P ortonormal[2] tal que

$$
Q = \hat{Q}P \tag{7.1.5}
$$

seja uma base ortonormal para $\mathcal{W}_+^{(i)}$.

Lembramos que as estratégias para encontrar uma solução para o problema (7.1.2) variam de método para método, pois o subespaço $\mathcal{G}^{(i-1)}$ pode ser construido de modo a possuir propriedades particulares.

Discutiremos a seguir algumas estratégias para a construção do subespaço de aumento.

7.2 Subespaço de Aumento Ótimo

Nesta seção proporemos uma forma de atualizar o subespaço $\mathcal{G}^{(i-1)}$. Suponha que dispomos de um resíduo $r^{(i-1)} \perp \mathcal{Z}^{(i-1)}$. Feito isto, vamos procurar um vetor $\tilde{g}^{(i)} \in \mathcal{H}$

[1] só podemos garantir que $\tilde{W}_+^{(i)}$ terá posto completo quando $\mathcal{W}^{(i)} \cap \mathcal{Z}^{(i-1)} = \emptyset$ (cf. [1, p.383])

[2] Um resultado já conhecido é que uma fatoração QR é equivalente a aplicação do método de Gram-Schmidt, e a matriz R contém os coeficientes desta ortogonalização. Caso a matriz $\tilde{W}_+^{(i)}$ não tenha posto completo, durante a ortogonalização obteremos pelo menos um vetor zero, e por consequência, $\hat{Q}$ possuirá pelo menos uma coluna igual a zero. A matriz P seria uma matriz ortonormal que exclui as colunas iguais a zero de $\hat{Q}$

apropriado, e expandir o subespaço $\mathcal{G}^{(i-1)}$ na direção de $\tilde{g}^{(i)}$, isto é, $\mathcal{G}^{(i)} = \mathcal{G}^{(i-1)} + \text{span}\{\tilde{g}^{(i)}\}$.

Como estamos tratando de métodos RM, o modo ótimo de expandir o subespaço é aquele que minimiza o resíduo. Por isto, buscaremos um $\tilde{g}^{(i)}$ tal que

$$\tilde{g}^{(i)} = \arg \min_{\substack{h \in \mathcal{H} \\ \mathcal{Z}^{(i)} = \mathcal{Z}^{(i-1)} + \text{span}\{Ah\}}} \left\| r^{(i-1)} - P_{\mathcal{Z}^{(i)}} r^{(i-1)} \right\|_2 . \tag{7.2.1}$$

Seja $Z^{(i-1)}$ uma base ortonormal para $\mathcal{Z}^{(i-1)}$. Definindo

$$\tilde{z}^{(i)} = A\tilde{g}^{(i)}, \tag{7.2.2}$$

e a ortogonalização de $\tilde{z}^{(i-1)}$

$$z^{(i)} = \frac{(I - Z^{(i-1)}(Z^{(i-1)})^H)\tilde{z}^{(i)}}{\left\|(I - Z^{(i-1)}(Z^{(i-1)})^H)\tilde{z}^{(i)}\right\|}. \tag{7.2.3}$$

Caso $z^{(i)} \neq 0$, então

$$Z^{(i)} = \begin{bmatrix} Z^{(i-1)} & z^{(i)} \end{bmatrix}$$

é uma base ortonormal para $\mathcal{Z}^{(i)}$. Repare que $z^{(i)} = 0$ significa que $\tilde{z}^{(i)} \in \mathcal{Z}^{(i-1)}$, e nenhuma direção nova é capaz de minimizar o resíduo, ou seja, já obtivemos a solução ótima do problema $Ax = b$. Podemos então, sem perda de generalidade, considerar que $z^{(i)} \neq 0$.

Ainda por (7.2.3) e da não-singularidade de A, concluímos que $z^{(i)} \in (I - Z^{(i-1)}(Z^{(i-1)})^H)A\mathcal{H} = (\mathcal{Z}^{(i-1)})^\perp$. Além disso, temos que $\left\|z^{(i)}\right\| = 1$. Isso significa que o problema (7.2.1) pode ser escrito da seguinte forma:

$$z^{(i)} = \arg \min_{\substack{h \in (\mathcal{Z}^{(i-1)})^\perp \\ \mathcal{Z}^{(i)} = \mathcal{Z}^{(i-1)} + \text{span}\{h\} \\ \|h\|_2 = 1}} \left\| r^{(i-1)} - P_{\mathcal{Z}^{(i)}} r^{(i-1)} \right\|_2 . \tag{7.2.4}$$

Vamos agora tentar reescrever o problema (7.2.4). Temos que

$$\left\| r^{(i-1)} - P_{\mathcal{Z}^{(i)}} r^{(i-1)} \right\|_2^2 = (r^{(i-1)})^H (I - P_{\mathcal{Z}^{(i)}})^H (I - P_{\mathcal{Z}^{(i)}}) r^{(i-1)} \tag{7.2.5}$$

$$= (r^{(i-1)})^H (I - P_{\mathcal{Z}^{(i)}}) r^{(i-1)} = (r^{(i-1)})^H r^{(i-1)} - (r^{(i-1)})^H P_{\mathcal{Z}^{(i)}} r^{(i-1)} \tag{7.2.6}$$

$$= \underbrace{(r^{(i-1)})^H r^{(i-1)}}_{cte.} - \left\| P_{\mathcal{Z}^{(i)}} r^{(i-1)} \right\|_2^2 \tag{7.2.7}$$

Como minimizar $\|.\|_2$ é o mesmo que minimizar $\|.\|_2^2$, e como $(r^{(i-1)})^H r^{(i-1)}$ é cons-

tante, teremos

$$\min_{\substack{h\in(\mathcal{Z}^{(i-1)})^{\perp}\\ \mathcal{Z}^{(i)}=\mathcal{Z}^{(i-1)}+\text{span}\{h\}\\ \|h\|_2=1}} -\left\|P_{\mathcal{Z}^{(i)}}r^{(i-1)}\right\|_2^2 = \max_{\substack{h\in(\mathcal{Z}^{(i-1)})^{\perp}\\ \mathcal{Z}^{(i)}=\mathcal{Z}^{(i-1)}+\text{span}\{h\}\\ \|h\|_2=1}} \left\|P_{\mathcal{Z}^{(i)}}r^{(i-1)}\right\|_2^2. \tag{7.2.8}$$

Entretanto, sabemos que

$$\begin{aligned}
\left\|P_{\mathcal{Z}^{(i)}}r^{(i-1)}\right\|_2^2 &= \left\|Z^{(i)}(Z^{(i)})^H r^{(i-1)}\right\|_2^2 = \left\|(Z^{(i)})^H r^{(i-1)}\right\|_2^2 && (7.2.9)\\
&= (r^{(i-1)})^H Z^{(i)}(Z^{(i)})^H r^{(i-1)} && (7.2.10)\\
&= (r^{(i-1)})^H Z^{(i-1)}(Z^{(i-1)})^H r^{(i-1)} + (r^{(i-1)})^H z^{(i)}(z^{(i)})^H r^{(i-1)} && (7.2.11)\\
&= \underbrace{(r^{(i-1)})^H Z^{(i-1)}(Z^{(i-1)})^H r^{(i-1)}}_{cte.} + \left\|(z^{(i)})^H r^{(i-1)}\right\|_2^2 && (7.2.12)
\end{aligned}$$

o que nos leva a

$$z^{(i)} = \arg\max_{\substack{h\in(\mathcal{Z}^{(i-1)})^{\perp}\\ \|h\|_2=1}} \left\|h^H r^{(i-1)}\right\|_2^2. \tag{7.2.13}$$

Embora o valor dentro da norma seja um escalar, pode se tratar de um valor complexo, e neste caso $\|.\|_2$ denotaria a magnitude deste valor.

Também temos que

$$h^H r^{(i-1)} = \cos\measuredangle(h, r^{(i-1)})\underbrace{\left\|r^{(i-1)}\right\|_2}_{cte.} \tag{7.2.14}$$

uma vez que a norma de h é um. Consequentemente, nosso problema se torna

$$z^{(i)} = \arg\max_{\substack{h\in(\mathcal{Z}^{(i-1)})^{\perp}\\ \|h\|_2=1}} \left\|\cos\measuredangle(h, r^{(i-1)})\right\|_2^2. \tag{7.2.15}$$

Como $h \in (\mathcal{Z}^{(i-1)})^{\perp}$ teremos então que a solução ótima para $z^{(i)}$ é a projeção de $r^{(i-1)}$ neste subespaço, isto é

$$z^{(i)} = \frac{(I - Z^{(i-1)}(Z^{(i-1)})^H)r^{(i-1)}}{\left\|(I - Z^{(i-1)}(Z^{(i-1)})^H)r^{(i-1)}\right\|_2} \tag{7.2.16}$$

significando que

$$\tilde{z}^{(i)} = r^{(i-1)} \quad \text{e} \quad \tilde{g}^{(i)} = A^{-1}r^{(i-1)} \tag{7.2.17}$$

configuram a solução ótima.

7.3 GMRESR - GMRES Recursivo

Conforme discutido na seção anterior, obteríamos um aumento ótimo do subespaço $\mathcal{G}^{(i)}$ caso possamos setar $\tilde{z}^{(i)} = r^{(i-1)}$. O problema desta estratégia é que não dispomos de $A^{-1}r^{(i-1)}$. Poderíamos, por outro lado, calcular uma aproximação para o problema

$$A\tilde{g} = r^{(i-1)} \tag{7.3.1}$$

com um número finito ρ de passos de algum método de Krylov, cujo *"chute inicial"* é o vetor zero.

Esta é a estratégia utilizada pelo método GMRESR (GMRES Recursivo) criado por van der Vorst e Vuik [32] em 1994. No ciclo i, este método calcula $c^{(i)} \in \mathcal{C}^{(i)} = \mathcal{K}(A, r^{(i-1)})$ e uma aproximação $w^{(i)}$ para $r^{(i-1)}$ em $\mathcal{W}^{(i)} = AC^{(i)}$ utilizando apenas um ciclo do método GMRES(ρ). Então expandimos o subespaço $\mathcal{G}^{(i-1)}$ na direção $c^{(i)}$, e consequentemente o subespaço $\mathcal{Z}^{(i-1)}$ na direção $w^{(i)}$, obtendo uma aproximação do subespaço de aumento ótimo.

Considerando $\tilde{G}^{(i-1)}$ uma base qualquer para $\mathcal{G}^{(i-1)}$ e $Z^{(i-1)}$ uma base ortonormal para $\mathcal{Z}^{(i-1)}$, atualizaremos estas bases da seguinte forma:

$$\tilde{G}^{(i)} = \begin{bmatrix} \tilde{G}^{(i-1)} & \tilde{g}^{(i)} \end{bmatrix} \quad \text{e} \quad Z^{(i)} = \begin{bmatrix} Z^{(i-1)} & z^{(i)} \end{bmatrix} \tag{7.3.2}$$

onde

$$z^{(i)} = \frac{(I - Z^{(i-1)}(Z^{(i-1)})^H)w^{(i)}}{\left\|(I - Z^{(i-1)}(Z^{(i-1)})^H)w^{(i)}\right\|_2} \tag{7.3.3}$$

$$\tilde{g}^{(i)} = \frac{c^{(i)} - \tilde{G}^{(i-1)}(Z^{(i-1)})^H w^{(i)}}{\left\|(I - Z^{(i-1)}(Z^{(i-1)})^H)w^{(i)}\right\|_2}. \tag{7.3.4}$$

Uma recursão mostra que, ao fazermos isto a cada ciclo, obteremos $A\tilde{G}^{(i)} = Z^{(i)}$, como no método GCR já discutido na seção 4.9.

O que na verdade estamos fazendo é a execução do GMRES(ρ), apenas com a adição de alguma informação do final do ciclo i num subespaço de aumento. Conforme foi definido em 7.1, a intenção do subespaço de aumento é guardar parte da informação de um ciclo para uso nos ciclos posteriores. No caso estamos guardando em $\mathcal{Z}^{(i)}$ as aproximações dos resíduos anteriores, e já consideramos descartada toda a informação a respeito de $\mathcal{C}^{(i)}$ e $\mathcal{W}^{(i)}$.

Agora, como de costume, mantemos o resíduo ortogonoal ao subespaço que dis-

pomos, no caso $\mathcal{Z}^{(i)}$. Isto pode ser obtido ao tomarmos

$$r^{(i)} = (I - Z^{(i)}(Z^{(i)})^H)r^{(i-1)} \tag{7.3.5}$$

$$x^{(i)} = x^{(i-1)} + \tilde{G}^{(i)}(Z^{(i)})^H r^{(i-1)} \tag{7.3.6}$$

Como o resíduo mantém ortogonalidade com "parte" dos subespaços $\mathcal{W}^{(i)}$ anteriores, temos que estamos armazenando informação dos ciclos anteriores. Podemos interpretar esta estratégia como um modo de acelerar a convergência da execução do GMRES(ρ).

O algoritmo 17 mostra o pseudocódigo para o GMRESR. Na linha 5 do algoritmo, GMRES($A, r^{(i)}, k$) significa que vamos aplicar ρ iterações do método GMRES para resolver o sistema $Au = r^{(i)}$.

Algoritmo 17: GMRES Recursivo

```
1  r_0 = b − Ax_0;
2  i = 0;
3  while ‖r_k‖_2 > tol do
4  |   i = i + 1;
5  |   Calcula c^(i) e w^(i) usando um ciclo do GMRES(A, r^(i−1), ρ);
6  |   g^(i) = c^(i);
7  |   z^(i) = w^(i);
8  |   for j = 1 até i − 1 do
9  |   |   β = ⟨z^(j), z^(i)⟩;
10 |   |   z^(i) = z^(i) − βz^(j);
11 |   |   g^(i) = g^(i) − βg^(j);
12 |   end for
13 |   z^(i) = z^(i)/‖z^(i)‖_2;
14 |   g^(i) = g^(i)/‖z^(i)‖_2;
15 |   α = ⟨r^(i−1), z^(i)⟩;
16 |   x^(i) = x^(i−1) + αg^(i);
17 |   r^(i) = r^(i−1) − αz^(i);
18 end while
```

Uma simples indução mostra que as linhas 16 e 17 do algoritmo estão de acordo com a equação (7.3.5). Repare a semelhança entre este algoritmo e o algoritmo 14. Na verdade este método foi desenvolvido como um *método aninhado*, que é basicamente um método iterativo que invoca outros métodos iterativos durante sua execução - no caso, trata-se de um *"método externo"*, no caso o GCR, invocando um *"método interno"*, no caso o GMRES(ρ). Não entraremos em detalhes a respeito desta interpretação, mas utilizaremos a nomenclatura "método interno" e "método externo" quando for conveniente.

Outra interpretação interessante a respeito deste método é a que envolve *pre-condicionadores flexíveis*. Também não entraremos em detalhes a respeito desta interpretação, mas citamos [33] e [34] para mais detalhes a respeito.

Denotaremos, daqui em diante, GMRESR(ρ) como o método GMRESR que utiliza o método GMRES(ρ) para calcular a nova direção de aumento. Mencionaremos apenas GMRESR quando não for necessário especificar o parâmetro ρ.

Porfim, a figura 7.1 mostra uma representação gráfica de um ciclo do método GMRESR.

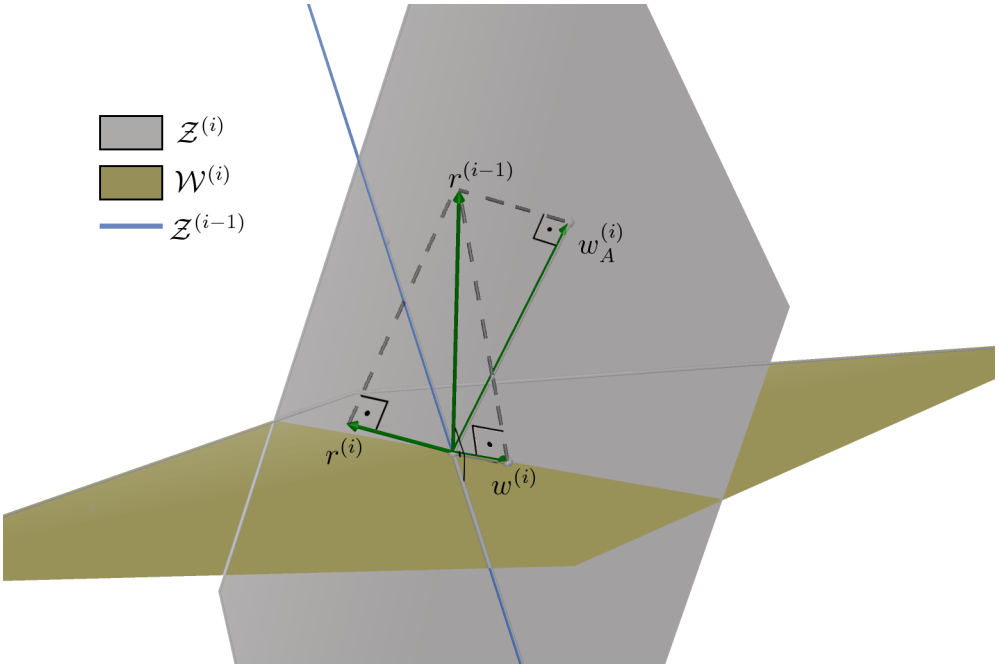

Figura 7.1: Representação gráfica de um ciclo do GMRESR

7.4 GCRO - GCR Ortogonalizado

O método GMRESR visto na seção anterior utiliza um ciclo do GMRES(ρ) para encontrar uma aproximação para $r^{(i-1)}$ num subespaço $\mathcal{W}^{(i)}$. Entretanto o que foi negligenciado é que dispomos também de um subespaço $\mathcal{Z}^{(i-1)}$, e poderíamos utilizá-lo para encontrar esta aproximação.

Estudaremos agora um método conhecido como GCRO (GCR Orotogonalizado) proposto por de Stuler [35] em 1996, baseado no GMRESR. A diferença é que este método busca uma aproximação RM de acordo com (7.1.2) de $r^{(i-1)}$ durante a geração da nova direção para expansão do subespaço de aumento.

Conforme foi discutido na seção 7.1, em um caso genérico nada garante que $\tilde{W}_+^{(i)}$ tenha posto completo, pois não temos garantia de que $\mathcal{C}^{(i)} \cap \mathcal{G}^{(i-1)} = \emptyset$. Por isso, o método GCRO constrói o subespaço $\mathcal{C}^{(i)}$ de modo a garantir que a interseção dos

subespaços será nula.

Ao invés de calcularmos uma solução aproximada para o problema $A\tilde{g} = r^{(i-1)}$ no método interno, buscaremos uma solução aproximada para

$$A_{\mathcal{Z}^{(i-1)}}\tilde{g} = r^{(i-1)} \tag{7.4.1}$$

onde

$$A_{\mathcal{Z}^{(i-1)}} = (I - P_{\mathcal{Z}^{(i-1)}})A, \tag{7.4.2}$$

utilizando um ciclo do GMRES(ρ). Para evitar ambiguidade, notaremos o subespaço de aumento obtido com isso por

$$\mathcal{C}_o^{(i)} = \mathcal{K}_\rho(A_{\mathcal{Z}^{(i-1)}}, r^{(i-1)}), \quad \text{e} \quad \mathcal{W}_o^{(i)} = A_{\mathcal{Z}^{(i-1)}}\mathcal{C}_o^{(i)} \tag{7.4.3}$$

enquanto $\mathcal{C}^{(i)} = \mathcal{K}_\rho(A, r^{(i-1)})$. Notemos que, caso possamos garantir a priori que $\mathcal{Z} \cap \mathcal{C} = \emptyset$, o mesmo resultado também pode ser obtido ao escrever

$$\mathcal{C}_o^{(i)} = (I - P_{\mathcal{Z}^{(i-1)}})\mathcal{C}^{(i)} \tag{7.4.4}$$

$$\mathcal{W}_o^{(i)} = (I - P_{\mathcal{Z}^{(i-1)}})A\mathcal{C}_o^{(i)} \tag{7.4.5}$$

Repare que ao obtermos qualquer base $\tilde{W}_o^{(i)}$ para $\mathcal{W}_o^{(i)}$, podemos garantir que $(\tilde{W}_o^{(i)})^H \tilde{Z}^{(i-1)} = 0$ onde $\tilde{Z}^{(i-1)}$ é qualquer base para $\mathcal{Z}^{(i-1)}$.

Considerando que já dispomos de uma base ortonormal $Z^{(i-1)}$ para $\mathcal{Z}^{(i-1)}$, no ciclo i geraremos uma base ortonormal[3] $W_o^{(i)}$ para $\mathcal{W}_o^{(i)}$, e com isto teremos

$$W_+^{(i)} = \begin{bmatrix} Z^{(i-1)} & W_o^{(i)} \end{bmatrix} \tag{7.4.6}$$

que é uma base ortonormal para $\mathcal{W}_+^{(i)}$. Assim podemos facilmente calcular uma aproximação RM para $r^{(i-1)}$ no subespaço aumentado, que é dada por

$$w_+^{(i)} = \begin{bmatrix} Z^{(i-1)} & W_o^{(i)} \end{bmatrix} \begin{bmatrix} (Z^{(i-1)})^H \\ (W_o^{(i)})^H \end{bmatrix} r^{(i-1)} \tag{7.4.7}$$

$$= \underbrace{Z^{(i-1)}(Z^{(i-1)})^H r^{(i-1)}}_{r^{(i-1)} \perp \mathcal{Z}^{(i-1)}} + W_o^{(i)}(W_o^{(i)})^H r^{(i-1)} \tag{7.4.8}$$

$$= w_o^{(i)} \tag{7.4.9}$$

Concluímos que, devido à construção especial do subespaço de aproximação $\mathcal{W}_o^{(i)}$, uma aproximação RM com relação a tal subespaço por si só já é uma aproximação

[3] como veremos mais a frente, esta base é na verdade gerada por um ciclo do método GMRES(ρ).

com relação ao subespaço aumentado. Para correção teremos

$$c_+^{(i)} = A^{-1}w_+^{(i)} = A^{-1}A_{\mathcal{Z}^{(i-1)}}c_o^{(i)} \tag{7.4.10}$$

$$= A^{-1}(I - Z^{(i-1)}(Z^{(i-1)})^H)Ac_o^{(i)} \tag{7.4.11}$$

$$= (I - \tilde{G}^{(i-1)}(Z^{(i-1)})^H A)c_o^{(i)}, \tag{7.4.12}$$

onde $c_o^{(i)} \in \mathcal{C}_o^{(i)}$ é a correção relativa à aproximação $w_o^{(i)}$.

Feito isto, atualizaremos o subespaço $\mathcal{Z}^{(i-1)}$. Utilizaremos a mesma estratégia do GMRESR aqui, mas graças às propriedades dos subespaços gerados pelo GCRO teremos

$$z^{(i)} = \frac{(I - Z^{(i-1)}(Z^{(i-1)})^H)w_+^{(i)}}{\left\|(I - Z^{(i-1)}(Z^{(i-1)})^H)w_+^{(i)}\right\|_2} = \frac{w_+^{(i)}}{\left\|w_+^{(i)}\right\|_2} \tag{7.4.13}$$

$$\tilde{g}^{(i)} = \frac{c_+^{(i)}}{\left\|w_+^{(i)}\right\|_2}. \tag{7.4.14}$$

Basta agora utilizar o subespaço de aumento para atualizar o resíduo e a aproximação da solução, como foi feito no GMRESR. Entretanto teremos algumas propriedades especiais novamente:

$$r^{(i)} = (I - Z^{(i)}(Z^{(i)})^H)r^{(i-1)} = r^{(i-1)} - z^{(i)}(z^{(i)})^H r^{(i-1)} \tag{7.4.15}$$

$$= r^{(i-1)} - \frac{(w_+^{(i)})^H r^{(i-1)}}{\left\|w_+^{(i)}\right\|_2^2} w_+^{(i)} \tag{7.4.16}$$

mas

$$(w_+^{(i)})^H r^{(i-1)} = (r^{(i-1)})^H W_o^{(i)}(W_o^{(i)})^H r^{(i-1)} \tag{7.4.17}$$

$$= (w_+^{(i)})^H w_+^{(i)} = \left\|w_+^{(i)}\right\|_2^2 \tag{7.4.18}$$

e portanto

$$r^{(i)} = r^{(i-1)} - w_+^{(i)} \tag{7.4.19}$$

o que imediatamente nos leva a

$$x^{(i)} = x^{(i-1)} + c_+^{(i)} \tag{7.4.20}$$

O algoritmo 18 mostra o pseudocódigo para o método externo do método GCRO (uma vez que o método interno é nada mais que o GMRES(ρ)). Mais uma vez,

notemos a semelhança entre este algoritmo e os algoritmos 17 e 14. Na linha 5 do

Algoritmo 18: GCR Ortogonalizado

1 $r^{(0)} = b - Ax^{(0)}$;
2 $i = 0$;
3 **while** $\left\|r^{(i)}\right\|_2 > tol$ **do**
4 $\quad i = i + 1$;
5 $\quad$ Calcula $c_o^{(i)}$ e $w_o^{(i)}$ usando GMRES($A_{\mathcal{Z}^{(i)}}, r^{(i-1)}, \rho$);
6 $\quad w_+^{(i)} = w_o^{(i)}$;
7 $\quad c_+^{(i)} = (I - \tilde{G}^{(i)}(Z^{(i)})^H)c_o^{(i)}$;
8 $\quad x^{(i)} = x^{(i-1)} + c_+^{(i)}$;
9 $\quad r^{(i)} = r^{(i-1)} - w_+^{(i)}$;
10 $\quad z^{(i)} = \dfrac{w_+^{(i)}}{\left\|w_+^{(i)}\right\|}$;
11 $\quad g^{(i)} = \dfrac{c_+^{(i)}}{\left\|w_+^{(i)}\right\|}$;
12 **end while**

algoritmo, GMRES($A_{\mathcal{Z}^{(i)}}, r^{(i)}, \rho$) significa que vamos aplicar no máximo ρ iterações do método GMRES para resolver o sistema $A_{\mathcal{Z}^{(i)}} u = r^{(i)}$.

Repare que ao tentar formular uma representação gráfica para o GCRO como a que foi feita para o GMRESR na figura 7.1, teremos um resultado particularmente interessante. No exemplo da figura 7.2 $r^{(i-1)} = w_o^{(i)} = w_+^{(i)}$ e portanto obtivemos a solução ótima. Isto acontece porque o espaço de busca para a aproximação $w_+^{(i)}$ já é na verdade o próprio $\mathbb{R}^3$, e portanto conseguimos encontrar a solução ótima.

Uma característica interessante ainda não mencionada é que o GCRO troca o operador não-singular A por um operador singular $A_{\mathcal{Z}^{(i)}}$ para a execução do método GMRES. Até agora consideramos A não-singular, mas quando o operador é singular, algumas características destes métodos são perdidas. O GMRES por exemplo, pode apresentar rupturas. Em [35] há uma discussão a respeito de rupturas e como evitá-las. Não abordaremos aqui este tópico, mencionando apenas que é possível remediar rupturas no GCRO, e que estas são muito raras.

Em [35] também é proposto o uso do BiGCStab ao invés do GMRES(ρ) para a solução do problema $A\tilde{g} = r^{(i-1)}$. Lembremos apenas que a princípio é possível utilizar qualquer método para resolver o sistema $A\tilde{g} = r^{(i-1)}$, mas aqui nos atentaremos apenas para o caso do GMRES(ρ).

Assim como denotamos no GMRESR, tomaremos GCRO(ρ) como o método GCRO que utiliza um ciclo do GMRES(ρ) para o cálculo da nova direção de aumento. Supressaremos o parâmetro ρ quando este não for necessário.

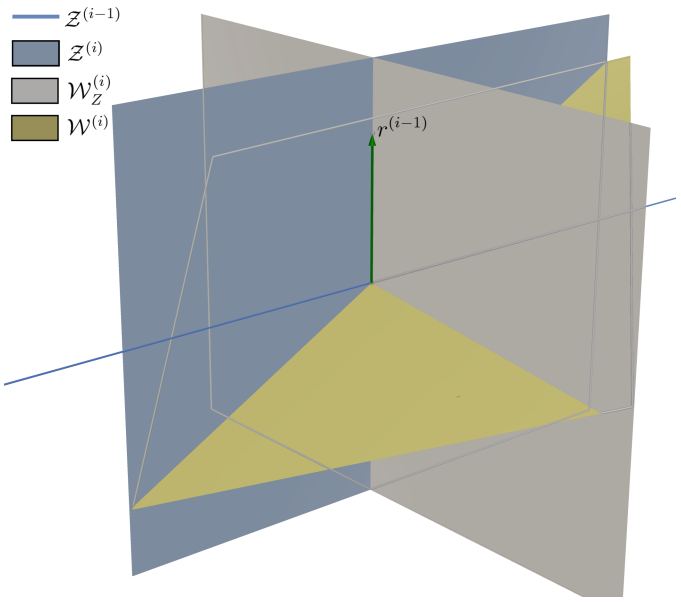

Figura 7.2: Representação gráfica de uma iteração do GCRO. Repare que como o espaço $\mathcal{W}_o^{(i)}$ tem duas dimensões e $Z^{(i-1)}$ tem uma dimensão, então o espaço $W_+^{(i)}$ é na verdade o próprio $\mathbb{R}^3$

7.5 LGMRES - GMRES Livre

Primeiramente mostraremos alguns resultados a respeito de ângulos entre resíduos no GMRES(ρ). Consideremos a seguinte definição e o seguinte teorema.

Definição 7.5.1. *Seja $r^{(i)}$ é o resíduo obtido no final do ciclo i do método GMRES(k). Denotaremos*

$$\theta^{(i,j)} = \measuredangle(r^{(i-j)}, r^{(i)}) \quad , \quad i \geq 1,\ 1 \leq j \leq i, \tag{7.5.1}$$

o que chamaremos de "ângulos sequenciais" *sempre que $j = 1$ e* "ângulos alternados" *sempre que $t > 1$. No caso dos ângulos sequenciais, os representaremos apenas como θ^i, uma vez que o índice j é imediato.*

Teorema 7.5.2 (Ângulos Entre Resíduos do GMRES(k))**.** *Seja $r^{(i)}$ o resíduo do i-ésimo ciclo do GMRES(k). Então*

$$\cos\theta^{(i,j)} = \frac{\left\|r^{(i)}\right\|_2}{\left\|r^{(i-j)}\right\|_2} - \frac{1}{\left\|r^{(i)}\right\|_2\left\|r^{(i-j)}\right\|_2}\left(\sum_{l=1}^{j-1}\sum_{t=1}^{l}\left\langle w^{(i-l+t)}, w^{(i-l)}\right\rangle\right). \tag{7.5.2}$$

Mais notavelmente, teremos

$$\cos\theta^{i} = \frac{\left\|r^{(i)}\right\|_2}{\left\|r^{(i-1)}\right\|_2} \quad e \quad \cos\theta^{(i,2)} = \frac{\left\|r^{(i)}\right\|_2}{\left\|r^{(i-2)}\right\|_2} - \frac{\left\langle w^{(i)}, w^{(i-1)}\right\rangle}{\left\|r^{(i)}\right\|_2\left\|r^{(i-2)}\right\|_2} \tag{7.5.3}$$

Demonstração. Este teorema é uma forma mais geral dos teoremas 4 e 5 que podem ser encontrados em [36, p.969-970]. A prova pode ser feita por indução. Lembremos que

$$r^{(i)} = r^{(i-1)} - w^{(i)} = (I - P_{\mathcal{W}^{(i)}})r^{(i-1)}, \quad w^{(i)} \in \mathcal{W}^{(i)}, \quad r^{(i)} \perp \mathcal{W}^{(i)}. \tag{7.5.4}$$

Para $j = 1$ teremos

$$\left\langle r^{(i)}, r^{(i-1)} \right\rangle = \left\langle r^{(i)}, r^{(i)} + w^{(i)} \right\rangle = \left\langle r^{(i)}, r^{(i)} \right\rangle + \underbrace{\left\langle r^{(i)}, w^{(i)} \right\rangle}_{=0}. \tag{7.5.5}$$

Pela definição de cosseno entre vetores vemos que vale para $j = 1$. Supondo que seja válido para $j = \phi$, para $j = \phi + 1$ teremos

$$\left\langle r^{(i)}, r^{(i-\phi-1))} \right\rangle = \left\langle r^{(i)}, r^{(i-\phi)} + w^{(i-\phi)} \right\rangle = \left\langle r^{(i)}, r^{(i-\phi)} \right\rangle + \left\langle r^{(i)}, w^{(i-\phi)} \right\rangle \tag{7.5.6}$$

$$= \left\langle r^{(i)}, r^{(i-\phi)} \right\rangle + \left\langle r^{(i-1)} - w^{(i)}, w^{(i-\phi)} \right\rangle \tag{7.5.7}$$

$$= \left\langle r^{(i)}, r^{(i-\phi)} \right\rangle + \underbrace{\left\langle r^{(i-\phi)}, w^{(i-\phi)} \right\rangle}_{=0} - \left\langle w^{(i-\phi+1)}, w^{(i-\phi)} \right\rangle \ldots \tag{7.5.8}$$

$$- \left\langle w^{(i-1)}, w^{(i-\phi)} \right\rangle - \left\langle w^{(i)}, w^{(i-\phi)} \right\rangle \tag{7.5.9}$$

$$= \left\langle r^{(i)}, r^{(i-\phi)} \right\rangle - \sum_{t=1}^{\phi} \left\langle w^{(i-\phi+t)}, w^{(i-\phi)} \right\rangle \tag{7.5.10}$$

$$= \left\langle r^{(i)}, r^{(i)} \right\rangle - \left(\sum_{l=1}^{\phi-1} \sum_{t=1}^{l} \left\langle w^{(i-l+t)}, w^{(i-l)} \right\rangle \right) - \sum_{t=1}^{\phi} \left\langle w^{(i-\phi+t)}, w^{(i-\phi)} \right\rangle \tag{7.5.11}$$

$$= \left\langle r^{(i)}, r^{(i)} \right\rangle - \left(\sum_{l=1}^{\phi} \sum_{t=1}^{l} \left\langle w^{(i-l+t)}, w^{(i-l)} \right\rangle \right) \tag{7.5.12}$$

A definição de cosseno do ângulo entre dois vetores completa a prova. □

Notemos a semelhança entre a relação

$$\cos \theta^i = \frac{\left\| r^{(i)} \right\|_2}{\left\| r^{(i-1)} \right\|_2} \tag{7.5.13}$$

e a equação 5.5.9 do teorema 5.5.2 no capítulo 5. Tal relação está diretamente relacionada à convergência de métodos RM, isto é, quanto menor a razão entre a norma dos resíduos consecutivos, mais rápida será a convergência.

Num caso ideal, o mesmo resultado seria válido para o GMRES(ρ), isto é, o cosseno dos θ^i estariam diretamente relacionados à convergência do algoritmo, e portanto, quando estes ângulos são próximos de $\pi/2$, a convergência deveria ser rápida. O problema é que, devido a perda de informação, o GMRES(ρ), em alguns

casos, apresenta θ^i muito próximos de $\pi/2$, mas uma convergência ruim.

O que foi observado por Baker et. al [36] é que o método GMRES(ρ) em alguns casos apresenta um comportamento cíclico. Foram feitas algumas comparações, e percebeu-se que em alguns casos, os ângulos alternados eram muito pequenos, embora os ângulos sequenciais fossem muito próximos de $\pi/2$. A figura 7.3 ilustra esta situação.

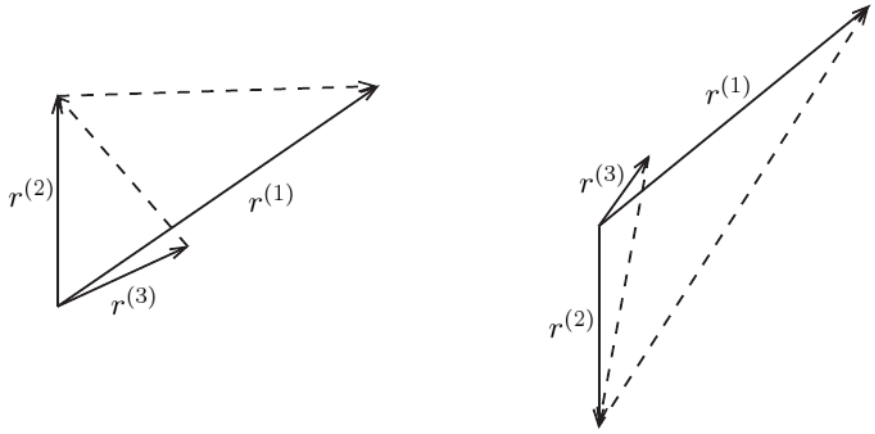

Figura 7.3: Duas representações gráficas do comportamento cíclico dos resíduos no GMRES(ρ)

Para evitar este comportamento, foi criado o método chamado de LGMRES (GMRES Livre) cuja principal estratégia de aceleração é evitar tais ciclos, fazendo com que os ângulos alternados também sejam mais próximos de $\pi/2$.

Trata-se basicamente do método GMRES(ρ) com um subespaço de aumento. O GMRES(ρ) gera como de costume, no i-ésimo ciclo, uma decomposição de Arnoldi

$$AV_\rho^{(i)} = V_{\rho+1}^{(i)} \tilde{H}_\rho^{(i)}, \tag{7.5.14}$$

onde $V_\rho^{(i)}$ é uma base ortonormal para $\mathcal{C}^{(i)}$. Então, o subespaço $\mathcal{Z}^{(i-1)}$ é gerado de modo que

$$\mathcal{Z}^{(i-1)} = (I - V_{\rho+1}^{(i)}(V_{\rho+1}^{(i)})^H) A\mathcal{G}^{(i)}. \tag{7.5.15}$$

Uma base ortonormal $Z^{(i-1)}$ para $\mathcal{Z}^{(i-1)}$ é construída e descartada a cada ciclo. Mantemos apenas uma base $\tilde{G}^{(i-1)}$ para $\mathcal{G}^{(i-1)}$. Tomando

$$\mathcal{W}_+^{(i)} = \mathcal{W}^{(i)} + \mathcal{Z}^{(i-1)}, \tag{7.5.16}$$

calculamos a aproximação RM $w_+^{(i)} \in \mathcal{W}_+^{(i)}$. Para isso, o LGMRES utiliza um truque:

a base para $\mathcal{W}_+^{(i)}$ é gerada de modo que uma relação de Arnoldi seja satisfeita:

$$A\left[V_k^{(i)} \quad \tilde{G}^{(i-1)}\right] = \left[V_{k+1}^{(i)} \quad Z^{(i-1)}\right] \tilde{H}_L^{(i)}. \tag{7.5.17}$$

Após o cálculo da aproximação RM $w_+^{(i)}$, o subespaço $\mathcal{G}^{(i-1)}$ é aumentado de uma forma bem simples. No LGMRES teremos

$$\mathcal{G}^{(i)} = \mathcal{G}^{(i-1)} + \text{span}\{c_+^{(i)}\} \quad \text{e} \quad \tilde{G}^{(i)} = \left[c_+^{(i)} \quad \tilde{G}^{(i-1)}\right] \tag{7.5.18}$$

onde $c_+^{(i)} \in \mathcal{C}_+^{(i)}$ é a correção gerada no ciclo i. Uma vez que a base para $\mathcal{Z}^{(i)}$ é gerada novamente a cada ciclo, não há necessidade de efetuar nenhum tipo de ortogonalização com relação ao subespaço de aumento.

Uma diferença importante entre o LGMRES e os outros dois métodos apresentados é que o LGMRES não atualiza o resíduo com relação ao novo subespaço de aumento, e sim com relação ao subespaço aumentado $\mathcal{W}_+^{(i)}$. Assim, a atualização é feita de uma forma bem simples:

$$x^{(i)} = x^{(i-1)} + c_+^{(i)} \quad \text{e} \quad r^{(i)} = r^{(i-1)} - w_+^{(i)} \tag{7.5.19}$$

O algoritmo 19 mostra um pseudocódigo para o LGMRES, tanto o método interno como o externo.

Baker et. al não propuseram nenhuma demonstração teórica de que o LGMRES de fato aumenta os ângulos alternados em relação aos ângulos alternados do GMRES(ρ) em seu artigo [36]. Ao invés disso, há uma discussão a respeito de resultados numéricos, que apontam para tal resultado.

O seguinte resultado pode ser encontrado em [36], página 970.

Teorema 7.5.3 (Ângulos Alternados do LGMRES)**.** *Sejam $r_L^{(i)}$ e $r_L^{(i-j)}$ os resíduos gerados pelo LGMRES nos ciclos i e $i-j$ respectivamente. Seja*

$$\theta_L^{(i,j)} = \measuredangle(r_L^{(i-j)}, r_L^{(i)}). \tag{7.5.20}$$

Então teremos

$$\cos\theta_L^{(i,j)} = \frac{\left\|r_L^{(i)}\right\|_2}{\left\|r_L^{(i-j)}\right\|_2}. \tag{7.5.21}$$

Sem muito esforço podemos também obter que

$$\prod_{l=i-j+1}^{j-1} \cos\theta_L^{(l)} = \cos\theta_L^{(i,j)}. \tag{7.5.22}$$

Algoritmo 19: GMRES Livre

$i = 0, \quad \mathcal{G}^{(1)} = \emptyset;$
$r^{(0)} = b - Ax^{(0)};$
while $\left\|r^{(i)}\right\|_2 > tol$ **do**
  $i = i + 1;$
  $\beta = \left\|r^{(i-1)}\right\|_2;$
  $v_1^{(i)} = r^{(i-1)}/\beta;$
  $s = k + \dim\left(\mathcal{G}^{(i)}\right);$
  for $j = 1$ *até* s **do**
    $u = \begin{cases} Av_j^{(i)} & \text{se } j \leq k \\ Ag_{j-k}^{(i)} & \text{se } j > k \end{cases};$
    for $l = 1$ *até* j **do**
      $\tilde{h}_{l,j}^{(i)} = \left\langle v_l^{(i)}, u \right\rangle;$
      $u = u - \tilde{h}_{l,j}^{(i)} v_l^{(i)};$
    end for
    $\tilde{h}_{j+1,j}^{(i)} = \|u\|_2;$
    define $\begin{cases} v_{j+1} = u/\tilde{h}_{j+1,j}^{(i)} & \text{se } j \leq k \\ z_{j-k} = u/\tilde{h}_{j+1,j}^{(i)} & \text{se } j > k \end{cases};$
  end for
  resolve $\min\left\|\beta e_1 - \tilde{H}_L^{(i)} y\right\|_2;$
  $c_+^{(i)} = \left[V_k^{(i)} \quad \tilde{G}^{(i-1)}\right] y;$
  $w_+^{(i)} = \left[V_{k+1}^{(i)} \quad Z^{(i-1)}\right] \tilde{H}_L^{(i)} y;$
  $\tilde{G}^{(i)} = \left[c_+^{(i)} \quad \tilde{G}^{(i-1)}\right];$
  $x^{(i)} = x^{(i-1)} + c_+^{(i)};$
  $r^{(i)} = r^{(i-1)} - w_+^{(i)};$
end while

Rediscutindo o resultado do teorema 5.5.2 do capítulo 5, isso nos mostra que podemos relacionar o produto do cosseno dos $\theta_L^{(i)}$ com o cosseno de algum ângulo alternado. Deste modo torna-se razoável que o aumento dos ângulos alternados causado pelo LGMRES também acelere a sua convergência.

Embora não entremos neste assunto, em [36] também é discutida a semelhança entre este método e o GMRES-E [28], que será brevemente discutido na seção 7.6.

Novamente, tomaremos LGMRES(ρ) como o método LGMRES que utiliza um ciclo do GMRES(ρ) para o cálculo da nova direção de aumento. Vamos suprimir o parâmetro ρ quando este não for necessário.

7.6 GMRES-E - GMRES Aumentado com Autovetores

Conforme foi mencionado na seção 5.7, caso a matriz A tenha a "distância à normalidade" pequena, então o espectro da mesma pode estar relacionado à convergência do algoritmo RM. Também vimos que quando o espectro da matriz está relacionada à convergência, os menores autovalores em valor absoluto "atrapalham" a convergência.

Baseado neste princípio, poderíamos tentar "retirar" os autovalores pequenos em valor absoluto do espectro de A. Com isto estaríamos "mudando o número de condicionamento" de A, que tenderia diminuir, e assim supostamente aceleraríamos a convergência.

Definição 7.6.1. *Sejam $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ os autovalores de A. Supondo que os autovalores λ_j, $1 \leq j < i$ já foram "excluídos" do espectro de A, denotaremos o "número de condicionamento efetivo" da matriz A o valor*

$$\kappa_i = \frac{\lambda_n}{\lambda_i}. \tag{7.6.1}$$

Morgan propôs em [28] que, para retirar estes autovalores do espectro de A, bastaria adicioná-los ao subespaço de correção, e assim, caso a matriz tenha a distância à normalidade pequena, a convergência será regida pelo espectro alterado. O seguinte teorema, que pode ser encontrado em [28, p.1156], mostra isto.

Teorema 7.6.2. *Suponha que A tenha uma decomposição espectral $A = U\Lambda U^{-1}$, onde todos os autovalores de A são positivos e reais. Suponhamos que $\hat{x}$ seja uma solução de mínimo resíduo encontrada no subespaço $\mathcal{C}_+ = \mathcal{C} + \mathcal{G}$ onde*

$$\mathcal{C} = \text{span}\{r_0, Ar_0, ..., A^{i-1}r_0\} \quad e \quad \mathcal{G} = \text{span}\{u_1, u_2, \ldots, u_k\}, \tag{7.6.2}$$

u_i é a i-ésima coluna de U e $r_0 = b - Ax_0$. Com isso, teremos

$$\frac{\|r\|_2}{\|r_0\|_2} \leq 2\|U\|_2\left\|U^{-1}\right\|_2 \left(1 - \frac{2}{\sqrt{\kappa_{k+1}} + 1}\right)^i, \tag{7.6.3}$$

onde $r = r_0 - A\hat{x}$ é o resíduo associado à aproximação $\hat{x}$.

Com este teorema concluímos que, caso $\|U\|_2\left\|U^{-1}\right\|_2$ seja pequeno então a convergência dependerá de κ_{k+1}: quanto mais próximo de 1 for este valor, melhor será a convergência.

Morgan admite, entretanto, que mesmo que utilizemos a estratégia acima podemos ainda ter uma convergência lenta. É o caso quando a matriz for indefinida ou

extremamente não simétrica[4], ou quando os autovalores não forem igualmente espaçados (digamos que os $k+1$ menores autovalores estão tão próximos que mesmo após retirá-los do espectro de A, continuamos com $\kappa_{k+1} \approxeq \kappa_1$, e portanto, a convergência não terá sido alterada).

O primeiro método a utilizar esta estratégia foi o GMRES-E [28], que aproxima os menores autovalores e autovetores de A. Este método procede da seguinte forma. Durante a iteração i, dispondo do subespaço de aumento $\mathcal{G}^{(i-1)}$, geraremos um subespaço de aproximação de modo que a seguinte relação seja satisfeita

$$A \begin{bmatrix} V_k^{(i)} & \tilde{G}^{(i-1)} \end{bmatrix} = \begin{bmatrix} V_{k+1}^{(i)} & Z^{(i-1)} \end{bmatrix} \tilde{H}_E^{(i)}, \tag{7.6.4}$$

exatamente como no LGMRES. Após isto, o resíduo e a aproximação da solução são atualizados exatamente como no GMRES, para então buscarmos um vetor para adicionar no subespaço de aumento. Este vetor é buscado calculando o "vetor harmônico de Ritz" associado ao menor "valor harmônico de Ritz" de A com relação ao subespaço $\mathcal{C}_+^{(i)}$, em valor absoluto. Para mais informações a respeito de valores e vetores harmônicos de Ritz e sua relação com espaços de Krylov, recomendamos a leitura de [37].

Um pseudocódigo para este método pode ser encontrado no algoritmo 20.

Lembramos então que este método pode não acelerar a convergência do GMRES(ρ), que é o principal objetivo dos métodos com subespaço de aumento, pois conforme foi mencionado, a convergência depende de uma série de fatores, e não somente do espectro de A.

Um outro problema não mencionado a respeito do GMRES-E é que este método calcula apenas uma aproximação dos autovetores e autovalores, que podem também estar sujeitos a erros.

Também notamos que existe um método mais barato para a implantação deste algoritmo, utilizando um processo conhecido como "processo de arnoldi com recomeço implícito". Este método se chama GMRES-IR [38], também desenvolvido por Morgan. Não entraremos em detalhes a respeito deste método, apenas mencionaremos que é matematicamente equivalente ao método GMRES-E.

7.7 Considerações a Respeito dos Métodos Aumentados

É notável que os métodos GMRESR, GCRO e LGMRES aumentam o subespaço $\mathcal{G}^{(i)}$ na direção $c_+^{(i)}$. No LGMRES e no GMRESR temos também que $c_+^{(i)}$ é o que

[4] isto é, quando o valor $\left\|A - A^H\right\|_2$ for grande

Algoritmo 20: GMRES-E - GMRES Aumentado com Autovetores

$i = 0, \quad \mathcal{G}^{(1)} = \emptyset;$
$r^{(0)} = b - Ax^{(0)};$
while $\left\|r^{(i)}\right\|_2 > tol$ **do**
  $i = i + 1;$
  $\beta = \left\|r^{(i-1)}\right\|_2;$
  $v_1^{(i)} = r^{(i-1)}/\beta;$
  $s = k + \dim\left(\mathcal{G}^{(i)}\right);$
  for $j = 1$ *até* s **do**
    $u = \begin{cases} Av_j^{(i)} & \text{se } j \leq k \\ Ag_{j-k}^{(i)} & \text{se } j > k \end{cases};$
    for $l = 1$ *até* j **do**
      $\tilde{h}_{l,j}^{(i)} = \left\langle v_l^{(i)} u \right\rangle;$
      $u = u - \tilde{h}_{l,j}^{(i)} v_l^{(i)};$
    end for
    $\tilde{h}_{j+1,j}^{(i)} = \|u\|_2;$
    define $\begin{cases} v_{j+1} = u/\tilde{h}_{j+1,j}^{(i)} & \text{se } j \leq k \\ z_{j-k} = u/\tilde{h}_{j+1,j}^{(i)} & \text{se } j > k \end{cases};$
  end for
  resolve $\min\left\|\beta e_1 - \tilde{H}_E^{(i)} y\right\|_2;$
  $c_+^{(i)} = \left[V_k^{(i)} \quad \tilde{G}^{(i-1)}\right] y;$
  $w_+^{(i)} = \left[V_{k+1}^{(i)} \quad Z^{(i-1)}\right] \tilde{H}_E^{(i)} y;$
  calcula u_i e θ_i, respectivamente, o vetor e o valor harmônico de Ritz de A com relação a $\mathcal{C}_+^{(i)}$;
  $\tilde{G}^{(i)} = \left[u_i \quad \tilde{G}^{(i-1)}\right];$
  $x^{(i)} = x^{(i-1)} + c_+^{(i)};$
  $r^{(i)} = r^{(i-1)} - w_+^{(i)};$
end while

podemos chamar de *"aproximação do erro em $x^{(i-1)}$"*. Isto é, supondo que x^* é a solução exata do problema $Ax = b$, teremos

$$e^{(i)} = x^* - x^{(i-1)} \tag{7.7.1}$$

que obviamente não dispomos. Multiplicando por A obtemos então

$$Ae^{(i)} = b - Ax^{(i-1)} = r^{(i-1)}. \tag{7.7.2}$$

Portanto, ao tentar solucionar o problema $Au = r^{(i)}$ estamos apenas buscando uma aproximação do erro. No caso do GCRO, foi simplesmente observado que, como já calculamos tais aproximações para $r^{(j)}$, $j < i$, seria mais interessante forçar uma ortogonalidade entre o subespaço gerado pelas aproximações anteriores e a aproximação atual, para evitar *"buscar duas vezes no mesmo lugar"*.

Isso nos mostra que a grande diferença entre tais métodos esta relacionada à ortogonalização tomada. No caso do GMRESR nenhuma ortogonalização é feita, enquanto no GCRO temos o que foi chamado por Eiermann et. al em [25] de *ortogonalização completa*. No caso do LGMRES, o novo subespaço de correção não é ortogonal ao subespaço de aumento, ao contrário do GCRO. O que é feito é simplesmente uma adição do subespaço de aumento no subespaço de correção novo. Embora não haja uma ortogonalização, o LGMRES também difere do GMRESR pelo fato de que o primeiro busca uma solução no espaço $\mathcal{C}_+^{(i)}$, e não somente no subespaço $\mathcal{C}^{(i)}$.

Na verdade, o primeiro método aumentado proposto foi o GMRES-E em 1995, enquanto o GMRESR havia sido proposto como um método aninhado em 1992, mas ambas interpretações são equivalentes. Quanto à semelhança GMRES-E e o LGMRES, não trata-se mera coincidência. Em uma das possíveis interpretações, o LGMRES foi proposto como uma alternativa ao subespaço de aumento do GMRES-E, que pode não funcionar em todos os casos. Como o subespaço de aumento gerado pelo GCRO é ótimo, torna-se natural a idéia de tentar utilizar este subespaço de aumento em métodos semelhantes.

Outro fato interessante é que em todos estes métodos (incluindo o GMRES-E), o subespaço de aumento é na verdade um subespaço do espaço de Krylov $K(A, r^{(0)})$. No caso do LGMRES, GCRO e GMRESR esta idéia é clara, uma vez que o subespaço é sempre aumentado com um vetor pertencente a um espaço de Krylov. Podemos contudo provar que o subespaço de aumento gerado pelo GMRES-E também é um subespaço de um espaço de Krylov. Tal demonstração pode ser encontrada em [25], por exemplo.

Com a seguinte definição estabeleciremos algumas bases e retiraremos alguns índices para facilitar a leitura das considerações que se seguem.

Definição 7.7.1 (Bases Biortonormais)**.** *Consideremos que* $\mathcal{G}^{(i)} \cap \mathcal{C}^{(i+1)} = \emptyset$. *Sejam*

$$\breve{Z} = \left[\breve{z}_1, \breve{z}_2, \ldots, \breve{z}_i\right], \qquad \breve{W} = \left[\breve{w}_1, \breve{w}_2, \ldots, \breve{w}_\rho\right]$$

onde $\breve{Z} \in \mathbb{C}^{n\times i}$ *e* $\breve{W} \in \mathbb{C}^{n\times\rho}$ *são bases biortonormais para* $\mathcal{Z}^{(i)}$ *e* $\mathcal{W}^{(i+1)}$ *respectivamente, de modo que a matriz diagonal* $\Sigma := (\breve{Z})^H\breve{W}$ *apresente elementos da diagonal principal* $\sigma_1, \sigma_2, \ldots, \sigma_p$, $p = \min(i, \rho)$ *em ordem não-crescente.*

Com isso

$$\breve{W}_o = (I - \breve{Z}\breve{Z}^H)\breve{W}\Gamma^{-1}, \qquad \Gamma^2 = I - \Sigma^H\Sigma$$
$$= \left[\breve{w}_1^o, \breve{w}_2^o, \ldots, \breve{w}_\rho^o\right]$$

é uma base ortonormal para o subespaço $\mathcal{W}_o^{(i+1)} = P_{(\mathcal{Z}^{(i)})^\perp}\mathcal{W}^{(i+1)}$. *Temos que (cf. [39, capítulo 4.5])*

$$\sigma_j = \cos\theta_j \quad e \quad \gamma_j = \text{sen}\theta_j$$

onde θ_j *é o* j*-ésimo maior ângulo canônico entre* $\mathcal{Z}^{(i)}$ *e* $\mathcal{W}^{(i+1)}$ *e os* γ_j *são os elementos da diagonal principal de* Γ. *Como supomos que* $\mathcal{Z}^{(i)} \cap \mathcal{W}^{(i+1)} = \emptyset$, *teremos* $0 < \gamma_j \leq 1$.

A próxima definição nos ajudará a efetuar uma comparação entre os resíduos obtidos com vários tipos de aproximações. Reforçamos que alguns desses resíduos não são de fato calculados em nenhum algoritmo e estão aqui apresentados apenas para entendimento de resultados teóricos.

Definição 7.7.2 (Resíduos dos Subespaços)**.** *Seja* $r^{(i-1)} \in \mathcal{H}$. *Denotaremos*

$$w^{(i)} = P_{\mathcal{Z}^{(i)}} r^{(i-1)} \qquad r^{(i)} = r^{(i-1)} - w^{(i)} \tag{7.7.3}$$
$$w = P_{\mathcal{W}^{(i+1)}} r^{(i)} \qquad r = r^{(i)} - w \tag{7.7.4}$$
$$w_o = P_{\mathcal{W}_o^{(i+1)}} r^{(i)} \qquad r_o = r^{(i)} - w_o \tag{7.7.5}$$

Como já foi discutido, a aproximação w_o em geral[5] é melhor que a aproximação w. O teorema mostra uma comparação entre essas duas aproximações. A demonstração pode ser encontrada em [25, p.271], teorema 2.7.

Teorema 7.7.3. *Considerando as definições 7.7.1 e 7.7.2, temos que*

$$w - w_o = \sum_{j=1}^{p} \left\langle r^{(i)}, \breve{w}_j^o \right\rangle \sigma_j \left(\gamma_j \breve{w}_j - \sigma_j \breve{w}_j^o\right) \tag{7.7.6}$$

$$\|w - w_o\|^2 = \sum_{j=1}^{p} \sigma_j^2 \left| \left\langle r^{(i)}, \breve{w}_j^o \right\rangle \right|^2 \tag{7.7.7}$$

Este teorema relaciona a norma da diferença das aproximações com o cosseno dos ângulos canônicos entre os subespaços. Isto mostra que *"quanto mais ortogonais"* forem os subespaços $\mathcal{W}^{(i+1)}$ e $\mathcal{Z}^{(i)}$, menor será a diferença entre as aproximações, o que é um resultado já esperado. Reparemos que este mesmo resultado relaciona os

[5] num caso extremo teremos $w = w_o$. Isto ocorrerá se e somente se $\mathcal{W}^{(i+1)} = \mathcal{W}_o^{(i+1)}$

resíduos, pois

$$w - w_o = r_o - r \tag{7.7.8}$$

A figura 7.4 mostra uma representação gráfica dos diferentes resíduos e aproximações estabelecidos na definição 7.7.2.

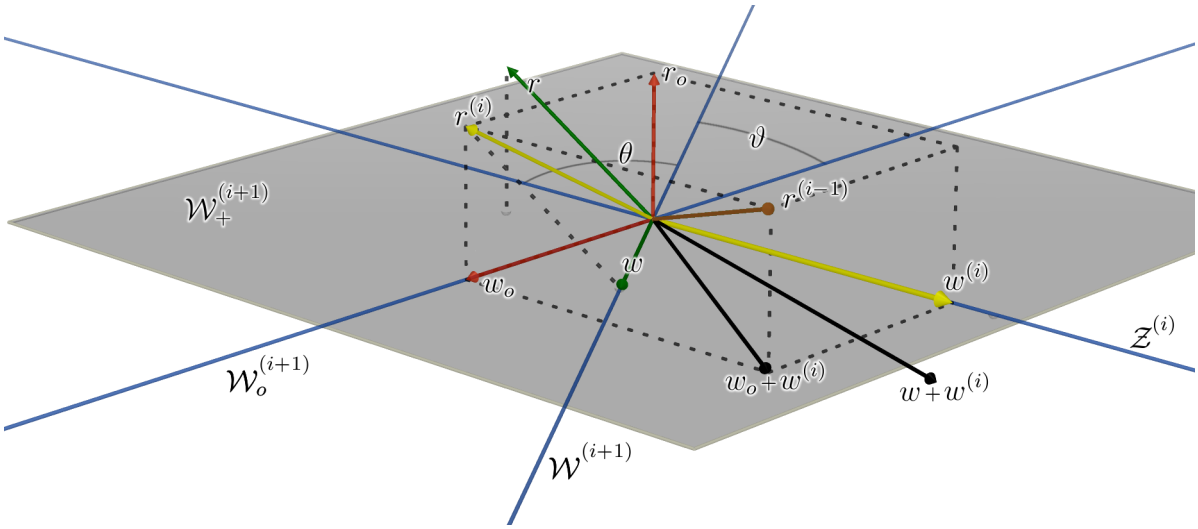

Figura 7.4: Uma representação gráfica dos resíduos e aproximações estabelecidas na definição 7.7.2. Na figura denotamos $\vartheta = \pi/2 - \theta$.

Reparemos que nesta figura, $w_o + w^{(i)}$ é exatamente a aproximação RM de $r^{(i-1)}$ no subespaço $\mathcal{W}_+^{(i+1)}$, enquanto $w_o + w$ não é. Outro fato interessante é que r é o resíduo associado a aproximação $w + w^{(i)}$ com relação a $r^{(i-1)}$, enquanto r_o é o resíduo associado a aproximação $w_o + w^{(i)}$ com relação a $r^{(i-1)}$. Notemos também que estes resultados estão diretamente relacionados ao método GCRO e GMRESR. A aproximação $w_o + w^{(i)}$ é gerada pelo primeiro e $w + w^{(i)}$ é gerada pelo segundo.

Capítulo 8

Truncamento de Subespaços

Os métodos com subespaço de aumento - ou simplesmente *métodos aumentados* - apresentados na seção anterior, apesar de todas as estratégias tomadas, não deixam de apresentar o problema de crescimento constante do subespaço, no caso do subespaço de aumento $\mathcal{G}^{(i)}$ e/ou $\mathcal{Z}^{(i)}$. Neste capítulo discutiremos mais uma estratégia para tentar solucionar este problema.

Para facilitar a leitura, supressaremos os índices de ciclo. Denotaremos

$$\begin{array}{lll} \mathcal{G} = \mathcal{G}^{(i-1)} & \mathcal{C} = \mathcal{C}^{(i)} & \\ \mathcal{Z} = \mathcal{Z}^{(i-1)} & \mathcal{W} = \mathcal{W}^{(i)} & r = r^{(i-1)} \perp \mathcal{Z} \end{array}$$

durante toda esta seção, exceto quando especificado.

8.1 Truncamento Simples

Vamos descrever agora uma técnica muito simples e que pode ser implantada para economizar recursos tanto em métodos aumentados como GMRESR e GCRO, quanto em métodos completos, como GMRES e FOM.

Consideremos um subespaço qualquer $\mathcal{U} \subset \mathcal{H}$ de dimensão ϕ, tal que

$$\mathcal{U} = \text{span}\{u_1, u_2, \ldots, u_\phi\}. \tag{8.1.1}$$

Sejam dados

$$\phi = \chi + \tau \tag{8.1.2}$$

onde χ e τ são inteiros maiores que zero, escolhidos previamente.

Para economizar recursos, escolheremos um subespaço $\bar{\mathcal{U}} \subset \mathcal{U}$ tal que $\dim(\mathcal{U}) = \tau$ para continuar guardando, e descartaremos as demais informações. Este processo

é chamado de *"truncamento"*. A forma mais simples de truncamento consiste em selecionar

$$\bar{\mathcal{U}} = \text{span}\{u_{\chi+1}, u_{\chi+2}, \ldots, u_{\phi}\}. \tag{8.1.3}$$

Nos casos mais simples, teremos $\chi = 1$, o que significa que será descartado apenas o primeiro vetor. Repare também que a técnica do recomeço é nada mais que um caso particular de truncamento onde $\chi = \phi$.

Esta técnica pode ser aplicada nos métodos completos como GMRES e FOM ao invés do recomeço. Entretanto, utilizaremos esta técnica especialmente para reduzir o tamanho dos subespaços de aumento dos métodos LGMRES, GCRO e GMRESR.

Denotaremos por (τ, χ) métodos truncados, quando necessário. Em especial, denotaremos por GMRESR(ρ, τ, χ) o método GMRESR(ρ) e onde o subespaço de aumento terá dimensão de no máximo $\tau + \chi$, sendo descartado um subespaço de dimensão χ quando a dimensão máxima for atingida. Supriremos estes valores quando não forem necessários.

8.2 Truncamento Ótimo

Nesta seção estudaremos uma forma mais "consciente" de efetuar o truncamento de um subespaço. O que faremos agora é considerar a escolha de um subespaço de forma que o resíduo obtido seja minimizado. Considerando um caso genérico, se r é o resíduo que dispomos no momento que efetuaremos o truncamento, então o subespaço ótimo a ser escolhido é

$$\bar{\mathcal{U}} = \arg \min_{\substack{\bar{\mathcal{V}} \subset \mathcal{U} \\ \dim(\bar{\mathcal{V}}) = \tau}} \left\| (I - P_{\mathcal{U}})r - (I - P_{\bar{\mathcal{V}}})r \right\|_2 \tag{8.2.1}$$

$$= \arg \min_{\substack{\bar{\mathcal{V}} \subset \mathcal{U} \\ \dim(\bar{\mathcal{V}}) = \tau}} \left\| (P_{\bar{\mathcal{V}}} - P_{\mathcal{U}})r \right\|_2 \tag{8.2.2}$$

Entretanto, em métodos aumentados como o GCRO, o resíduo pode ser gerado com relação a dois espaços, e estamos interessados em truncar apenas o subespaço de aumento, uma vez que o subespaço de aproximação será descartado no final do ciclo de qualquer forma. Além disso, na ausência de propriedades especiais nestes subespaços, torna-se difícil estabelecer uma forma geral de resolver o problema acima.

Consideremos a definição a seguir.

Definição 8.2.1 (Subespaços para Truncamento)**.** *Sejam*

$$\mathcal{Z}^{(\chi)} \subset \mathcal{Z}, \qquad \dim\left(\mathcal{Z}^{(\chi)}\right) = \chi, \tag{8.2.3}$$

$$\bar{\mathcal{Z}} \subset \mathcal{Z}, \qquad \dim\left(\bar{\mathcal{Z}}\right) = \tau, \tag{8.2.4}$$

$$\mathcal{Z} = \bar{\mathcal{Z}} \oplus \mathcal{Z}^{(\chi)}, \qquad \dim\left(\mathcal{Z}\right) = \phi, \tag{8.2.5}$$

onde $\bar{\mathcal{Z}} \cap \mathcal{Z}^{(\chi)} = \emptyset$*. Denotaremos* $\mathcal{Z}$ *como o subespaço a ser truncado,* $\mathcal{Z}^{(\chi)}$ *como sendo o subespaço escolhido para ser descartado e o subespaço* $\bar{\mathcal{Z}}$ *como sendo o subespaço a ser mantido. Também denotaremos*

$$\mathcal{C} = \mathcal{K}_\rho(A, r), \quad \dim\left(\mathcal{C}\right) = \rho \qquad \mathcal{C}_o = (I - P_{\mathcal{Z}})\mathcal{C}, \tag{8.2.6}$$

$$\mathcal{W} = A\mathcal{C}, \qquad \mathcal{W}_o = (I - P_{\mathcal{Z}})\mathcal{W}, \tag{8.2.7}$$

onde $r \perp \mathcal{Z}$ *é dado e a matriz* A *é não-singular. Suporemos também que* $\mathcal{C} \cap \mathcal{G} = \emptyset$ *o que nos leva a* $\dim\left(\mathcal{W}_o\right) = \dim\left(\mathcal{W}\right) = \rho$*. Com isso definimos*

$$\bar{\mathcal{W}}_+ = \bar{\mathcal{Z}} \oplus \mathcal{W} \qquad \dim\left(\bar{\mathcal{W}}_+\right) = \rho + \tau$$

$$\bar{\mathcal{W}}_o = (I - P_{\bar{\mathcal{Z}}})\mathcal{W} \quad \textit{ou equivalentemente} \quad \bar{\mathcal{W}}_o = \bar{\mathcal{W}}_+ \cap (\bar{\mathcal{Z}})^\perp$$

Será também necessário definir um conjunto de bases.

Definição 8.2.2 (Bases para os Subespaços)**.** *Sejam os subespaços da definição 8.2.1 e* $A \in \mathbb{C}^{n\times n}$*. Consideraremos dados* $Z \in \mathbb{C}^{n\times\phi}$*,* $\tilde{W} \in \mathbb{C}^{n\times\rho}$ *and* $W_o \in \mathbb{C}^{n\times\rho}$ *tais que*

$$Z = \left[z_1, z_2, \ldots, z_\phi\right], \quad \mathrm{img}(Z) = \mathcal{Z}, \quad Z^H Z = I_\phi$$

$$\tilde{W} = \left[\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_\rho\right], \quad \mathrm{img}(\tilde{W}) = \mathcal{W}$$

$$W_o = \left[w_1^{(o)}, w_2^{(o)}, \ldots, w_\rho^{(o)}\right], \quad \mathrm{img}(W_o) = \mathcal{W}_o, \quad (W_o)^H W_o = I_\rho.$$

Definimos

$$M = \left(Z^H \tilde{W}\right)\left(W_o^H \tilde{W}\right)^{-1}$$

E tomamos a decomposição do valor singular

$$M = X \Xi Y^H \tag{8.2.8}$$

onde $X = \left[x_1, x_2, \ldots, x_\phi\right]$ *e* $Y = \left[y_1, y_2, \ldots, y_\rho\right]$ *estão ordenados de modo a seguir a convenção*

$$\xi_1 \geq \xi_2 \geq \ldots \geq \xi_p,$$

e onde os ξ_j *são os elementos da diagonal principal de* $\Xi \in \mathbb{C}^{\phi\times\rho}$*,* $p = \min(\phi, \rho)$*.*

Sejam as matrizes $\bar{Z} \in \mathbb{C}^{n\times\tau}$, $\bar{\tilde{W}}_C, \bar{W}_C \in \mathbb{C}^{n\times(\rho+\tau)}$, *and* $Z^{(\chi)} \in \mathbb{C}^{n\times\chi}$ *dadas de modo que*

$$\begin{aligned}
\bar{Z} &= [\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_\tau], & \operatorname{img}(\bar{Z}) &= \bar{\mathcal{Z}}, & \bar{Z}^H\bar{Z} &= I_\tau \\
Z^{(\chi)} &= \left[z_1^{(\chi)}, z_2^{(\chi)}, \ldots, z_\chi^{(\chi)}\right], & \operatorname{img}(Z^{(\chi)}) &= \mathcal{Z}^{(\chi)}, & (Z^{(\chi)})^H Z^{(\chi)} &= I_\chi \\
\bar{W}_+ &= \begin{bmatrix}\bar{Z} & W_o\end{bmatrix}, & \operatorname{img}(\bar{W}_+) &= \bar{\mathcal{W}}_+, & \bar{W}_+^H\bar{W}_+ &= I_{(\rho+\tau)} \\
\check{W}_+ &= \begin{bmatrix}\bar{Z} & \tilde{W}\end{bmatrix}, & \operatorname{img}(\check{W}_+) &= \bar{\mathcal{W}}_+
\end{aligned}$$

Definimos

$$\bar{M} = \left((W^{(\chi)})^H\check{W}_+\right)\left(\bar{W}_+^H\check{W}_+\right)^{-1}$$

e tomamos a decomposição do valor singular

$$\bar{M} = \bar{X}\bar{\Xi}\bar{Y}^H \tag{8.2.9}$$

onde $\bar{X} = \left[\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_\chi\right]$ *e* $\bar{Y} = \left[\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{(\rho+\tau)}\right]$ *estão ordenados de modo a seguir a convenção*

$$\bar{\xi}_1 \geq \bar{\xi}_2 \geq \ldots \geq \bar{\xi}_{\bar{p}},$$

onde $\bar{\xi}_j$ *são os elementos da diagonal principal de* $\bar{\Xi} \in \mathbb{C}^{\chi\times(\rho+\tau)}$, $\bar{p} = \min(\chi, \rho+\tau)$.

Uma vez que definimos $\bar{\mathcal{Z}} \subset \mathcal{Z}$, é possível escrever

$$\bar{Z} = ZT^{(\tau)}$$

para alguma matriz ortonormal $T^{(\tau)} \in \mathbb{C}^{\phi\times\tau}$. O mesmo pode ser dito com respeito a

$$Z^{(\chi)} = ZT^{(\chi)}$$

onde $T^{(\chi)} \in \mathbb{C}^{\phi\times\chi}$ é ortonormal. Com isto, caso possamos encontrar $T^{(\tau)}$ tal que

$$T^{(\tau)} = \arg \min_{\substack{V\in\mathbb{C}^{\phi\times\tau} \\ V^HV=I_\tau \\ \bar{\mathcal{V}}=\operatorname{img}(ZV)}} \left\|(P_{\mathcal{Z}} - P_{\bar{\mathcal{V}}})r\right\|_2 \tag{8.2.10}$$

podemos efetuar o truncamento ótimo simplesmente multiplicando Z por $T^{(\tau)}$ à direita.

Definição 8.2.3 (Resíduos dos Subespaços para Truncamento)**.** *Notaremos resíduos com relação aos subespaços definidos em 8.2.1 da seguinte forma:*

$$w_o = P_{\mathcal{W}_o}r \qquad r_o = r - w_o \tag{8.2.11}$$

$$\bar{w}_o = P_{\bar{\mathcal{W}}_o}r \qquad \bar{r}_o = r - \bar{w}_o \tag{8.2.12}$$

O problema (8.2.10) então é nada mais que uma minimização da norma da diferença entre os resíduos r_o e $\bar{r}_o$. O teorema a seguir nos dá uma expressão para tal norma.

Teorema 8.2.4. *A norma da diferença entre os resíduos é dada por*

$$\|\bar{r}_o - r_o\|_2^2 = \sum_{j=1}^{\bar{p}} \left|\bar{\nu}_j\right|^2 \frac{\bar{\xi}_j^2}{1+\bar{\xi}_j^2}$$

onde $\bar{\nu}_j = \left\langle \bar{W}_+ \bar{y}_j, r \right\rangle$.

Demonstração. Este teorema é um pedaço do teorema 2.4 que pode ser encontrado em [40, p.871]. □

Foi observado por de Sturler em [40], se negligenciarmos os coeficiente $\bar{\nu}_j$, o problema (8.2.10) torna-se a minimização dos valores singulares $\bar{\xi}_j$. Baseado nesta observação, vamos buscar $T^{(\tau)}$ e $T^{(\chi)}$ de modo que os τ maiores valores singulares de M sejam retirados de $\bar{M}$.

Teorema 8.2.5. *Seguindo a notação de 8.2.1 e 8.2.2, se tomarmos* $\bar{Z} = ZT^{(\tau)}$ *e* $Z^{(\chi)} = ZT^{(\chi)}$ *com*

$$T^{(\tau)} = [x_1, x_2, \ldots, x_\tau] \qquad e \qquad T^{(\chi)} = [x_{(\tau+1)}, x_{(\tau+2)}, \ldots, x_\phi],$$

então os valores singulares de $\bar{M}$ *são*

$$\xi_{(p-\tau)} \geq \xi_{(p-\tau+1)} \geq \ldots \geq \xi_{(p)}.$$

Demonstração. Temos que

$$\begin{aligned}
\bar{M} &= (Z^{(\chi)})^H \begin{bmatrix} \bar{Z} & \tilde{W} \end{bmatrix} \left(\begin{bmatrix} (\bar{Z})^H \\ W_o^H \end{bmatrix} \begin{bmatrix} \bar{Z} & \tilde{W} \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} 0 & (Z^{(\chi)})^H \tilde{W} \end{bmatrix} \begin{bmatrix} I & (\bar{Z})^H \tilde{W} \\ 0 & W_o^H \tilde{W} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} 0 & (Z^{(\chi)})^H \tilde{W} \end{bmatrix} \begin{bmatrix} I & -(T^{(\tau)})^H M \\ 0 & (W_o^H \tilde{W})^{-1} \end{bmatrix} \\
&= \begin{bmatrix} 0 & (T^{(\chi)})^H M \end{bmatrix}
\end{aligned}$$

De acordo com o teorema 7.3.10 ([41])

$$\bar{\xi}_j = \min_{q_1,\ldots,q_{(j-1)} \in \mathbb{C}^m} \max_{\substack{u \neq 0, u \in \mathbb{C}^m \\ u \perp q_1,\ldots,q_{(j-1)}}} \frac{\left\|\bar{M}u\right\|_2}{\|u\|_2}$$

Observemos que, fazendo $\breve{y}_i^T = \begin{bmatrix} 0 & y_i^T \end{bmatrix}$, teremos

$$\begin{aligned}\frac{\left\|\bar{M}\breve{y}_i\right\|_2}{\left\|\breve{y}_i\right\|_2} &= \left\|\begin{bmatrix} 0 & (T^{(\chi)})^H X \Xi Y^H \end{bmatrix} \breve{y}_i\right\|_2 = \left\|(T^{(\chi)})^H X \Xi Y^H y_i\right\|_2 = \left\|(T^{(\chi)})^H X \Xi e_i\right\|_2 \\ &= \xi_i \left\|(T^{(\chi)})^H X e_i\right\|_2 = \xi_i \left\|(T^{(\chi)})^H x_i\right\|_2 .\end{aligned}$$

Como x_i tem norma um e $T^{(\chi)}$ é ortonormal, caso $x_i \in \mathrm{img}(T^{(\chi)})$, teremos

$$\xi_i \left\|(T^{(\chi)})^H x_i\right\|_2 = \xi_i.$$

Se quisermos que os valores singulares $\bar{\xi}_i$ sejam iguais a ξ_i, então basta que $x_i \in \mathrm{img}(T^{(\chi)})$. Como nós guardaremos apenas os χ menores valores singulares, devemos nos certificar de que $x_{(\tau+1)}, x_{(\tau+2)}, \ldots, x_{(\phi)} \in \mathrm{img}(T^{(\chi)})$, e uma escollha imediata seria

$$T^{(\chi)} = [x_{(\tau+1)}\ x_{(\tau+2)}\ \ldots\ x_{(\phi)}]$$

.

A escolha $T^{(\tau)} = [x_1\ x_2\ \ldots\ x_\tau]$ nos certifica de que (8.2.3) será satisfeito. □

Até agora, nada garante que esta escolha nos leve de fato ao truncamento ótimo, uma vez que negligenciamos os $\bar{\nu}_j$; quando um $\bar{\nu}_j$ é muito pequeno, os valores singulares não precisam ser levados em consideração. Entretanto, o teorema 2.8 em [25, p.272] nos mostra que esta é a solução ótima para o problema (8.2.10). Colocamos aqui uma forma adaptada deste teorema, embora a demonstração seja equivalente. O teorema 8.3.3 que veremos na seção 8.3 também se assemelha com o teorema 2.8 de [25, p.272].

Teorema 8.2.6 (Truncamento Ótimo)**.** *De todos os subespaços τ-dimensionais $\bar{\mathcal{Z}} \subset \mathcal{Z}$, o que minimiza $\|(P_{\bar{\mathcal{Z}}} - P_{\mathcal{Z}})r\|$ para $r \in \mathcal{Z} \perp$ é dado por $\bar{\mathcal{Z}} = \mathrm{span}\{\breve{z}_1, \breve{z}_2, \ldots, \breve{z}_\tau\}$ onde $\breve{z}_j$ é a j-ésima coluna de $\breve{Z} = ZX$.*

Demonstração. O teorema 2.8 de [25, p.272] considera que $\breve{Z}$ e $\breve{W}$ são bases biortonormais para o subespaço $\mathcal{Z}$ e $\mathcal{W}$ respectivamente, tais que $(\breve{Z})^H\breve{W}$ possui os elementos da diagonal principal em ordem não-crescente, assim como na definição 7.7.1. Basta então mostrar que, dada a base $\breve{W}$, ZX satisfaz esta condição.

Consideremos a decomposição QR $\tilde{W} = WS$. Então os valores singulares de Z^HW são o cosseno dos ângulos canônicos[1] entre $\mathcal{Z}$ e $\mathcal{W}$, que de acordo com a definição 7.7.1, são os elementos da diagonal principal de Σ. De (8.2.8) nós já temos os vetores singulares a direita da decomposição de valores singulares de Z^HW. Então

$$(Z)^HW = X\Sigma\breve{Y}^H$$

[1] trata-se de um resultado conhecido, que pode ser encontrado, por exemplo, em [1, p.456] e [42]

é uma decomposição do valor singular para alguma matriz $\breve{Y}$, onde X e $\breve{Y}$ estão ordenados de forma a obedecer à convenção de que os elementods da diagonal principal de Σ estão em ordem não-crescente. Notemos que

$$(ZX)^H W\breve{Y} = \Sigma$$

e consequentemente, ZX e $W\breve{Y}$ são bases biortonormais ordenadas de modo que Σ tenha os elementos da diagonal principal dispostos em ordem não-crescente, completando a prova. □

8.3 Considerações a Respeito dos Truncamentos

Vamos agora considerar algumas interpretações geométricas a respeito do truncamento ótimo proposto na seção 8.2. A proposição a seguir pode ser encontrada em [25, p.289]:

Proposição 8.3.1. *Seguindo a notação da definição, 8.2.2, os valores singulares ξ_j são a contangente dos θ_j, que são os ângulos canônicos entre os subespaços $\mathcal{Z}$ e $\mathcal{W}$.*

Isto nos dá uma interpretação geométrica do truncamento apresentado. Se descartamos os maiores valores singulares de M estamos na verdade descartando os maiores cossenos, o que significa que estamos na verdade truncando o subespaço $\mathcal{Z}$ de modo a manter o subespaço "mais ortogonal" a $\mathcal{W}$.

Proposição 8.3.2. *Seguindo a notação da definição 8.2.2, teremos*

$$\xi_j = \frac{\cos\theta_j}{\cos\vartheta_j}, \tag{8.3.1}$$

onde θ_j são os ângulos canônicos entre os subespaços $\mathcal{Z}$ e $\mathcal{W}$, e ϑ_j são os ângulos canônicos entre os subespaços $\mathcal{W}_o$ e $\mathcal{W}$.

Demonstração. Primeiramente, notemos que para qualquer ângulo α teremos $\text{sen}(\alpha) = \cos(\frac{\pi}{2} - \alpha)$. De acordo com a própria definição de $\mathcal{W}_o$, concluímos que $\vartheta_j = (\frac{\pi}{2} - \theta_j)$, de onde temos que

$$\xi_j = \frac{\cos\theta_j}{\text{sen}\theta_j} = \frac{\cos\theta_j}{\cos(\frac{\pi}{2} - \theta_j)} = \frac{\cos\theta_j}{\cos\vartheta_j}, \tag{8.3.2}$$

provando a proposição. □

O teorema a seguir nos mostra a relação entre os σ_j e a norma da diferença entre as aproximações, ou a norma da diferença dos resíduos. Usaremos neste teorema

as mesmas bases definidas no capítulo anterior, na definição 7.7.1, e alguns dos resultados mostrados no teorema 8.2.6.

Teorema 8.3.3. *Seguindo a notação estabelecida nas definições 7.7.1 e 8.2.2, se tomarmos a matriz $\bar{Z} = ZT^{(\tau)}$ de acordo com o teorema 8.2.5, nós teremos*

$$w_o - \bar{w}_o = \sum_{j=\tau+1}^{p} \left\langle r, \breve{w}_j^o \right\rangle \sigma_j \left(\gamma_j \breve{z}_j - \sigma_j \breve{w}_j^o \right) \tag{8.3.3}$$

$$\|w_o - \bar{w}_o\|_2^2 = \sum_{j=\tau+1}^{p} \sigma_j^2 \left| \left\langle r, \breve{w}_j^o \right\rangle \right|^2 . \tag{8.3.4}$$

Demonstração. Do teorema 2.6 em [40, p.872], se a matriz $T^{(\tau)}$ for tomada de acordo com 8.2.5, teremos

$$\bar{r}_o - r_o = \sum_{j=\tau+1}^{p} \left(\frac{\nu_j \xi_j^2}{1+\xi^2} W_o y_j - \frac{\nu_j \xi_j}{1+\xi^2} Z x_j \right) \tag{8.3.5}$$

$$\|\bar{r}_o - r_o\|_2^2 = \sum_{j=\tau+1}^{p} \frac{\left|\nu_j\right|^2 \xi_j^2}{1+\xi_j^2}, \tag{8.3.6}$$

onde $\nu_j = \left\langle W_o y_j, r \right\rangle$.

Nos baseando na prova do teorema 8.2.6, definiremos $\breve{Z} = ZX$ e $\breve{W} = W\breve{Y}$ para alguma matriz unitária $\breve{Y}$. A definição 7.7.1 nos dá

$$\begin{aligned} \Gamma^2 = I - \Sigma^H \Sigma &= \breve{W}^H \breve{W} - \breve{W}^H \breve{Z} \breve{Z}^H \breve{W} \\ &= \breve{W}^H (I - \breve{Z}\breve{Z}^H) \breve{W} = \breve{W}^H \breve{W}_o \Gamma \\ \breve{W}_o^H \breve{W} &= \Gamma. \end{aligned}$$

Usando as proposições 8.3.1 e 8.3.2, concluímos que $\Xi = \Sigma\Gamma^{-1}$, e da decomposição do valor singular de M temos que

$$\begin{aligned} Z^H \tilde{W} (W_o^H \tilde{W})^{-1} &= X \Xi Y^H \\ X^H Z^H W S (W_o^H W S)^{-1} &= \Xi Y^H \\ (ZX)^H W \breve{Y} \breve{Y}^H S S^{-1} (W_o^H W)^{-1} &= \Xi Y^H \\ \breve{Z}^H \breve{W} (W_o^H W \breve{Y})^{-1} &= \Xi Y^H \\ \Sigma (W_o^H \breve{W})^{-1} &= \Sigma \Gamma^{-1} Y^H \\ \Sigma (Y^H W_o^H \breve{W})^{-1} &= \Sigma (\breve{W}_o^H \breve{W})^{-1} \end{aligned}$$

Ao definirmos $\breve{W}_o = W_o Y$, a igualdade acima é satisfeita. Substituindo as bases

teremos

$$\bar{r}_o - r_o = \sum_{j=\tau+1}^{p} \left\langle r, \breve{w}_j^o \right\rangle \left(\frac{\xi_j^2}{1+\xi^2} \breve{w}_j^o - \frac{\xi_j}{1+\xi^2} \breve{z}_j \right)$$
$$\|\bar{r}_o - r_o\|^2 = \sum_{j=\tau+1}^{p} \left| \left\langle r, \breve{w}_j^o \right\rangle \right|^2 \frac{\xi_j^2}{1+\xi_j^2}.$$

Notando que $1 + \cot\alpha = \csc\alpha$ para qualquer ângulo α, teremos

$$\frac{\xi_j^2}{1+\xi_j^2} = \frac{\cos^2\theta_j}{\text{sen}^2\theta_j} \frac{1}{\csc^2\theta_j} = \cos^2\theta_j = \sigma_j^2$$
$$\frac{\xi_j}{1+\xi_j^2} = \frac{\sigma_j^2}{\xi_j} = \sigma_j^2 \frac{\text{sen}\theta_j}{\cos\theta_j} = \sigma_j\gamma_j.$$

Lembrando novamente a relação entre os resíduos e as aproximações

$$\bar{r}_o - r_o = w_o - \bar{w}_o$$

finalizamos a prova. □

Repare a semelhança deste teorema e do teorema 7.7.3 com teorema 2.8 de [25].

Este resultado nos mostra que $\|\bar{r}_o - r_o\|$ depende unicamente dos $p - \tau$ maiores cossenos dos ângulos canônicos θ_j. Esta é uma idéia natural; no teorema 7.7.3 vimos qual era a diferença das aproximações quando negligenciávamos completamente a ortogonalidade entre dois subespaços. Neste caso estamos escolhendo apenas uma parte de um subespaço para negligenciar a ortogonalidade (que é o subespaço que vamos descartar, $\mathcal{Z}^{(\chi)}$). Continuaremos mantendo ortogonalidade com relação ao subespaço de dimensão τ, e portanto, os cossenos referentes à tal subespaço serão zero. Por isso, o resultado deste teorema relaciona a norma da diferença apenas com os cossenos dos ângulos que iremos negligenciar.

Retomemos agora os índices relativos aos ciclos. Outra observação interessante a respeito do truncamento ótimo é que estes ângulos são calculados com respeito ao subespaço $\mathcal{W}^{(i)}$; entretanto, nada garante que estes ângulos sejam os ângulos ótimos com relação ao subespaço $\mathcal{W}^{(j)}$, $j > i$. Trata-se portanto de uma heurística: supomos que estes valores singulares serão os menores pois nos ciclos anteriores, eles foram os menores.

8.4 Métodos com Subespaço Truncado

Nesta seção discutiremos como foi proposta a forma de truncamento a cada método apresentado no capítulo anterior.

Primeiramente, vejamos o método GMRESR. Conforme foi proposto por Van der Vorst e Vuik em [32], o método GMRESR necessita de três parâmetros além do já esperado ρ. O primeiro parâmetro, k, indica se o GMRESR utilizará a estratégia do recomeço para o subespaço de aumento, e qual será o tamanho máximo do subespaço de aumento neste caso. O segundo parâmetro, semelhantemente, indica qual será o tamanho máximo do subespaço de aumento caso este apresente um truncamento simples. Este parâmetro portanto é o que chamamos de ϕ nas sessões anteriores, sendo $\chi = 1$ por padrão. Porfim, foi sugerido pelos autores que não fosse executado apenas um ciclo do GMRES(ρ), mas sim que este fosse executado até que a norma do resíduo atingisse uma certa tolerância. Esta tolerância é dada pelo terceiro parâmetro, $n(\epsilon)$.

Para o método LGMRES, foi recomendada por Baker, Jessup *et al.* em [36] o truncamento simples com $\chi = 1$. Isto porque, ao mantermos os τ últimos $c^{(j)}$ em $\mathcal{G}^{(i)}$ no ciclo i, estaremos prevenindo o comportamento de alternância entre os τ resíduos anteriores, isto é, estaremos aumentando os ângulos alternados $\vartheta^{(i,\tau)}$. Segundo Baker, Jessup, *et al.*, resultados numéricos mostram que um valor satisfatório de τ em geral é pequeno, na maioria dos casos menor ou igual a 3.

A princípio, para o método GCRO foi proposto um truncamento utilizando a mesma idéia da seção 8.2, exceto que a matriz $T^{(\tau)}$ não havia sido definida ainda. Em [35] é recomendada a leitura de [43] e [44] para mais detalhes a respeito desta escolha.

Embora o truncamento ótimo tenha sido proposto por de Sturler em [40], ele não é de fato utilizado por nenhum método deste autor. Isto porque conforme vimos no capítulo 6, o método GCRO não constrói uma base para o subespaço $\mathcal{W}$, construindo uma base para $\mathcal{W}_o$ ao invés disto. O truncamento ótimo proposto na seção 8.2 necessita de uma base para $\mathcal{W}$ para calcular a decomposição do valor singular de M. Repare que, como projeções em geral não são operações invertíveis, nada garante que será possível extrair informações do subespaço $\mathcal{W}$ de $\mathcal{W}_o = (I - P_{\mathcal{Z}})\mathcal{W}$.

O que de Sturler propôs foi uma alternativa que chamaremos de *"truncamento quase-ótimo"*. Ao invés da decomposição do valor singular da matriz M, no truncamento quase ótimo faremos a decomposição do valor singular de

$$\tilde{M} = Z^H \tilde{W}_o (W_o^H \tilde{W}_o)^{-1} \tag{8.4.1}$$

onde $\tilde{W}_o = AC_o$ e C_o é uma base ortonormal para $\mathcal{C}_o$. O método que usa esta estratégia chama-se GCROT [40], e foi pubilcado em 1999.

Baseados nesta observação, criamos um método chamado OT[45] (Optimal Truncation, ou Truncamento Ótimo). Trata-se do método GCRO com a única diferença que, durante o truncamento, utiliza-se o método de Arnoldi para gerar uma base

para o subespaço $\mathcal{W}$ para então efetuar o truncamento ótimo.

Daqui em diante notaremos os métodos pelo nome do próprio seguido de (ρ, τ, χ) quando se fizer necessário especificar os índices.

Capítulo 9

Implantação e Resultados

Nesta seção discutiremos como implantar e alguns resultados obtidos com a execução dos métodos apresentados no capítulo 8. Nos atentaremos à comparação entre os métodos LGMRES, GCROT e OT.

9.1 GCRO - Versão Econômica

Baseado no trabalho de Fokkema em [46], de Sturler propôs uma versão mais econômica para a implantação do método GCRO, que explicaremos a seguir.

Definição 9.1.1. *Consideremos as bases e subespaços já definidos na seção 7.4 do capítulo 6. Então, na iteração $i+1$ definiremos*

$$\begin{aligned} Z^{(i)} &= \bar{Z}^{(i)} N^{(i)} \\ N^{(i)} &= diag(\|z_1\|_2^{-1}, \|z_2\|_2^{-1}, \ldots, \left\|z^{(i)}\right\|_2^{-1}) \\ A\tilde{G}^{(i)} &= \bar{Z}^{(i)} S^{(i)} \end{aligned}$$

onde $S^{(i)}$ é triangular superior. Finalmente, o vetor d é definido de modo a satisfazer a relação

$$x^{(i)} = \tilde{G}^{(i)} (S^{(i)})^{-1} d.$$

Tomamos uma base

$$\mathrm{img}(V_\rho) = \mathcal{C}_o^{(i+1)}, \qquad V_\rho^H V_\rho = I_\rho$$

que é a base gerada por ρ passos do GMRES. Lembremos a decomposição de Arnoldi

$$(I - Z^{(i)}(Z^{(i)})^H) A V_\rho = V_{(\rho+1)} \tilde{H}_\rho. \tag{9.1.1}$$

Definimos então

$$w_o = P_{\mathcal{W}_o^{(i+1)}} r^{(i)} \qquad c_o = A^{-1} w_o \qquad r_o = r^{(i)} - w_o \tag{9.1.2}$$

Com estas bases definidas, reescreveremos (9.1.1) da seguinte forma:

$$\begin{aligned} AV_\rho - Z^{(i)}(Z^{(i)})^H AV_\rho &= V_{(\rho+1)}\tilde{H}_\rho \\ AV_\rho - \bar{Z}^{(i)}N^{(i)}(Z^{(i)})^H AV_\rho &= V_{(\rho+1)}\tilde{H}_\rho \\ AV_\rho &= \bar{Z}^{(i)}B + V_{(\rho+1)}\tilde{H}_\rho \end{aligned}$$

onde $B = N^{(i)}(Z^{(i)})^H AV_\rho$. Lembremos que o GMRES calcula sua aproximação e correção resolvendo o problema

$$\min_{y\in\mathbb{C}_o^{(i)}} \left\| V_{\rho+1}(\beta e_1^{\rho+1} - \tilde{H}_\rho y) \right\|_2,$$

obtendo $w_o = V_{(\rho+1)}\tilde{H}_\rho y$ e $c_o = V_\rho y$. Considerando que já dispomos de y, setaremos

$$\begin{aligned} \tilde{g}^{(i+1)} &= c_o = V_\rho y \\ \bar{z}^{(i+1)} &= w_o = V_{(\rho+1)}\tilde{H}_\rho y, \end{aligned}$$

e com isto,

$$A\tilde{g}^{(i+1)} = AV_\rho y = \bar{Z}^{(i)}B_\rho y + V_{(\rho+1)}\tilde{H}_\rho y = \bar{Z}^{(i)}B_\rho y + \bar{z}^{(i+1)}.$$

Para obter $A\tilde{G}^{(i+1)} = \bar{Z}^{(i+1)}S^{(i+1)}$, definiremos

$$S^{(i+1)} = \begin{bmatrix} S^{(i)} & B_\rho y \\ 0 & 1 \end{bmatrix}$$

É mostrado pelo teorema 2.2 em [35, p.20] que $r^{(i+1)} = r_o$.

Também não será necessário o cálculo da norma de $\bar{z}^{(i+1)}$:

$$\left\|\bar{z}^{(i+1)}\right\|_2^2 = \left\|r^{(i)} - r_o\right\|_2^2 = \left\|r^{(i)}\right\|_2^2 - 2\left\langle r^{(i)}, r_o\right\rangle + \|r_o\|_2^2.$$

Como os resíduos são ortogonais entre si, temos que $\left\|\bar{z}^{(i+1)}\right\|_2^2 = \left\|r^{(i)}\right\|_2^2 + \|r_o\|_2^2$. Lembramos que ambos provavelmente já foram calculados para efeitos de teste de convergência.

O algoritmo 21 mostra o esquema para o método externo do GCRO, e o algoritmo 22 mostra a adaptação do método interno GMRES(ρ) para a busca da nova direção de aumento.

No apêndice E mostramos nosso protótipo do GCRO escrito em MATLAB. Note que não utilizamos o recomeço neste protótipo, somente o método completo.

Algoritmo 21: GCRO - Método Externo

$r^{(0)} = b - Ax^{(0)}, i = 0;$
while $\left\|r^{(i)}\right\|_2 > tol$ **do**
  invoca GMRES_Interno;
  $\bar{z}^{(i+1)} = r^{(i)} - r_o;$
  $N_{(i+1,i+1)} = \left(\left\|r^{(i)}\right\|_2^2 + \|r_o\|_2^2\right)^{-1/2};$
  $S_{(i+1,i+1)} = 1;$
  $d_{(i+1)} = 1;$
  $i = i + 1;$
end while
$x^{(i+1)} = x^{(0)} + \tilde{G}^{(i)}(S^{(i)})^{-1}d$

Algoritmo 22: GCRO - Método Interno (GMRES(ρ) Modificado)

$v_1 = r^{(i)}/\left\|r^{(i)}\right\|_2;$
for $j = 1, \ldots, \rho$ **do**
  $v_{(j+1)} = Av_j;$
  for $i = 1, \ldots, i$ **do**
    $b_{(i,j)} = \left\langle N_{(i,i)}^2 \bar{z}^{(i)}, v_{(j+1)} \right\rangle;$
    $v_{(j+1)} = v_{(j+1)} - b_{(i,j)}\bar{z}^{(i)};$
  end for
  for $i = 1, \ldots, j$ **do**
    $h_{(i,j)} = \left\langle v_i, v_{(j+1)} \right\rangle;$
    $v_{(j+1)} = v_{(j+1)} - h_{(i,j)}v_i;$
  end for
  $h_{(j+1,j)} = \left\|v_{(j+1)}\right\|_2;$
  $v_{(j+1)} = v_{(j+1)}/h_{(j+1,j)};$
end for
$y = \arg\min_{y \in \mathbb{C}^{\rho+1}} \left\| \left\|r^{(i)}\right\|_2 e_1 - \tilde{H}_\rho y \right\|_2;$
$\tilde{g}^{(i+1)} = V_\rho y;$
$r_o = r^{(i)} - V_{(\rho+1)}\tilde{H}_\rho y;$
$S_{(:,i+1)} = B_\rho y$

9.2 GCROT e OT - Detalhes do Algoritmo

Explicaremos a seguir a implantação do GCROT. Embora na seção 9.1 tenhamos discutido uma implantação mais econômica do GCRO, não implemantamos de fato esta versão a princípio, optando por uma versão mais intuitiva, baseado na discussão da seção 7.4. O algoritmo 23 mostra esta versão.

Algoritmo 23: GCRO - Versão Intuititva

$r^{(0)} = b - Ax^{(0)}, i = 0;$
while $\left\|r^{(i)}\right\|_2 > tol$ **do**
  $i = i + 1;$
  $v_1 = r^{(i-1)} / \left\|r^{(i-1)}\right\|_2;$
  for $j = 1$ *até* ρ **do**
    $v_{(j+1)} = Av_j;$
    for $l = 1$ *até* $i - 1$ **do**
      $B_{(l,j)} = \left\langle z_l, v_{(j+1)} \right\rangle;$
      $v_{(j+1)} = v_{(j+1)} - B_{(l,j)} z_l;$
    end for
    for $l = 1, j$ **do**
      $H_{(l,j)} = \left\langle v_l, v_{(j+1)} \right\rangle;$
      $v_{(j+1)} = v_{(j+1)} - H_{(l,j)} v_l;$
    end for
    $H_{(j+1,j)} = \left\|v_{(j+1)}\right\|_2;$
    $v_{(j+1)} = v_{(j+1)} / H_{(j+1,j)};$
  end for
  Calcula decomposição QR: $H = QR;$
  $y = \left\|r^{(i-1)}\right\|_2 R^{-1} Q^H e_1;$
  $z^{(i)} = V_{\rho+1} H y;$
  $\tilde{g}^{(i)} = (V_\rho - \tilde{G}^{(i-1)} B) y;$
  $r^{(i)} = r^{(i-1)} - z^{(i)};$
  $x^{(i)} = x^{(i-1)} + \tilde{g}^{(i)};$
  Truncar caso necessário;
  $z^{(i)} = z^{(i)} / \left\|z^{(i)}\right\|_2;$
  $\tilde{g}^{(i)} = \tilde{g}^{(i)} / \left\|z^{(i)}\right\|_2;$
end while

Repare que da forma como foi definido no algoritmo 23, teremos

$$B = (Z^{(i-1)})^H A V_\rho. \tag{9.2.1}$$

Lembrando que no GMRES teremos $c_o = V_\rho y$ então

$$\tilde{g}^{(i)} = (V_\rho - \tilde{G}^{(i-1)} B) y = c_o - \tilde{G}^{(i-1)} (Z^{(i)})^H A c_o = (I - \tilde{G}^{(i-1)} (Z^{(i)})^H A) c_o \tag{9.2.2}$$

conforme mostrado na seção 7.4. O apêndice A mostra o protótipo escrito em Matlab desse algoritmo.

O mais importante desta forma de guardar a matriz B e de encontrar y através da decomposição QR de $\tilde{H}_\rho$ é que tais matrizes serão úteis durante o truncamento do

GCROT. Seja a matriz $\tilde{M}$ conforme definido na equação 8.4.1 da seção 8.4. Teremos

$$\tilde{W}_o = AV_\rho, \quad \text{e} \quad (Z^{(i-1)})^H \tilde{W}_o = B \tag{9.2.3}$$

e da decomposição de Arnoldi do GMRES interno teremos

$$(I - Z^{(i-1)}(Z^{(i-1)})^H)AV_\rho = V_{\rho+1}\tilde{H}_\rho \tag{9.2.4}$$

$$(V_{\rho+1})^H(I - Z^{(i-1)}(Z^{(i-1)})^H)AV_\rho = QR \tag{9.2.5}$$

$$(V_{\rho+1}Q)^H AV_\rho = R \tag{9.2.6}$$

$$W_o^H \tilde{W}_o = R. \tag{9.2.7}$$

Assim, teremos

$$\tilde{M} = BR^{-1}. \tag{9.2.8}$$

Lembramos que a matriz R já foi invertida durante o cálculo de y.

O algoritmo 24 mostra o procedimento do truncamento do GCROT. O apêndice B mostra o protótipo escrito em Matlab para esse algoritmo.

Algoritmo 24: GCROT - Truncamento

```
1 if i >= τ + χ − 1 then
2 |   Calcular a decomposição do valor singular XΞY de BR^{-1};
3 |   G̃^{(τ−1)} = G̃^{(i−1)} X_{(1:τ−1)};
4 |   Z^{(τ−1)} = Z^{(i−1)} X_{(1:τ−1)};
5 |   i = τ;
6 end if
```

Para o truncamento do método OT, não calcularemos toda a matriz M, conforme mostrado na seção 8.2. Isto porque precisamos apenas da matriz X, e conforme já foi mencionado na demonstração do teoream 8.2.6, a decomposição do valor singular de $Z^H W$ será $X\Sigma\breve{Y}$ para alguma matriz unitária $\breve{Y}$.

O algoritmo 25 mostra o procedimento do truncamento do OT. O apêndice C mostra o protótipo escrito em Matlab para esse algoritmo.

9.3 Protótipos em MATLAB

Nós testamos problemas retirados da coleção de matrizes do Matrix Market[47] para efetuar comparações entre os métodos LGMRES, GCRO(ρ, τ, χ), GCRO, GCROT e OT. Foram implementados protótipos dos referidos métodos na linguagem MATLAB. O código-fonte pode ser encontrado nos apêndices A(GCRO), B(GCROT), C (OT) e D (LGMRES).

Algoritmo 25: OT - Truncamento

if $i >= \tau + \chi - 1$ **then**
  Calcular uma base ortonormal W para $\mathcal{W}^{(i)}$ utilizando ρ iterações de Arnoldi;
  Calcular a decomposição do valor singular $X\Sigma\breve{Y}$ de $(Z^{(i-1)})^H W$;
  $\tilde{G}^{(\tau-1)} = \tilde{G}^{(i-1)} X_{(1:\tau-1)}$;
  $Z^{(\tau-1)} = Z^{(i-1)} X_{(1:\tau-1)}$;
  $i = \tau$;
end if

Pelo fato de os resultados serem muito extensos, nas tabelas que se seguem escolhemos apenas parte dos resultados. Para facilitar a leitura, notaremos também GCRO(ρ, τ, χ) por GCRO(τ) apenas.

Segue a notação usada para nomear as colunas das tabelas. As colunas G-Par mostram os parâmetros utilizados em métodos derivados do GCRO - isto é, GCRO(τ), GCROT e OT. Os valores (τ, ρ) significam que o método foi executado com ρ iterações no método interno (na maior parte dos casos, um ciclo do método GMRES(ρ))e que o truncamento do subespaço de aumento é efetuado sempre que a dimensão do mesmo atinge um valor igual o maior que 2τ, pois definimos $\chi = \tau$. Naturalmente, o GCRO utiliza apenas o parâmetro ρ, já que não é efetuado nenhum tipo de truncamento no mesmo.

A coluna L-Par mostra os parâmetros utilizados no método LGMRES. Os valores (ρ, ϕ) indicam que são executados ρ iterações internas e que o subespaço de aumento possui dimensão ϕ. Aqui adotamos o padrão $\tau = 1$, e portanto um vetor é descartado a cada ciclo.

As colunas GCRO, GCRO(τ), GCROT, OT e LGMRES contém o número de iterações que o respectivo método necessitou até atingir convergência. O valor ∞ neste campo indica que o método não convergiu para os parâmetros dados. Sempre que o método LGMRES apresentar ∞, teremos também indicado $(\rho, -)$ na coluna L-Par, significando que o método não convergiu para $\phi = 3, 4, ..., \rho$.

O critério de convergência utilizado em todos os testes foi

$$\frac{\left\|r^{(i)}\right\|_2}{\left\|r^{(0)}\right\|_2} \leq 10^{-10}. \tag{9.3.1}$$

Além disso, consideramos que o algoritmo não convergiu sempre que o número de iterações do método externo tornava-se maior que a dimensão da matriz A.

Nas tabelas auxiliares colocamos dados relativos à estrutura da matriz A. As colunas "Dim." e "Den." apresentam, respectivamente, a dimensão do problema, a densidade da matriz A. Conforme foi discutido na seção 5.7, a "distância à nor-

malidade" pode dar alguma informação a respeito da convergência de métodos de Krylov caso a matriz A não seja normal, e o número de condicionamento da matriz trará informação a respeito da convergência caso a matriz A seja normal.

Colunas que apresentam o nome de dois métodos contém a razão entre o número de iterações do primeiro método pelo número de iterações do segundo método. A coluna indicada pelo nome de um método seguida de "Dim." contém a razão entre o número de iterações executadas deste método pela dimensão do problema.

Por hora, comparamos apenas o número de iterações externas, e não o esforço computacional para executar cada método. O objetivo desta comparação é estabelecer simplesmente qual estratégia, tanto de aumento quanto de truncamento, é capaz de acelerar mais a convergência de um problema em específico.

Na tabela 9.1 mostramos os problemas em que a melhora apresentada pelo GCROT em relação ao GCRO(ρ) foram mais significativas. Notemos que o método LGMRES apresentou uma convergência ruim nestes casos. A tabela 9.2 mostra informações a respeito da estrutura do problema em si. Podemos também verificar nesta tabela que em média o método GCROT convergiu 37% mais rápido que o método GCRO(ρ). Lembramos que este resultado se refere somente ao número de iterações e não ao tempo de CPU, e que nestas tabelas mostramos apenas parte dos resultados.

Tabela 9.1: Ganho mais significativo do GCROT com relação ao GCRO(ρ)

Nome	**G-Par**	**L-Par**	**GCRO**	**GCRO(ρ)**	**GCROT**	**OT**	**LGMRES**
tub1000	(11,18)	(18,19)	110	990	240	362	∞
psmigr_1	(11,18)	(18,14)	230	881	292	290	1690
bwm2000	(11,18)	(18,19)	234	1675	705	802	∞
ck400	(6,11)	(11,12)	63	329	144	138	∞
bcsstk16	(4,7)	(7,3)	156	817	364	372	401
steam2	(6,11)	(11,12)	55	284	128	155	∞
ck656	(6,11)	(11,12)	103	640	292	310	∞
cdde3	(4,7)	(7,8)	57	311	145	186	∞
cavity10	(4,7)	(7,3)	194	1197	622	680	842
cavity16	(4,7)	(7,3)	263	1700	912	905	741
fs_541_3	(20,34)	(34,35)	237	442	240	262	∞
fs_541_4	(11,18)	(18,19)	152	510	280	251	∞
pts5ldd16	(4,7)	(7,8)	286	1577	922	945	∞
e40r0000	(4,7)	(7,4)	523	4987	3024	3325	4509
e05r0200	(20,34)	(34,35)	81	167	103	102	∞

A tabela 9.3 mostra alguns resultados onde o método OT convergiu para um número menor de iterações que o método GCROT. Lembremos que mesmo assim, em alguns destes casos o método GCRO(ρ) convergiu para um número menor de iterações que o método OT. Isto é especialmente verdade para métodos derivados do GCRO, onde o subespaço de aumento é expandido através da solução de um sistema

Tabela 9.2: Informações adicionais à 9.1

Nome	**Dim.**	**Den.**	**Cond.**	**Norm.**	$\frac{\text{GCROT}}{\text{GCRO}(\rho)}$	$\frac{\text{GCROT}}{\text{Dim.}}$
utm1700a	1000	0.40%	2.31E+06	8.20E+09	0.242	0.240
rdb800l	3140	5.51%	6.82E+09	2.66E+11	0.331	0.093
rdb2048l	2000	0.20%	2.87E+05	5.75E+05	0.421	0.353
rdb200	400	1.79%	2.17E+06	8.04E+00	0.438	0.360
bcsstk17	4884	1.22%	7.01E+09	0.00E+00	0.446	0.075
rdb1250l	600	3.82%	3.55E+06	7.81E+14	0.451	0.213
olm1000	656	0.90%	1.18E+07	8.04E+00	0.456	0.445
pts5ldd24	961	0.51%	1.01E+04	3.75E-001	0.466	0.151
can___161	2597	1.13%	4.46E+06	2.73E+01	0.520	0.240
rdb3200l	4562	0.66%	1.39E+07	2.73E+01	0.536	0.200
rdb450	541	1.46%	7.74E+12	1.61E+14	0.543	0.444
dw2048	541	1.46%	2.92E+11	2.44E+12	0.549	0.518
bcsstm06	12033	0.04%	1.31E+05	7.07E+07	0.585	0.077
pts5ldd23	17281	0.19%	2.19E+08	0.00E+00	0.606	0.175
cavity02	236	10.51%	3.88E+05	3.77E+02	0.617	0.436
				Média	0.480	0.268

singular com a matriz $A_{\mathcal{Z}^{(i-1)}}$. Embora o método OT trunque o subespaço de aumento de maneira ótima, esta otimalidade está baseada apenas em uma informação local, e não global, uma vez que descartamos vários subespaços.

A coluna $\frac{OT}{GCRO(\rho)}$ na tabela 9.4 mostra um resultado interessante. O método OT geralmente converge para um número menor de iterações que o GCROT quando a performance do GCRO(ρ) é também melhor, isto é $\frac{OT}{GCROT}$ é geralmente pequeno quando $\frac{OT}{GCRO(\rho)}$ cresce.

Tabela 9.3: Ganho mais significativo do OT com relação ao GCROT

Nome	**G-Par**	**L-Par**	**GCRO**	**GCRO(ρ)**	**GCROT**	**OT**	**LGMRES**
utm1700a	(11,18)	(18,-)	106	532	973	513	∞
rdb800l	(4,7)	(7,3)	47	198	308	184	72
rdb2048l	(4,7)	(7,3)	68	336	420	258	173
rdb200	(4,7)	(7,3)	26	49	53	35	52
bcsstk17	(18,30)	(30,7)	1804	9769	10342	7096	7333
rdb1250l	(4,7)	(7,3)	54	245	272	192	87
olm1000	(20,34)	(34,-)	94	371	548	409	∞
pts5ldd24	(16,26)	(26,-)	88	294	205	157	∞
can___161	(18,30)	(30,3)	70	92	115	89	47
rdb3200l	(4,7)	(7,3)	82	296	432	335	295
rdb450	(4,7)	(7,3)	36	88	79	63	157
dw2048	(6,11)	(11,3)	576	1395	1542	1257	1251
bcsstm06	(4,7)	(7,3)	24	57	65	53	161
pts5ldd23	(9,15)	(15,12)	45	77	78	64	134
cavity02	(16,26)	(26,-)	104	235	303	258	∞

Tabela 9.4: Informações adicionais à tabela 9.3

Nome	**Dim.**	**Den.**	**Cond.**	**Norm.**	$\frac{\mathbf{OT}}{\mathbf{GCRO}(\rho)}$	$\frac{\mathbf{OT}}{\mathbf{GCROT}}$	$\frac{\mathbf{OT}}{\mathbf{Dim.}}$
utm1700a	1700	0.74%	6.24E+06	3.61E+00	0.527	0.964	0.302
rdb800l	800	0.73%	9.65E+02	2.11E+02	0.597	0.929	0.230
rdb2048l	2048	0.29%	2.07E+03	4.06E+02	0.614	0.768	0.126
rdb200	200	2.80%	7.46E+02	2.15E+02	0.660	0.714	0.175
bcsstk17	10974	0.36%	1.95E+10	0.00E+00	0.686	0.726	0.647
rdb1250l	1250	0.47%	1.38E+03	2.84E+02	0.706	0.784	0.154
olm1000	1000	0.40%	3.04E+06	8.42E+09	0.746	1.102	0.409
pts5ldd24	705	0.68%	1.80E+05	3.88E+05	0.766	0.534	0.223
can___161	161	5.31%	8.45E+02	0.00E+00	0.774	0.967	0.553
rdb3200l	3200	0.18%	2.76E+03	5.85E+02	0.775	1.132	0.105
rdb450	450	1.27%	1.64E+03	3.87E+02	0.797	0.716	0.140
dw2048	2048	0.24%	5.30E+03	1.04E-001	0.815	0.901	0.614
bcsstm06	420	0.24%	3.46E+06	0.00E+00	0.815	0.930	0.126
pts5ldd23	161	2.87%	7.34E+03	2.39E+04	0.821	0.831	0.398
cavity02	317	7.29%	2.90E+05	3.96E+02	0.851	1.098	0.814
				Média	0.730	0.873	0.334

Para analisar melhor este caso, mostramos na tabela 9.5 os casos onde o GCROT convergiu mais rápido com relação ao OT. Na tabela 9.6 vemos que $\frac{OT}{GCRO}$ são menores que na tabela. 9.4.

Tabela 9.5: Ganho mais significativo do GCROT com relação ao OT

Nome	**G-Par**	**L-Par**	**GCRO**	**GCRO(ρ)**	**GCROT**	**OT**	**LGMRES**
olm2000	(23,37)	(37,-)	266	1069	681	1801	∞
watt___1	(4,7)	(7,3)	∞	297	296	481	131
tub1000	(11,18)	(18,-)	110	990	240	362	∞
cdde6	(4,7)	(7,8)	27	65	66	98	∞
cdde5	(9,15)	(15,-)	69	447	345	494	∞
odep400a	(9,15)	(15,-)	66	195	256	346	∞
cdde3	(4,7)	(7,-)	57	311	145	186	∞
rdb2048	(4,7)	(7,3)	71	271	217	278	1614
steam2	(6,11)	(11,-)	55	284	128	155	∞
lshp1561	(25,41)	(41,3)	406	1231	903	1072	909
494_bus	(6,11)	(11,3)	180	500	476	563	206
lshp1270	(18,30)	(30,3)	414	941	844	997	547
jagmesh9	(25,41)	(41,3)	324	1108	841	987	639
bfw62a	(4,7)	(7,6)	21	41	29	34	60
lshp_778	(27,45)	(45,3)	176	662	541	625	689

Agora comparamos o método GCROT com o método LGMRES. Recomendamos [36] para uma outra comparação entre estes métodos.

As tabelas 9.7 e 9.8 mostram os problemas para os quais o LGMRES apresentou uma performance melhor que o GCROT em número de iterações. Notamos que o

Tabela 9.6: Informações adicionais à tabela 9.5

Nome	**Dim.**	**Den.**	**Cond.**	**Norm.**	$\frac{\text{GCROT}}{\text{OT}}$	$\frac{\text{OT}}{\text{GCRO}(\rho)}$	$\frac{\text{GCROT}}{\text{Dim.}}$
olm2000	2000	0.20%	1.21E+07	1.34E+11	0.378	0.366	0.341
watt___1	1856	0.33%	5.38E+09	4.75E-007	0.615	0.329	0.159
tub1000	1000	0.40%	2.31E+06	8.20E+09	0.663	0.479	0.240
cdde6	961	0.51%	5.00E+02	9.37E+00	0.673	0.419	0.069
cdde5	961	0.51%	5.32E+04	3.75E-001	0.698	0.455	0.359
odep400a	400	0.75%	8.31E+05	2.00E+01	0.740	0.546	0.640
cdde3	961	0.51%	1.01E+04	3.75E-001	0.780	0.484	0.151
rdb2048	2048	0.29%	5.89E+03	1.39E+03	0.781	0.598	0.106
steam2	600	3.82%	3.55E+06	7.81E+14	0.826	0.568	0.213
lshp1561	1561	0.44%	4.53E+04	0.00E+00	0.842	0.532	0.578
494_bus	685	0.69%	5.31E+05	0.00E+00	0.845	0.593	0.695
lshp1270	1270	0.54%	6.61E+04	0.00E+00	0.847	0.492	0.665
jagmesh9	1349	0.50%	1.43E+04	0.00E+00	0.852	0.599	0.623
bfw62a	62	11.71%	1.48E+03	2.26E+01	0.853	0.667	0.468
lshp_778	778	0.87%	1.46E+04	0.00E+00	0.866	0.611	0.695
				Média	0.724	0.516	0.351

LGMRES apresentou um acréscimo significativo, com uma média de aproximadamente 35.4% do número de iterações do GCROT.

Tabela 9.7: Ganho mais significativo do LGMRES com relação ao GCROT

Nome	**G-Par**	**L-Par**	**GCRO**	**GCRO(ρ)**	**GCROT**	**OT**	**LGMRES**
rdb450l	(4,7)	(7,3)	40	195	167	152	37
rdb800l	(4,7)	(7,3)	47	198	308	184	72
bcsstm20	(4,7)	(7,3)	33	250	321	317	80
lshp2614	(23,37)	(37,3)	665	2594	2062	2221	593
lshp_406	(23,37)	(37,3)	100	374	271	304	83
rdb1250l	(4,7)	(7,3)	54	245	272	192	87
fs_183_6	(11,18)	(18,6)	∞	53	180	163	60
can___161	(18,30)	(30,3)	70	92	115	89	47
bcsstm23	(4,7)	(7,3)	144	803	965	927	395
fidap004	(11,18)	(18,3)	910	1551	1477	1393	608
rdb2048l	(4,7)	(7,3)	68	336	420	258	173
lshp2233	(16,26)	(26,3)	830	2105	1844	1717	767
jagmesh2	(16,26)	(26,3)	366	967	781	705	332
494_bus	(6,11)	(11,3)	180	500	476	563	206
pde2961	(4,7)	(7,3)	66	114	140	122	61

A tabela 9.9 mostra os casos em que o GCROT apresentou uma melhora superior ao LGMRES. Nestes casos, vemos que a melhora do GCROT pode ser muito significativa - uma média de 40.7%. Entretanto, lembramos que em muitos casos o LGMRES nem mesmo convergiu (ver tabela 9.1, por exemplo). Reparemos que nos casos onde o GCROT apresentou melhora mais significativa, o LGMRES convergiu com um número de iterações próximo da dimensão do problema, e portanto, em

Tabela 9.8: Informações adicionais à tabela 9.7

Nome	**Dim.**	**Den.**	**Cond.**	**Norm.**	$\frac{\text{LGMRES}}{\text{GCROT}}$	$\frac{\text{LGMRES}}{\text{Dim.}}$
rdb450l	450	1.27%	5.52E+02	1.57E+02	0.222	0.082
rdb800l	800	0.73%	9.65E+02	2.11E+02	0.234	0.090
bcsstm20	485	0.21%	2.55E+05	0.00E+00	0.249	0.165
lshp2614	2614	0.26%	1.30E+04	0.00E+00	0.288	0.227
lshp_406	406	1.65%	2.89E+03	0.00E+00	0.306	0.204
rdb1250l	1250	0.47%	1.38E+03	2.84E+02	0.320	0.070
fs_183_6	183	3.19%	1.50E+11	9.39E+17	0.333	0.328
can___161	161	5.31%	8.45E+02	0.00E+00	0.409	0.292
bcsstm23	3134	0.03%	9.46E+08	0.00E+00	0.409	0.126
fidap004	1601	1.26%	5.17E+03	0.00E+00	0.412	0.380
rdb2048l	2048	0.29%	2.07E+03	4.06E+02	0.412	0.084
lshp2233	2233	0.31%	2.33E+04	0.00E+00	0.416	0.343
jagmesh2	1009	0.67%	6.02E+03	0.00E+00	0.425	0.329
494_bus	685	0.69%	5.31E+05	0.00E+00	0.433	0.301
pde2961	2961	0.17%	9.49E+02	5.20E-01	0.436	0.021
				Média	0.354	0.203

alguns desses casos pode até mesmo ser considerado que o método não convergiu.

Tabela 9.9: Ganho mais significativo do GCROT com relação ao LGMRES

Nome	**G-Par**	**L-Par**	**GCRO**	**GCRO(ρ)**	**GCROT**	**OT**	**LGMRES**
rdb1250	(4,7)	(7,3)	61	172	133	130	1200
rdb2048	(4,7)	(7,3)	71	271	217	278	1614
psmigr_1	(11,18)	(18,14)	230	881	292	290	1690
cavity17	(9,15)	(15,14)	190	1193	838	868	4096
bcsstm06	(4,7)	(7,3)	24	57	65	53	161
bfw398a	(4,7)	(7,5)	54	203	136	117	306
bfw62a	(4,7)	(7,6)	21	41	29	34	60
dwt___221	(27,45)	(45,3)	81	81	81	81	166
cdde1	(4,7)	(7,3)	41	69	56	53	114
bfw782a	(4,7)	(7,7)	82	517	380	435	771
rdb450	(4,7)	(7,3)	36	88	79	63	157
steam1	(27,45)	(45,35)	99	178	123	121	236
ash85	(13,22)	(22,3)	42	42	42	42	80
bcspwr03	(16,26)	(26,3)	52	52	52	52	95
e30r0000	(4,7)	(7,3)	385	3536	3842	3699	6662

Porfim, a tabela 9.11 mostra um resumo da execução dos métodos. A coluna "Testes" mostra quantos problemas foram testados para o referido método, e a coluna "Falha" mostra para quantos dos problemas testados o método não conseguiu satisfazer o critério de convergência.

Como nem todos os métodos foram testados para os mesmos problemas, os dados da tabela 9.11 não nos passa uma idéia muito clara do comportamento dos métodos.

Tabela 9.10: Informações adicionais à tabela 9.9

Nome	**Dim.**	**Den.**	**Cond.**	**Norm.**	$\frac{\text{GCROT}}{\text{LGMRES}}$	$\frac{\text{GCROT}}{\text{Dim.}}$
rdb1250	1250	0.47%	3.72E+03	8.95E+02	0.111	0.11
rdb2048	2048	0.29%	5.89E+03	1.39E+03	0.134	0.11
psmigr_1	3140	5.51%	6.82E+09	2.66E+11	0.173	0.09
cavity17	4562	0.66%	1.44E+07	2.14E+01	0.205	0.18
bcsstm06	420	0.24%	3.46E+06	0.00E+00	0.404	0.15
bfw398a	398	2.32%	7.58E+03	2.26E+01	0.444	0.34
bfw62a	62	11.71%	1.48E+03	2.26E+01	0.483	0.47
dwt___221	221	3.34%	∞	0.00E+00	0.488	0.37
cdde1	961	0.51%	4.08E+03	3.75E-001	0.491	0.06
bfw782a	782	1.23%	3.36E+03	6.66E+01	0.493	0.49
rdb450	450	1.27%	1.64E+03	3.87E+02	0.503	0.18
steam1	240	6.53%	2.99E+07	1.55E+10	0.521	0.51
ash85	85	7.24%	1.93E+03	0.00E+00	0.525	0.49
bcspwr03	118	3.42%	1.95E+03	0.00E+00	0.547	0.44
e30r0000	9661	0.33%	6.93E+07	0.00E+00	0.577	0.398
				Média	0.407	0.292

Tabela 9.11: Resumo dos testes efetuados

Método	**Testes**	**Falhas**	**Taxa de Convergência**
GCRO(ρ)	474	227	52.110%
GCROT	286	61	78.671%
OT	250	26	89.600%
LGMRES	250	63	74.800%

Por isso, a tabela 9.12 mostra o resumo apenas dos 250 problemas que foram testados para o GCROT, OT e LGMRES.

Tabela 9.12: Resultado dos 250 problemas que foram testados para o GCROT, OT e LGMRES

Método	**Falhas**	**Taxa de Convergência**
GCROT	25	90.000%
OT	26	89.600%
LGMRES	63	74.800%

A tabela 9.13 mostra a razão entre o número de iterações de cada método. A primeira linha contém o número de problemas onde a razão é menor que 0.9, e portanto o método do numerador apresentou um número menor de iterações que o método do denominador. A segunda linha mostra o número de problemas onde a razão está entre 0.9 e 1.1, e portanto, a performance dos métodos foi considerada equivalente. Porfim, a terceira linha mostra o número de problemas onde a razão foi superior a 1.1, e portanto, o método do denominador apresentou um número de iterações menor com relação ao método do numerador. Nesta tabela, cada coluna

compara apenas os problemas onde ambos os métodos convergiram.

Tabela 9.13: Comparação entre a razão do número de iterações de cada método

Taxa (ρ)	$\frac{\text{GCROT}}{\text{GCRO}(\rho)}$	$\frac{\text{GCROT}}{\text{OT}}$	$\frac{\text{GCROT}}{\text{LGMRES}}$	$\frac{\text{LGMRES}}{\text{OT}}$
$\rho < 0.9$	89	21	41	112
$0.9 \leq \rho \leq 1.1$	108	171	22	23
$\rho > 1.1$	25	27	112	41

9.4 GCROT e OT - Comparação de FLOPS

Uma questão que pode ser levantada ao analisarmos a descrição do OT e do GCROT na seção 9.2 é quão menor deve ser o número de iterações executadas pelo OT com relação ao GCROT para que aa execução do OT seja mais rápida em tempo de CPU, uma vez que o truncamento do método OT é consideravelmente mais caro que o truncamento do método GCROT.

Para efetuar esta comparação, efetuamos uma contagem do *"número de operações de ponto flutuante"* - ou simplesmente, contagem de FLOPS.

O valor de cada operação é dado abaixo:

- $\langle u, v \rangle, u, v \in \mathbb{C}^n : 2n - 1$
- $\|u\|_2, u \in \mathbb{C}^n : 2n$
- $Uv, U \in \mathbb{C}^{m \times n}, v \in \mathbb{C}^n : m(2n - 1)$
- $UV, U \in \mathbb{C}^{m \times n}, V \in \mathbb{C}^{n \times t} : mt(2n - 1)$

Entretanto, durante a contagem, consideramos que a matriz de coeficientes $A \in \mathbb{C}^{n \times n}$ é esparsa, e que todo e qualquer vetor é denso. Como não hão multiplicações de A por uma matriz em nenhum dos dois algoritmos, a única alteração será

- $Av, v \in \mathbb{C}^n : 2\mu - n$

onde μ é o número de não-zeros da matriz esparsa A. Repare que caso A seja densa, então $\mu = n^2$ e teremos um caso equivalente ao anterior.

Consideremos agora os métodoss GCROT(ρ, τ, χ) e OT(ρ, τ, χ).

Proposição 9.4.1 (FLOPS do truncamento do GCROT e do OT)**.** *Sejam os métodos GCROT(ρ, τ, χ) e OT(ρ, τ, χ). Lembrando que $\phi = \tau + \chi$, a função*

$$t^{GT}(n, \rho, \tau, \chi) = 2n(\tau - 1)(2\phi - 3) + O(\rho) \tag{9.4.1}$$

nos dá o número de FLOPS executadas durante o truncamento do GCROT. A função

$$t^{OT}(n, \mu, \rho, \tau, \chi) = 2\rho\mu + 2n(\rho^2 + \rho(\phi - 1)) + t^{GT}(n, \rho, \tau, \chi) + O(\rho) \tag{9.4.2}$$

nos dá o número de FLOPS executadas durante o truncamento do OT, onde μ é o número de não-zeros da matriz A.

Esta proposição revela que a diferença durante o truncamento de cada um destes métodos pode ser muito grande, o que tornaria inviável o método OT. Entretanto, ao efetuar o cálculo de todo o algoritmo vemos que a ordem de complexidade do OT não é alterada em relação ao GCROT. Mostramos isso nas proposições a seguir:

Proposição 9.4.2. *A função*

$$f^{G}(n, \mu, \rho) = 2\rho\mu + n(2\rho^2 + 4\rho(\delta + 2) + 2\delta + 9) - O(\rho) \tag{9.4.3}$$

nos dá o número de FLOPS executado por uma única iteração do algoritmo 23 sem o truncamento, onde δ é a dimensão do subespaço de aumento na referida iteração.

Proposição 9.4.3. *Sejam os métodos GCROT(ρ, τ, χ) e OT(ρ, τ, χ). Denotemos por i^{OT} e i^{GT} o número de iterações do OT e do GCROT respectivamente, e t^{OT} o número de truncamentos efetuados pelo método OT. Então as funções*

$$f^{OT}(n, \mu, \rho, \tau, \chi, i^{OT}, t^{OT}) = 2\rho\mu(i^{OT} + t^{OT}) + O(n) + O(\rho) \tag{9.4.4}$$

$$f^{GT}(n, \mu, \rho, \tau, \chi, i^{GT}) = 2\rho\mu i^{GT} + O(n) + O(\rho) \tag{9.4.5}$$

dão o número de FLOPS da execução do OT e do GCROT respectivamente, onde μ é o número de não-zeros da matriz de coeficientes A do problema em questão.

Com estas proposições, podemos finalmente mostrar o teorema a seguir:

Teorema 9.4.4. *Quando o número de não-zeros da matriz A cresce para infinito, a razão entre o número de FLOPS do OT e do GCROT é $O(1)$, isto é, não depende do número de não-zeros.*

Demonstração. Basta apenas efetuar o limite entre as funções dadas na proposição 9.4.3, com μ tendendo ao infinito e teremos

$$\lim_{\mu \to \infty} \frac{f^{OT}}{f^{GT}} = \lim_{\mu \to \infty} \frac{2\rho\mu(i^{OT} + t^{OT})}{2\rho\mu i^{GT}} \tag{9.4.6}$$

$$= \frac{(i^{OT} + t^{OT})}{i^{GT}}. \tag{9.4.7}$$

Como os valores i^{OT}, i^{OT} e i^{OT} são constantes, trata-se então de $O(1)$. □

Caso a matriz A não seja esparsa, então o mesmo resultado obtido no teorema 9.4.4, mas ao invés do número de não-zeros deveríamos comparar com a dimensão n do problema. A demonstração e os cálculos são equivalentes.

Com este resultado podemos também estimar qual será a razão quando μ cresce para infinito. Temos que

$$t^{OT} \leq \frac{i^{OT} - \tau}{\chi} \tag{9.4.8}$$

$$\frac{i^{OT} + t^{OT}}{i^{GT}} \leq \frac{\frac{\chi+1}{\chi} i^{OT} - \frac{\tau}{\chi}}{i^{GT}} \leq \frac{(\chi - 1)i^{OT} - \tau}{\chi i^{GT}}. \tag{9.4.9}$$

9.5 Um Estudo de Caso - Sistema de Reação-Difusão

Durante os testes observamos que o OT apresentou uma convergência consideravelmente mais rápida que o GCROT para o problema de reação-difusão, muito comum na engenharia química.

Um sistema de reação-difusão é um modelo que descreve como um ou mais componentes distribuidos em um espaço são localmente transformados em outro componente através de uma reação química, e como eles se espalham neste espaço. A equação que rege tal sistema é dada por

$$\frac{\partial q}{\partial t} = D\nabla^2 q + R(q) \tag{9.5.1}$$

onde cada componente do vetor q representa um dos componentes em questão, D é uma matriz diagonal com os coeficientes da difusão, e R rege o fenômeo da reação entre as substâncias[1].

Utilizaremos o modelo Brusselator para solucionar a equação de reação-difusão, obtendo as equações

$$\frac{\partial u}{\partial t} = \frac{D_u}{L^2}\left(\frac{\partial^2 u}{\partial X^2} + \frac{\partial^2 u}{\partial Y^2}\right) - (B+1)u + u^2 v + C \tag{9.5.2}$$

$$\frac{\partial v}{\partial t} = \frac{D_v}{L^2}\left(\frac{\partial^2 v}{\partial X^2} + \frac{\partial^2 v}{\partial Y^2}\right) - u^2 v + Bu \tag{9.5.3}$$

para $u, v \in (0,1) \times (0,1)$ com as condições de fronteira de Dirichlet homogênea, formado um modelo de reação-difusão de duas dimensões, onde u e v representam a concentração das duas reações.

As equações foram discretizadas utilizando o método das diferenças centrais com

[1] boa parte desta informação foi retirada de [48]

uma malha de tamanho $hu = hv = 1/(n+1)$, onde $n = (N/2)^{1/2}$. Para

$$x^T = \left[u_{(1,1)}, v_{(1,1)}, u_{(1,2)}, u_{(1,2)}, \ldots, u_{(n,n)}, v_{(n,n)}\right], \tag{9.5.4}$$

as equações discretizadas podem ser escritas como $\dot{x} = f(x)$.

A matriz de coeficientes é simplesmente a matriz de Jacobi $A = \partial f/\partial x$, com os parâmetros $B = 5.45$, $C = 2$, $Du = 0.004$, $Dv = 0.008$. O parâmetro L varia; em alguns casos teremos $L = 0.5$ e em outros $L = 1.0$. As matrizes obtidas são reais não-hermitianas. O lado direito foi gerado aleatoriamente por rotinas do Matlab.

A tabela 9.14 mostra a comparação entre o GCROT e OT para a solução do problema mencionado, sem o uso de nenhum precondicionador. Como na seção anterior, i^{OT} e i^{GT} significam o número de iterações que cada método executou antes de atingir a convergência. As colunas f^{OT} e f^{GT} mostram o número de FLOPS que cada método executou. A coluna "Param." mostra os parâmetros (ρ, τ), usados no teste. Mais uma vez aqui tomamos $\chi = \tau$.

A tabela 9.15 nos dá informações adicionais a respeito do problema tratado. A coluna Dim. nos dá a dimensão do problema, a coluna "Nnz." nos dá o número de entradas diferentes de zero na matriz e a coluna "Den." nos dá a referida densidade. A coluna "Cond." nos dá o número de condicionamento da matriz, a coluna "Norm." nos dá o valor da distância à normalidade e porfim, a coluna L nos dá o valor escolhido para o parâmetro L durante a discretização do problema.

Tabela 9.14: Comparação entre a contagem de FLOPS do OT e do GCROT

Nome	i^{OT}	i^{GT}	$\frac{i^{OT}}{i^{GT}}$	f^{OT}	f^{GT}	$\frac{f^{OT}}{f^{GT}}$	Param.
rdb800l	121	239	0.506	4.3198E+07	8.2933E+07	0.521	(4,10)
rdb2048l	245	435	0.563	1.8234E+08	3.0280E+08	0.602	(4,7)
rdb1250l	127	226	0.562	9.8579E+07	1.6347E+08	0.603	(7,7)
rdb200	35	53	0.660	2.2911E+06	3.3163E+06	0.691	(4,7)
rdb450	148	229	0.646	1.9677E+07	2.7699E+07	0.710	(4,5)
rdb1250	287	465	0.617	8.6724E+07	1.1815E+08	0.734	(4,3)
rdb2048	141	195	0.723	1.4219E+08	1.7026E+08	0.835	(7,4)
		Média	0.611			0.671	

Podemos facilmente notar que em todos estes casos o método OT executou um número menor de FLOPS que o método GCROT, o que sugere que o método OT seja uma boa escolha para a solução deste tipo de problema, e que a média da razão entre o número de FLOPS foi bastante significante nestes casos.

Tabela 9.15: Informações adicionais à tabela 9.14

Nome	Dim.	Nnz.	Den.	Cond.	Norm.	L
rdb800l	800	4.6400E+03	0.725%	9.5084E+02	2.1069E+02	1.0
rdb2048l	2048	1.2032E+04	0.287%	1.9081E+03	4.0641E+02	1.0
rdb1250l	1250	7.3000E+03	0.467%	1.4029E+03	2.8385E+02	1.0
rdb200	200	1.1200E+03	2.800%	8.3178E+02	2.1503E+02	0.5
rdb450	450	2.5800E+03	1.274%	1.6382E+03	3.8678E+02	0.5
rdb1250	1250	7.3000E+03	0.467%	3.9802E+03	8.9465E+02	0.5
rdb2048	2048	1.2032E+04	0.287%	5.8897E+03	1.3852E+03	0.5

9.6 PETSc - Implementação Paralela

Para efetuar testes mais realistas, decidimos implantar os métodos GMRESR, GCRO, GCROT e OT numa línguagem de baixo nível. Decidimos então implantar estes métodos na biblioteca PETSc. De acordo com o próprio website do PETSc[49], *"trata-se de uma ferramenta com um conjunto de estruturas de dados e rotinas para solução escalável (paralela) de aplicações científicas modeladas por equações diferenciais parciais. O PETSc emprega os padrões MPI para efetuar paralelismo"*

As rotinas PESC são escritas na linguagem de programação C, mas são estruturadas de modo a apresentar alguns aspectos de uma linguagem orientada a objeto, como o conceito de classe e herança. Estas bibliotecas e objetos estão organizados em camadas de modo que é possível utilizar um nível de astração apropriado, evitando que o usuário final tenha que se preocupar com o funcionamento interno de outras rotinas. A figura 9.1, extraída de http://www.mcs.anl.gov/petsc/petsc-as/index.html, mostra a distribuição dessas camadas.

As seguintes bibliotecas são as mais relevantes para o nosso trabalho (a descrição foi tirada diretamente do site do PETSc)

libvec Fornece operações com vetores, requerida para a configuração e solução de problemas lineares e não lineares de larga escala. Inclui Provides the vector operations required for setting up and solving large-scale linear and nonlinear problems. Inclui dispersão e união de operações paralelas de fácil uso, bem como um código especial para lidar com "ghost points" para estruturas de dados regulares.

libmat Um grnade conjunto de estruturas de dados para manipulação em paralelo de matrizes esparsas. Inclui quatro diferentes estruturas de dados de matrizes esparsas, cada um apropriado para um tipo diferente de problema.

libksp

- Uma coleção de precondicionadores sequenciais e paralelos, incluindo (sequenciais) ILU(k), LU, e (sequenciais e paralelos) Jacobi em blocos, método de Aditivo de Shwarz com sobreposição e MG estruturado usando

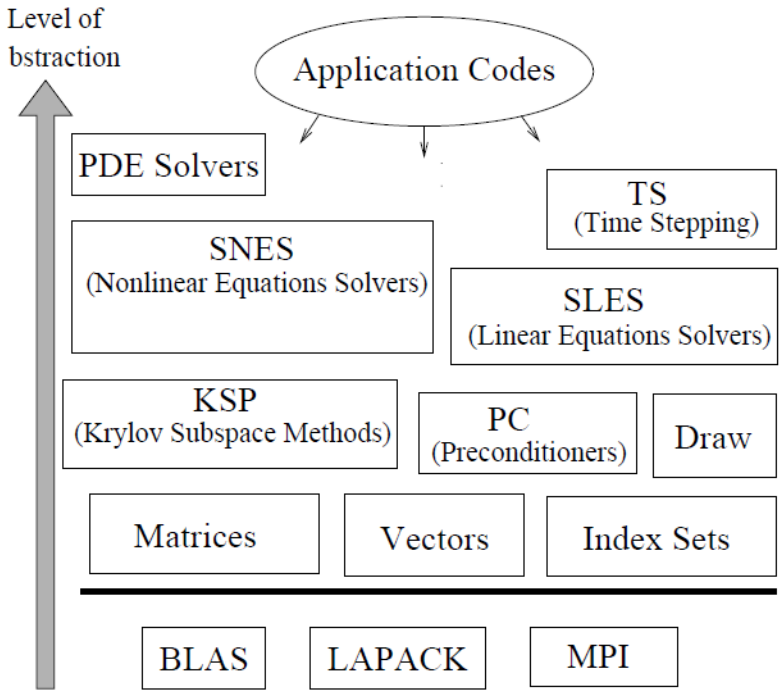

Figura 9.1: Organização das bibliotecas PETSc

DMMG.

- Implantação paralela de muitos dos populares iterativos métodos de subespaço de Krylov, incluindo GMRES, CG, CGS, BiCGStab, duas variantes do TFQMR, CR e LSQR. Todos foram codificados de modo que sejam imediatamente utilizáveis com qualquer um dos precondicionadores e estruturas de matrizes disponíveis, incluindo problemas onde a matriz de coeficientes é desconhecida.

Nós mostraremos brevemente como resolver um sistema linear utilizando as biblioteas do PETSc. Introduzimos o objeto *Vec*, que lida com as estruturas dos vetores, *Mat* que lida com as estruturas matriciais, e o objeto KSP, que lida de forma abstrata com qualquer método de Krylov implantado na biblioteca PETSc. Supondo que a matriz de coeficientes do sistema linear já tenha sido carregada no objeto A do tipo Mat, que o lado direito já tenha sido carregado no objeto b do tipo Vec, o objeto x do tipo Vec já tenha sido alocado e o objeto ksp do tipo KSP também já tenha sido declarado, então

KSPCreate($\overbrace{\text{PETSC\_COMM\_WORLD}}^{\text{Identificador de comunicação do MPI}}$,&ksp);

KSPSetOperators(ksp,A,A,DIFFERENT_NONZERO_PATTERN);

KSPSolve(ksp,b,x);

é o suficiente para resolver o sistema linear. A função KSPSolve guarda a solução encontrada em x, ou retorna um erro caso uma solução não tenha sido encontrada. Na função KSPSetOperators, o primeiro A representa a matriz de coeficientes do sistema linear a ser resolvido, o segundo A é um objeto do tipo Mat a ser utilizada para efetuar o precondicionamento[2], em geral a própria matriz de coeficientes, e a flag DIFFERENT_NONZERO_PATTERN é alterada apenas quando estamos resolvendo mais de um sistema linear.

Caso desejemos mudar o precondicionador a ser utilizado, basta utilizar as funções

```
KSPGetPC(ksp,&pc);
PCSetType(pc,PCSOR);
```

onde pc é um objeto do tipo PC, que gerencia o precondicionamento a ser utilizado em um objeto do tipo KSP. O exemplo acima muda o precondicionador para SOR.

A biblioteca PETSc também provê uma extensa lista de métodos de Krylov para solução de sistemas lineares. O método padrão utilizado pelo KSP é o CG para matrizes Hermitianas positiva definitas ou GMRES com recomeço para matrizes não-hermitianas. Similar ao que fizemos com o precondicionador, também podemos alterar o método a ser utilizado. Por exemplo, caso desejemos utilizar o método GCR, basta fazermos

```
KSPSetType(ksp,KSPGCR);
```

antes de chamar a função KSPSolve.

O PETSc também é construído de modo que alguns ajustes possam ser feitos após a compilação do código; se um arquivo executável *example.exe* utiliza as biblioteas PETSc, é possível alterar o método de Krylov a ser utilizado para resolver sistemas lineares, caso este código o faça. Basta simplesmente utilizarmos

```
example.exe -ksp_type gcr                    (9.6.1)
```

para forçar o uso do GCR, por exemplo.

Como o GCR é a base para os métodos GMRESR, GCRO, GCROT e OT, nós primeiramente implantamos o método GCR. Notemos que, embora a lista de métodos constantes na biblioteca PETSc seja grande, os métodos GCR, GMRESR, GCRO, GCROT e OT não constam desta lista.

[2] o precondicionador padrão para matrizes esparsas é Fatoração Incompleta ou Fatoração de Cholesky Incompleta para procedimentos sequenciais, e Jacobi em blocos com Fatoração Incompleta ou Fatoração de Cholesky Incompleta quando em paralelo. Para matrizes densas, o padrão é não efetuar nenhum tipo de precondicionamento

Até o momento, estamos trabalhando nos detalhes do precondicionamento do método GCR, embora para problemas não precondicionados o código já esteja maduro. Após mais alguns testes, trabalharemos na implantação dos demais métodos. Nós já implantamos as seguintes funcionalidades opcionais:

KSPSetFromOptions Permite a possibilidade de mudar aspectos do GCR passando parâmetreos via linha-de comando após o código haver sido compilado. Um exemplo é

$$\text{example.exe -ksp\_gcr\_restart 15} \tag{9.6.2}$$

que muda o tamanho máximo do subespaço do GCR, neste caso, forçando um recomeço de 15 em 15 iterações.

Gram-Schmidt Modificado e Gram-Schmidt Clássico Devido a instabilidades, em alguns casos pode ser preferível utilizar o método de Gram-Schmidt Modificado ao invés do método de Gram-Schmidt Clássico, que é mais rápido. Implantamos ambos os métodos, bastando apenas incluir a opção de alteração do método de ortogonalização via linha de comando -ksp_gcr_modified_gramschmidt" e "-ksp_gcr_classical_gramschmidt".

KSPView_GCR Função chamada pelo KSPView quando o GCR é utilizado para resolver um sistema linear. Esta função mostra informações úteis na tela, como o resíduo, o número de iterações, a estrutura da matriz A e informações do precondicionamento. No apêndice I temos um exemplo da saída produzida por essa função.

No apêndice H, mostramos um exemplo de um programa escrito em C utilizando a biblioteca PETSc para carregar um sistema linear, e resolve-lo utilizando o GCR. Trata-se do exemplo 10 que pode ser encontrado em

http://www.mcs.anl.gov/petsc/petsc-as/snapshots/petsc-current/src/ksp/ksp/examples/tutorials/ex10.c.html

na página oficial do PETSc, com pequenas alterações para execução do GCR sem precondicionamento.

Executamos alguns testes com matrizes esparsas de uma pequena coleção que pode ser encontrada em

http://ftp.mcs.anl.gov/pub/petsc/matrices/

Essas matrizes estão no formato específico do PETSc. Sem precondicionamento, confirmamos que a nossa implantação do método GCR é equivalente à implantação do método GMRES já presente no PETSc, como era de se esperar.

No apêndice G mostramos o código fonte da nossa implantação do GCR em PETSC, e no apêndice F mostramos o cabeçalho desta implantação (o arquivo .h). Entretanto, para compilar corretamente e testar todas as funcionalidades do código, faz-se necesário alterar alguns makefiles e cabelhaços nativos do PETSc.

Capítulo 10

Conclusão

Nesta dissertação, estudamos a estrutura e as propriedades básicas dos espaços de Krylov. Discutimos também a utilidade de espaços de Krylov na solução de sistemas lineares, durante a geração dos subespaços de correção. Estabelecemos as condições suficientes e necessárias para que a solução de um sistema linear esteja num determinado espaço de Krylov.

Vimos que, caso a solução esteja num espaço de Krylov, podemos encontrá-la através de construção de bases ortonormais com propriedades especiais, como a decomposição de Arnoldi, e da solução de um sistema de mínimos quadrados que, em geral, é menor que o problema original.

Mostramos vários métodos que utilizam a teoria por trás dos subespaços de Krylov para encontrar a solução de sistemas lineares de grande porte. Mostramos propriedades especiais já conhecidas de cada um desses métodos, e discutimos a otimalidade dos mesmos. Vimos que muito embora seja possível estabelecer alguns critérios para problemas específicos (como por exemplo, o método CG é um método considerado consolidado para a solução de sistemas lineares hermitianos positivo-definidos), é muito difícil estabelecer um método realizável e que seja aplicável a todo e qualquer tipo de problema.

Discutimos uma interpretação geométrica dos métodos e várias propriedades interessantes decorrentes desta interpetação geométrica, já discutidas por Eiermann e Ernst em [22]. Além disto, esta interpretação forneceu um excelente pano de fundo para a compreensão das discussões que se seguiram. Ainda em [22], graças à esta interpretação geométrica, foi possível classificar os métodos em duas grandes famílias, de modo a facilitar a enumeração de propriedades e relações entre os métodos, bem como o estudo dos mesmos.

Discutimos também, brevemente, a questão da taxa de convergência de um método e sua relação com os autovalores da matriz de coeficientes, e discutimos o fato de que a busca por um bom limitante superior para tal relação pode ser difícil de ser encontrada caso não consideremos que a matriz de coeficientes tenha propriedades

especiais, como normalidade.

Num próximo momento, discutimos o fato de algumas das estratégias mostradas, apesar de serem ótimas num sentido de garantir a convergência (ao menos considerando precisão infinita), não eram realizáveis. Por isto estudamos a técnica do recomeço e do truncamento para viabilizar a implantação destas estratégias. Com isto, a convergência ficava severamente prejudicada, podendo haver estagnação. Mostramos uma das possíveis origens da estagnação, e que trata-se de um problema relacionado à estrutura da matriz e à forma como o recomeço é implantado. Mostramos também uma interpretação diferente da estagnação, como sendo consequência da negligência da ortogonalização, também amplamente divugada na literatura.

Para atenuar os efeitos da negligência da ortogonalização, expomos a técnica da adição de um subespaço de aumento aos métodos previamente propostos que faziam uso do recomeço. Como o GMRES é o método mais utilizado por suas propriedades especiais, tratamos basicamente do método GMRES com recomeço e com algum subespaço de aumento. Discutimos algumas estratégias de aumento e o porquê de tais escolhas, e quando estas podem ser bem sucedidas. Além disso, discutimos uma interpretação dos métodos GCRO e GMRESR como métodos aumentados, que até onde nosso conhecimento se estende, é inédita. Esta nova visão nos ajuda a compreender e agrupar métodos em uma família com características mais similares.

Propomos também um método novo que chamamos de OT, que faz uso de uma estratégia de truncamento já conhecida e publicada, mas que não havia sido ainda explorada por ter sido considerada inviável.

Porfim mostramos e discutimos o resultado das nossas implantações em protótipos, escritos em Matlab. Mostramos uma extensa comparação entre os métodos LGMRES, GCRO, OT e GCROT, e como era de se esperar, nenhum dos métodos se mostrou definitivamente superior.

Em especial, destacamos um estudo de caso onde o método OT apresenta uma performance particularmente atrativa, mostrando que este método pode não só ser viável como fortemente recomendado para alguns problemas específicos.

Enfim, discutimos e exibimos o nosso código do método GCR escrito em C a ser adicionado à biblioteca PETSc. Trata-se de um código em paralelo usando a estrutura MPI, e que provê uma série de facilidades. Esperamos em breve concluir a implantação do método GCR e então dar início a implantação dos métodos GMRESR, GCRO, GCROT e OT a serem adicionados na mesma biblioteca.

Referências Bibliográficas

[1] MEYER, C. D. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001.

[2] SAAD, Y. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, April 2003.

[3] IPSEN, I. C. F., MEYER, C. D. "The Idea Behind Krylov Methods", *American Mathematical Monthly*, v. 105, pp. 889–899, 1998.

[4] GOLUB, G. H., VAN LOAN, C. F. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press, October 1996.

[5] BJÖRCK, A., PAIGE, C. C. "Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm", *SIAM J. Matrix Anal. Appl.*, v. 13, n. 1, pp. 176–190, 1992. ISSN: 0895-4798. doi: http://dx.doi.org/10.1137/0613015.

[6] MEURANT, G. *Computer Solution of Large Linear Systems*, v. 28, *Studies in Mathematics and its Applications*. Elsevier North Holland, 1999.

[7] BREZINSKI, C., ZAGLIA, M. R., SADOK, H., et al. "Problems of Breakdown and Near-Breakdown in Lanczos-Based Algorithms". , 1996.

[8] SAAD, Y. "Krylov Subspace Methods for Solving Large Unsymmetric Linear Systems", *Mathematical Computing*, v. 37, n. 155, pp. 105–126, 1981.

[9] 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.

[10] PAIGE, C. C., ROZLOŽNÍK, M., STRAKOS, Z. "Modified Gram-Schmidt (MGS), Least Squares, and Backward Stability of MGS-GMRES", *SIAM Journal on Matrix Analysis Applications*, v. 28, n. 1, pp. 264–284, 2006. ISSN: 0895-4798. doi: http://dx.doi.org/10.1137/050630416.

[11] ROZLOŽNÍK, M. *Numerical Stability of the GMRES Method*. Tese de Doutorado, Institute of Computer Science, Academy of Science of Czech Republic, December 1996.

[12] HESTENES, M. R., STIEFEL, E. "Methods of Conjugate Gradients for Solving Linear Systems", *Journal of Research of the National Bureau of Standards*, v. 49, n. 6, pp. 409–436, December 1952.

[13] LANCZOS, C. "Solution of Systems of linear equations by minimized iterations", *Journal of Research of the National Bureau of Standards*, v. 49, n. 1, pp. 33–53, July 1952.

[14] SHEWCHUK, J. R. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Relatório técnico, 1994.

[15] STIEFEL, E. "Relaxationsmethoden bester Strategie zur losung linearer Gleichungssystem", *Commentarii Mathematici Helvetici*, v. 29, n. 1, pp. 157–179, 1955.

[16] FREUND, R. W., NACHTIGAL, N. M. "QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems", *Numerische Mathematik*, v. 60, n. 1, pp. 315–339, December 1991.

[17] FLETCHER, R. "Conjugate gradient methods for indefinite systems." In: *Proceedings Dundee Conference on Numerical Analysis*, v. 506, *Lecture Notes in Mathematics*, Springer, Berlin, 1976.

[18] SONNEVELD, P. "CGS, a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems", *SIAM J. Sci. Stat. Comput.*, v. 10, n. 1, pp. 36–52, 1989. ISSN: 0196-5204. doi: http://dx.doi.org/10.1137/0910004.

[19] VAN DER VORST, H. A. "The Convergence Behaviour of Preconditioned CG and CG-S in the Presence of Rounding Errors". In: *Proceedings of a conference on Preconditioned s gradient methods*, pp. 126–136, New York, NY, USA, 1991. Springer-Verlag New York, Inc. ISBN: 0-387-53515-2.

[20] VAN DER VORST, H. A. "BI-CGSTAB: a Fast and Smoothly Converging Variant of BI-CG for the Solution of Nonsmmetric Linear Systems", *SIAM Journal on Scientific and Statistical Computing*, v. 13, n. 2, pp. 631–644, March 1992.

[21] EISENSTAT, S. C., ELMAN, H. C., SCHULTZ, M. H. "Variational Iterative Methods for Nonsymmetric System of Linear Equations", *SIAM Journal on Numerical Analysis*, v. 20, n. 2, April 1983.

[22] EIERMANN, M., ERNST, O. G. "Geometric aspects in the theory of Krylov subspace methods", *Acta Numerica*, v. 10, n. 10, pp. 251–312, 2001.

[23] WEDIN, P. Å. "On Angles Between Wubspaces of a Finite Dimensional Inner Product Space", *Lecture Notes in Mathematics*, v. Volume 973, pp. 263–285, 1983. ISSN: 978-3-540-11983-8.

[24] GUTKNECHT, M. H., ROZLOŽNIK, M. "By How Much Can Residual Minimization Accelerate The Convergence Of Orthogonal Residual Methods?" , 2001.

[25] EIERMANN, M., ERNST, O. G., SCHNEIDER, O. "Analysis of acceleration strategies for restarted minimal residual methods", *Journal of Computational and Applied Mathematics*, v. 123, n. 1-2, pp. 261–292, 2000.

[26] SIMONCINI, V., SZYLD, D. B. "Recent computational developments in Krylov subspace methods for linear systems", *Numerical Linear Algebra with Applications*, v. 14, pp. 1–59, 2007.

[27] GREENBAUM, A., STRAKOS, Z. "Matrices that Generate the Same Krylov Residual Spaces". In: *in Recent Advances in Iterative Methods*, pp. 95–118. Springer, 1994.

[28] MORGAN, R. B. "A Restarted GMRES Method Augmented with Eigenvectors", *SIAM J. Matrix Anal. Appl.*, v. 16, n. 4, pp. 1154–1171, 1995. ISSN: 0895-4798. doi: http://dx.doi.org/10.1137/S0895479893253975.

[29] FREUND, R. W. "Quasi-kernel Polynomials and Convergence Results for Quasi-Minimal Residual Iterations", v. 9, pp. 77–95, 1992.

[30] GREENBAUM, A., PTÁK, V., STRAKOUS, Z. "Any Nonincreasing Convergence Curve is Possible for GMRES", *SIAM Journal on Matrix Analysis Applications*, v. 17, n. 3, pp. 465–469, 1996.

[31] IPSEN, I. C. F. "Expressions And Bounds For The GMRES Residual", *BIT*, v. 40, pp. 524–533, 1999.

[32] VAN DER VORST, H. A., VUIK, K. *GMRESR: A family of nested GMRES methods*. Relatório técnico, Faculty of Technical Matehmatics and Informatics, 1994.

[33] SZYLD, D. B., SIMONCINI, V. "Flexible Inner-Outer Krylov Subspace Methods", *SIAM J. Numer. Anal*, v. 40, pp. 2002, 2002.

[34] SAAD, Y. "A Flexible Inner-Outer Preconditioned GMRES Algorithm". , 1993.

[35] DE STURLER, E. "Nested Krylov methods based on GCR", *J. Comput. Appl. Math.*, v. 67, n. 1, pp. 15–41, 1996. ISSN: 0377-0427. doi: http://dx.doi.org/10.1016/0377-0427(94)00123-5.

[36] BAKER, A. H., JESSUP, E. R., MANTEUFFEL, T. "A Technique for Accelerating the Convergence of Restarted GMRES", *SIAM J. Matrix Anal. Appl.*, v. 26, n. 4, pp. 962–984, 2005. ISSN: 0895-4798. doi: http://dx.doi.org/10.1137/S0895479803422014.

[37] CARVALHO, L. M., GRATON, S. *Avanços em Métodos de Krylov para Solução de Sistemas Lineares de Grande Porte*, v. 49, *Notas em Matemática Aplicada*. Sociedade Brasileira de Matemática Aplicada e Computacional, 2009.

[38] MORGAN, R. B. "Implicitly Restarted GMRES and Arnoldi Methods for Nonsymmetric Systems of Equations", *SIAM J. Matrix Anal. Appl.*, v. 21, n. 4, pp. 1112–1135, 2000. ISSN: 0895-4798. doi: http://dx.doi.org/10.1137/S0895479897321362.

[39] STEWART, G. W. *Matrix Algorithms: Volume 1, Basic Decompositions*. Society for Industrial Mathematics, 1998. ISBN: 0898714141.

[40] DE STURLER, E. "Truncation Strategies for Optimal Krylov Subspace Methods", *SIAM Journal on Numerical Analysis*, v. 36, n. 3, pp. 864–889, 1999.

[41] HORN, R. A., JOHNSON, C. R. *Matrix Analysis*. Cambridge University Press, February 1990. ISBN: 0521386322.

[42] CHAITIN-CHATELIN, F. *Eingenvalues of Matrices*. Wiley, New York, 1993.

[43] FOKKEMA, D. R. "Hybrid Methods Based on GCR Principle", . to appear.

[44] DE STURLER, E., FOKKEMA, D. R. "Nested Krylov Methods and Preserving Orthogonality". In: Melson, N. D., Manteuffel, T., McCormick, S. (Eds.), *6th Copper Mountain Conference on Multigrid Methods*, v. Part 1, *NASA Conf. Publication 3224*, pp. 111–125. NASA Conf. Publication 3224, 1993.

[45] LAGO, R. F., CARVALHO, L. M. "Optimal Truncated Method Based on GCR", . to appear, 2010.

[46] FOKKEMA, D. R. *Subspace Methods for Linear, Nonlinear, and Eigen Problems*. Tese de Doutorado, Universiteit Utrecht, 1996.

[47] “Matrix Market”. , 2009. Disponível em: <http://math.nist.gov/MatrixMarket/>. Acesso em: 23 set. 2009.

[48] “wikipedia.org”. , 2010. Disponível em: <http://en.wikipedia.org/>. Acesso em: 12 jan. 2010.

[49] BALAY, S., BUSCHELMAN, K., GROPP, W. D., et al. “PETSc Web page”. , 2009. http://www.mcs.anl.gov/petsc.

Apêndice A

GCRO - Protótipo em MATLAB

```
1  function [R RES_HST] = GCRO(PROBLEM, INN_MAX, OUT_DIM, TRC_ADD, OUT_MAX,
2                              TOL)
3  %-------------------------------------------------------------------------
4  % GCRO is a nested Minimal Residual Method derivated from GMRESR method. It
5  %   performs an orthogonalization between inner and outer approximation
6  %   subspaces to improve the residual minimization.
7  %
8  %   E. de Sturler, "Nested Krylov Methods Base on GCR", J. Comput. Appl.
9  %   Math., 67 (1996), pp.15-41
10 %-------------------------------------------------------------------------
11 %
12 % [R RES_HST] = GCRO(PROBLEM,INN_MAX,OUT_DIM,TRC_ADD,OUT_MAX,TOL)
13 %
14 % R is a structure with some informations about the algorithm and the
15 % problem solved
16 %
17 % RES_HST is a vector that provides a history with the norm of residuals of
18 % all iterations
19 %
20 % PROBLEM is the path to the .m file containing the matrix A and the right
21 % hand side b. It may also contain the initial guess x; if it doesn't then
22 % the zero vector is used as initial guess. PROBLEM is a mandatory
23 % parameter
24 %
```

```
25 % INN_MAX is the maximum number of inner iterations. If not given, then the
26 % default number of inner iterations (which is 10) is used.
27 %
28 % OUT_DIM is the dimension of the outer subspace to be kept after
29 % truncation. To perform no truncation just set OUT_DIM = inf. The default
30 % value is 10
31 %
32 % TRC_ADD is the number of vectors to be added before truncating. That is,
33 % the algorithm will truncate when the number of OUTER vectors reaches
34 % OUT_DIM + TRC_ADD, and will keep only OUT_DIM vectors. Setting this
35 % value bigger than one avoid the algorithm to truncate every iteration.
36 % The default value is 1 (truncate every iteration)
37 %
38 % OUT_MAX is the maximum number of outer iterations. If not given, then the
39 % default number (that is the dimension of the matrix A) is used.
40 %
41 % TOL is the tolerance for convergence to be used. If not given, then the
42 % value given by the PROBLEM file is used or the default value (1e-10)
43 %-------------------------------------------------------------------------
44 %
45
46 %-------------------------------------------------------------------------
47 % Parameters Check
48 %-------------------------------------------------------------------------
49 if    exist('PROBLEM')
50     error('The PROBLEM parameter is mandatory. Exiting...');
51 else
52     if exist(PROBLEM,'file') || exist( strcat(PROBLEM,'.mat'),'file' )
53         load(PROBLEM);
54         % TODO: check if the loaded file contains the expected structure
55     else
56         error('File "%s" not found. Exiting...',PROBLEM);
57     end
```

```
58 end
59
60 if    exist('INN_MAX') || isempty(INN_MAX)
61     INN_MAX = 10;
62 end
63
64 if    exist('OUT_DIM') || isempty(OUT_DIM)
65     OUT_DIM = 10;
66 end
67
68 if    exist('TRC_ADD') || isempty(TRC_ADD)
69     TRC_ADD = 1;
70 end
71
72 if    exist('OUT_MAX') || isempty(OUT_MAX)
73     OUT_MAX = dim;
74 end
75
76 if    exist('x')
77     x = zeros(dim,1);
78     ZERO_INIT_GUESS = 'True';
79 else
80     ZERO_INIT_GUESS = 'False';
81 end
82
83 if    exist('TOL') || isempty(TOL)
84     if    exist('tol')
85         TOL = tol;
86     else
87         TOL = 1e-10;
88     end
89 end
90 %-----------------------------------------------------------------------
```

```
91
92
93 %-------------------------------------------------------------------------
94 % Initialization...
95 %-------------------------------------------------------------------------
96 i = 1;      DIM_W1 = 0;
97
98 r1 = b - A*x;
99 RES_HST(1) = norm(r1);
100
101 V1 = zeros(dim,1);    B = zeros(1,INN_MAX);
102 W1 = [];              H = [];
103
104 fprintf(1,'\nInitializing GCRO method\nParameters:( %i,%i,',INN_MAX,OUT_DIM);
105 fprintf(1,'%i,%i,%e)\n',TRC_ADD,OUT_MAX,TOL);
106 fprintf(1,'Problem :  %s\nDimension:%i\n',PROBLEM, dim);
107 fprintf(1,'\nNorm of residual:  %.10e',RES_HST(1));
108
109 %-------------------------------------------------------------------------
110
111
112
113 %-------------------------------------------------------------------------
114 % Main Loop
115 %   This implementation is basically a GCROT method with a "naive"
116 %   truncation strategy instead. For a more efficient implementation of
117 %   GCRO, use GCRO_lite
118 %-------------------------------------------------------------------------
119 while (RES_HST(i)/RES_HST(1) > TOL) && (i <= OUT_MAX)
120
121    V2 = r1 / RES_HST(i);
122
123     %-------------------------------------------------------------------------
```

```
124     % Apply (inner) GMRES
125     %-----------------------------------------------------------------------
126     for j = 1:INN_MAX
127         V2(:,j+1) = A * V2(:,j);
128
129         %Orthogonalize with W1
130         for l = 1:DIM_W1
131             B(l,j) = W1(:,l)' * V2(:,j+1);
132             V2(:,j+1) = V2(:,j+1) - B(l,j) * W1(:,l);
133           end
134
135           %Orthogonalize with V2
136         for l = 1:j
137                 H(l,j) = V2(:,l)' * V2(:,j+1);
138             V2(:,j+1) = V2(:,j+1) - H(l,j) * V2(:,l);
139           end
140
141           %TODO: Add a norm calculation and verify if its small enougth - in
142           % that case we don't need to perform all inner iterations
143
144         H(j+1,j) = norm( V2(:,j+1) );
145         V2(:,j+1) = V2(:,j+1) / H(j+1,j);
146     end
147
148     % qr(a,0) calculates the 'economic' QR decomposition instead of
149     % complete one, that is what we want
150     [Q R] = qr(H,0);
151     R = inv(R);       % We use only the inverse of R
152
153         y = RES_HST(i) * R * Q' * eye(INN_MAX+1,1);
154      new_w = V2 * H * y;
155      new_v = (V2(:,1:INN_MAX) - V1 * B ) * y;
156
```

```
157     r1 = r1 - new_w;
158     x = x + new_v;
159    RES_HST(i+1) = norm(r1);
160
161     % Truncate if necessary
162     % Remembering that we already have one vector to add, so we have
163     % DIM_W1 + 1 vectors
164    if DIM_W1 +1 >= OUT_DIM + TRC_ADD
165       W1 = W1(:,TRC_ADD+1:DIM_W1);
166       V1 = V1(:,TRC_ADD+1:DIM_W1);
167         B = []; % Prevents problems with size without complicating the code
168       DIM_W1 = OUT_DIM - 1;
169     end
170
171       % Update V1 and W1
172    alpha = norm(new_w);
173    V1(:,DIM_W1+1) = new_v / alpha;
174    W1(:,DIM_W1+1) = new_w / alpha;
175
195     fprintf(1,'\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b %.10e',RES_HST(i+1));
177
178     DIM_W1 = DIM_W1 + 1;
179     i = i + 1;
180 end
181 %-------------------------------------------------------------------------
182 if ( i > OUT_MAX )
183     error('Algorithm failed to converge: number of maximum iterations
184           exceeded');
185 else
186     fprintf('\n\nAlgorithm has converged.\nNumber of Iterations:  %i',i-1);
187     fprintf('\nNorm of True Residual:  %e\n',RES_HST(i));
188 end
189
```

```
190 RES_HST(i+1) = norm(b - A*x);  % True Residual
191
192 R = struct('PROBLEM',PROBLEM,'INN_MAX',INN_MAX,'OUT_DIM',OUT_DIM,'OUT_MAX',...
193     OUT_MAX,'DIMENSION',dim,'TRC_ADD',TRC_ADD,'DENSITY',density,...
194     'TOLERANCE',TOL,'ZERO_INIT_GUESS',ZERO_INIT_GUESS,'NUM_ITERATIONS',i-1,...
195     'TRUE_RESIDUAL_NORM',RES_HST(i+1));
196
197 end
```

Apêndice B

GCROT - Protótipo em MATLAB

```
1  function [R RES_HST] = GCROT(PROBLEM, INN_MAX, OUT_DIM, TRC_ADD, OUT_MAX,
2                               TOL)
3  %-------------------------------------------------------------------------
4  % GCROT is the improvement of GCRO method. It peforms a truncation in the
5  %   outer subspace by using information about angles between subspaces
6  %
7  %   E. de Sturler, "Truncation Strategies for Optimal Krylov Subspace
8  %   Methods", SIAM J. Numer. Anal., 36 (1999), pp.864-889
9  %-------------------------------------------------------------------------
10 %
11 % [R RES_HST] = GCROT(PROBLEM,INN_MAX,OUT_DIM,TRC_ADD,OUT_MAX,TOL)
12 %
13 % R is a structure with some informations about the algorithm and the
14 % problem solved
15 %
16 % RES_HST is a vector that provides a history with the norm of residuals of
17 % all iterations
18 %
19 % PROBLEM is the path to the .m file containing the matrix A and the right
20 % hand side b. It may also contain the initial guess x; if it doesn't then
21 % the zero vector is used as initial guess. PROBLEM is a mandatory
22 % parameter
23 %
24 % INN_MAX is the maximum number of inner iterations. If not given, then the
```

```
25 % default number of inner iterations (which is 10) is used.
26 %
27 % OUT_DIM is the dimension of the outer subspace to be kept after
28 % truncation. To perform no truncation just set OUT_DIM = inf. The default
29 % value is 10
30 %
31 % TRC_ADD is the number of vectors to be added before truncating. That is,
32 % the algorithm will truncate when the number of OUTER vectors reaches
33 % OUT_DIM + TRC_ADD, and will keep only OUT_DIM vectors. Setting this
34 % value bigger than one avoid the algorithm to truncate every iteration.
35 % The default value is 1 (truncate every iteration)
36 %
37 % OUT_MAX is the maximum number of outer iterations. If not given, then the
38 % default number (that is the dimension of the matrix A) is used.
39 %
40 % TOL is the tolerance for convergence to be used. If not given, then the
41 % value given by the PROBLEM file is used or the default value (1e-10)
42 %-------------------------------------------------------------------------
43 %
44
45 %-------------------------------------------------------------------------
46 % Parameters Check
47 %-------------------------------------------------------------------------
48 if    exist('PROBLEM')
49     error('The PROBLEM parameter is mandatory. Exiting...');
50 else
51     if exist(PROBLEM,'file') || exist( strcat(PROBLEM,'.mat'),'file' )
52         load(PROBLEM);
53         % TODO: check if the loaded file contains the expected structure
54     else
55         error('File "%s" not found. Exiting...',PROBLEM);
56     end
57 end
```

```
58
59 if    exist('INN_MAX') || isempty(INN_MAX)
60     INN_MAX = 10;
61 end
62
63 if    exist('OUT_DIM') || isempty(OUT_DIM)
64     OUT_DIM = 10;
65 end
66
67 if    exist('TRC_ADD') || isempty(TRC_ADD)
68     TRC_ADD = 1;
69 end
70
71 if    exist('OUT_MAX') || isempty(OUT_MAX)
72     OUT_MAX = dim;
73 end
74
75 if    exist('x')
76     x = zeros(dim,1);
77     ZERO_INIT_GUESS = 'True';
78 else
79     ZERO_INIT_GUESS = 'False';
80 end
81
82 if    exist('TOL') || isempty(TOL)
83     if    exist('tol')
84         TOL = tol;
85     else
86         TOL = 1e-10;
87     end
88 end
89 %--------------------------------------------------------------------------
90
```

```
91
92 %-------------------------------------------------------------------
93 % Initialization...
94 %-------------------------------------------------------------------
95 i = 1;      DIM_W1 = 0;
96
97 r1 = b - A*x;
98 RES_HST(1) = norm(r1);
99
100 V1 = zeros(dim,1);    B = zeros(1,INN_MAX);
101 W1 = [];              H = [];
102
103 fprintf(1,'\nInitializing GCROT method\nParameters:( %i,%i,',INN_MAX,OUT_DIM);
104 fprintf(1,'%i,%i,%e)\n',TRC_ADD,OUT_MAX,TOL);
105 fprintf(1,'Problem :  %s\nDimension:%i\n',PROBLEM, dim);
106 fprintf(1,'\nNorm of residual:  %.10e',RES_HST(1));
107
108
109 %-------------------------------------------------------------------
110
111
112
113 %-------------------------------------------------------------------
114 % Main Loop
115 %-------------------------------------------------------------------
116 while (RES_HST(i)/RES_HST(1) > TOL) && (i <= OUT_MAX)
117
118    V2 = r1 / RES_HST(i);
119
120     %-------------------------------------------------------------------
121     % Apply (inner) GMRES
122     %-------------------------------------------------------------------
123    for j = 1:INN_MAX
```

```
124       V2(:,j+1) = A * V2(:,j);
125
126       %Orthogonalize with W1
127       for l = 1:DIM_W1
128          B(l,j) = W1(:,l)' * V2(:,j+1);
129          V2(:,j+1) = V2(:,j+1) - B(l,j) * W1(:,l);
130         end
131
132         %Orthogonalize with V2
133       for l = 1:j
134             H(l,j) = V2(:,l)' * V2(:,j+1);
135          V2(:,j+1) = V2(:,j+1) - H(l,j) * V2(:,l);
136         end
137
138         %TODO: Add a norm calculation and verify if its small enougth - in
139         % that case we don't need to perform all inner iterations
140
141       H(j+1,j) = norm( V2(:,j+1) );
142       V2(:,j+1) = V2(:,j+1) / H(j+1,j);
143    end
144
145    % qr(a,0) calculates the 'economic' QR decomposition instead of
146    % complete one, that is what we want
147    [Q R] = qr(H,0);
148    R = inv(R);       % We use only the inverse of R
149
150       y = RES_HST(i) * R * Q' * eye(INN_MAX+1,1);
151     new_w = V2 * H * y;
152     new_v = (V2(:,1:INN_MAX) - V1 * B ) * y;
153
154     r1 = r1 - new_w;
155     x = x + new_v;
156    RES_HST(i+1) = norm(r1);
```

```
157
158     % Truncate if necessary
159     % Remembering that we already have one vector to add, so we have
160     % DIM_W1 + 1 vectors
161    if DIM_W1 +1 >= OUT_DIM + TRC_ADD
162
163         [X, S , S] = svd( B * R );
164       W1 = W1*X(:,TRC_ADD+1:DIM_W1);
165       V1 = V1*X(:,TRC_ADD+1:DIM_W1);
166
167         B = []; % Prevents problems with size without complicating the code
168       DIM_W1 = OUT_DIM - 1;
169     end
170
171       % Update V1 and W1
172    alpha = norm(new_w);
173    V1(:,DIM_W1+1) = new_v / alpha;
174    W1(:,DIM_W1+1) = new_w / alpha;
175
176     fprintf(1,'\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b %.10e',RES_HST(i+1));
177
178     DIM_W1 = DIM_W1 + 1;
179     i = i + 1;
180 end
181 %-----------------------------------------------------------------------
182
183 if ( i > OUT_MAX )
184     error('Algorithm failed to converge: number of maximum iterations
185           exceeded');
186 else
187     fprintf('\n\nAlgorithm has converged.\nNumber of Iterations:  %i',i-1);
188     fprintf('\nNorm of True Residual:  %e\n',RES_HST(i));
189 end
```

```
190
191 RES_HST(i+1) = norm(b - A*x);  % True Residual
192
193 R = struct('PROBLEM',PROBLEM,'INN_MAX',INN_MAX,'OUT_DIM',OUT_DIM,'OUT_MAX',...
194     OUT_MAX,'DIMENSION',dim,'TRC_ADD',TRC_ADD,'DENSITY',density,...
195     'TOLERANCE',TOL,'ZERO_INIT_GUESS',ZERO_INIT_GUESS,'NUM_ITERATIONS',i-1,...
196     'TRUE_RESIDUAL_NORM',RES_HST(i+1));
197
198 end
```

Apêndice C

OT - Protótipo em MATLAB

```
1  function [R RES_HST] = OT(PROBLEM, INN_MAX, OUT_DIM, TRC_ADD, OUT_MAX, TOL)
2  %-------------------------------------------------------------------------
3  % OT is the GCROT method, but with the optimal truncation. In his article,
4  %   de Sturler proposes the Optimal truncation but he also states that its
5  %   not possible to perform such truncation because GCRO does not have the
6  %   proper basis. Here we generate such basis just to test what kind of
7  %   improvement the optimal truncation would bring to the method
8  %
9  %   R. F. Lago, e L. M. Carvalho, Optimal Truncated Method Based on
10 %   GCR", to appear
11 %-------------------------------------------------------------------------
12 %
13 % [R RES_HST] = OT(PROBLEM,INN_MAX,OUT_DIM,TRC_ADD,OUT_MAX,TOL)
14 %
15 % R is a structure with some informations about the algorithm and the
16 % problem solved
17 %
18 % RES_HST is a vector that provides a history with the norm of residuals of
19 % all iterations
20 %
21 % PROBLEM is the path to the .m file containing the matrix A and the right
22 % hand side b. It may also contain the initial guess x; if it doesn't then
23 % the zero vector is used as initial guess. PROBLEM is a mandatory
24 % parameter
```

```
25 %
26 % INN_MAX is the maximum number of inner iterations. If not given, then the
27 % default number of inner iterations (which is 10) is used.
28 %
29 % OUT_DIM is the dimension of the outer subspace to be kept after
30 % truncation. To perform no truncation just set OUT_DIM = inf. The default
31 % value is 10
32 %
33 % TRC_ADD is the number of vectors to be added before truncating. That is,
34 % the algorithm will truncate when the number of OUTER vectors reaches
35 % OUT_DIM + TRC_ADD, and will keep only OUT_DIM vectors. Setting this
36 % value bigger than one avoid the algorithm to truncate every iteration.
37 % The default value is 1 (truncate every iteration)
38 %
39 % OUT_MAX is the maximum number of outer iterations. If not given, then the
40 % default number (that is the dimension of the matrix A) is used.
41 %
42 % TOL is the tolerance for convergence to be used. If not given, then the
43 % value given by the PROBLEM file is used or the default value (1e-10)
44 %--------------------------------------------------------------------------
45 %
46
47 %--------------------------------------------------------------------------
48 % Parameters Check
49 %--------------------------------------------------------------------------
50 if    exist('PROBLEM')
51     error('The PROBLEM parameter is mandatory. Exiting...');
52 else
53     if exist(PROBLEM,'file') || exist( strcat(PROBLEM,'.mat'),'file' )
54         load(PROBLEM);
55         % TODO: check if the loaded file contains the expected structure
56     else
57         error('File "%s" not found. Exiting...',PROBLEM);
```

```
58      end
59  end
60
61  if    exist('INN_MAX') || isempty(INN_MAX)
62      INN_MAX = 10;
63  end
64
65  if    exist('OUT_DIM') || isempty(OUT_DIM)
66      OUT_DIM = 10;
67  end
68
69  if    exist('TRC_ADD') || isempty(TRC_ADD)
70      TRC_ADD = 1;
71  end
72
73  if    exist('OUT_MAX') || isempty(OUT_MAX)
74      OUT_MAX = dim;
75  end
76
77  if    exist('x')
78      x = zeros(dim,1);
79      ZERO_INIT_GUESS = 'True';
80  else
81      ZERO_INIT_GUESS = 'False';
82  end
83
84  if    exist('TOL') || isempty(TOL)
85      if    exist('tol')
86          TOL = tol;
87      else
88          TOL = 1e-10;
89      end
90  end
```

```
91 %-------------------------------------------------------------------------
92
93 NNZ = nnz(A);
94 %-------------------------------------------------------------------------
95 % Initialization...
96 %-------------------------------------------------------------------------
97 i = 1;      DIM_W1 = 0;
98
99 r1 = b - A*x;
100 RES_HST(1) = norm(r1);
101
102 V1 = zeros(dim,1);    B = zeros(1,INN_MAX);
103 W1 = [];              H = [];
104
105 fprintf(1,'\nInitializing OT method\nParameters:( %i,%i,',INN_MAX,OUT_DIM);
106 fprintf(1,'%i,%i,%e)\n',TRC_ADD,OUT_MAX,TOL);
107 fprintf(1,'Problem :  %s\nDimension:%i\n',PROBLEM, dim);
108 fprintf(1,'\nNorm of residual:  %.10e',RES_HST(1));
109
110 %-------------------------------------------------------------------------
111
112 %-------------------------------------------------------------------------
113 % Main Loop
114 %-------------------------------------------------------------------------
115 while (RES_HST(i)/RES_HST(1) > TOL) && (i <= OUT_MAX)
116
117    V2 = r1 / RES_HST(i);
118
119     %-------------------------------------------------------------------------
120     % Apply (inner) GMRES
121     %-------------------------------------------------------------------------
122    for j = 1:INN_MAX
123       V2(:,j+1) = A * V2(:,j);
```

```
124
125       %Orthogonalize with W1
126       for l = 1:DIM_W1
127          B(l,j) = W1(:,l)' * V2(:,j+1);
128          V2(:,j+1) = V2(:,j+1) - B(l,j) * W1(:,l);
129         end
130
131         %Orthogonalize with V2
132       for l = 1:j
133             H(l,j) = V2(:,l)' * V2(:,j+1);
134             V2(:,j+1) = V2(:,j+1) - H(l,j) * V2(:,l);
135         end
136
137         %TODO: Add a norm calculation and verify if its small enougth - in
138         % that case we don't need to perform all inner iterations
139
140       H(j+1,j) = norm( V2(:,j+1) );
141       V2(:,j+1) = V2(:,j+1) / H(j+1,j);
142    end
143
144    % qr(a,0) calculates the 'economic' QR decomposition instead of
145    % complete one, that is what we want
146    [Q R] = qr(H,0);
147    R = inv(R);       % We use only the inverse of R
148
149       y = RES_HST(i) * R * Q' * eye(INN_MAX+1,1);
150     new_w = V2 * H * y;
151     new_v = V2(:,1:INN_MAX)*y;
152
153     new_v = new_v - V1 * (B * y);
154
155     r_old = r1;
156     r1 = r1 - new_w;
```

```
157     x = x + new_v;
158     RES_HST(i+1) = norm(r1);
159
160     % Truncate if necessary
161     % Remembering that we already have one vector to add, so we have
162     % DIM_W1 + 1 vectors
163    if DIM_W1 +1 >= OUT_DIM + TRC_ADD
164
165         % Generate a basis to W2
166         % It could be any basis, but we generate an orthonormal basis to
167         % ensure that it will no degenerate
168         W2 = A * r_old;
169         W2 = W2 / norm(W2);
170         for j=1:INN_MAX-1
171             W2(:,j+1) = A*W2(:,j);
172             for l = 1:j
173                 alpha = W2(:,j+1)'*W2(:,l);
174                 W2(:,j+1) = W2(:,j+1) - alpha * W2(:,l);
175             end
176             W2(:,j+1) = W2(:,j+1) / norm(W2(:,j+1));
177         end
178
179         [X, S, S] = svd( W1'*W2 ); % This is an equivalent way to perform
180                                    % the  optimal truncation since we have
181                                    % an orthonormal basis to W2
182
183         W1 = W1*X(:,TRC_ADD+1:DIM_W1);
184         V1 = V1*X(:,TRC_ADD+1:DIM_W1);
185
186         B = []; % Prevents problems with size without complicating the code
187       DIM_W1 = OUT_DIM - 1;
188     end
189
```

```
190         % Update V1 and W1
191     alpha = norm(new_w);
192     V1(:,DIM_W1+1) = new_v / alpha;
193     W1(:,DIM_W1+1) = new_w / alpha;
194
195      fprintf(1,'\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b %.10e',RES_HST(i+1));
196
197      DIM_W1 = DIM_W1 + 1;
198      i = i + 1;
199 end
200 %-------------------------------------------------------------------------
201
202 if ( i > OUT_MAX )
203      error('Algorithm failed to converge: number of maximum iterations
204            exceeded');
205 else
206      fprintf('\n\nAlgorithm has converged.\nNumber of Iterations:  %i',i-1);
207      fprintf('\nNorm of True Residual:  %e\n',RES_HST(i));
208 end
209
210 RES_HST(i+1) = norm(b - A*x);  % True Residual
211
212 R = struct('PROBLEM',PROBLEM,'INN_MAX',INN_MAX,'OUT_DIM',OUT_DIM,'OUT_MAX',...
213      OUT_MAX,'DIMENSION',dim,'TRC_ADD',TRC_ADD,'DENSITY',density,...
214      'TOLERANCE',TOL,'ZERO_INIT_GUESS',ZERO_INIT_GUESS,'NUM_ITERATIONS',i-1,...
215      'TRUE_RESIDUAL_NORM',RES_HST(i+1));
216
217 end
```

Apêndice D

LGMES - Protótipo em MATLAB

```
1  function [R RES_HST] = LGMRES(PROBLEM,INN_MAX,AUG_DIM, OUT_MAX, TOL)
2  %-----------------------------------------------------------------------
3  % LGMRES (Loose GMRES) is method that attempts to accelerate restarted GMRES
4  %   convergence by adding approximation to the error in the approximation
5  %   subspace each iteration.
6  %
7  %   A. H. Baker, E. R. Jessup and T. Manteuffel, "A Technique for
8  %   Accelerating the Convergence of Restarted GMRES", SIAM J. Matrix
9  %   Anal. Appl, 26 (2005), No 4, pp. 962-984
10 %-----------------------------------------------------------------------
11 %
12 % [R RES_HST] = LGMRES(PROBLEM,INN_MAX,AUG_DIM,OUT_MAX,TOL)
13 %
14 % R is a structure with some informations about the algorithm and the
15 % problem solved
16 %
17 % RES_HST is a vector that provides a history with the norm of residuals of
18 % all iterations
19 %
20 % PROBLEM is the path to the .m file containing the matrix A and the right
21 % hand side b. It may also contain the initial guess x; if it doesn't then
22 % the zero vector is used as initial guess. PROBLEM is a mandatory
23 % parameter
24 %
```

```
25 % INN_MAX is the maximum number of inner iterations. If not given, then the
26 % default number of inner iterations (which is 10) is used.
27 %
28 % AUG_DIM dimension of the space to augment the approximation subspace. If
29 % not given, then the default dimension (which is 3) is used.
30 %
31 % OUT_MAX is the maximum number of outer iterations. If not given, then the
32 % default number (that is the dimension of the matrix A) is used.
33 %
34 % TOL is the tolerance for convergence to be used. If not given, then the
35 % value given by the PROBLEM file is used or the default value (1e-10)
36 %-------------------------------------------------------------------------
37 %
38
39
40 %-------------------------------------------------------------------------
41 % Parameters Check
42 %-------------------------------------------------------------------------
43 if    exist('PROBLEM')
44     error('The PROBLEM parameter is mandatory. Exiting...');
45 else
46     if exist(PROBLEM,'file') || exist( strcat(PROBLEM,'.mat'),'file' )
47         load(PROBLEM);
48         % TODO: check if the loaded file contains the expected structure
49     else
50         error('File "%s" not found. Exiting...',PROBLEM);
51     end
52 end
53
54 if    exist('INN_MAX') || isempty(INN_MAX)
55     INN_MAX = 10;
56 end
57
```

```
58 if    exist('AUG_DIM') || isempty(AUG_DIM)
59    AUG_DIM = 3;
60 end
61
62 if    exist('OUT_MAX') || isempty(OUT_MAX)
63    OUT_MAX = dim;
64 end
65
66 if    exist('x')
67    x = zeros(dim,1);
68    ZERO_INIT_GUESS = 'True';
69 else
70    ZERO_INIT_GUESS = 'False';
71 end
72
73 if    exist('TOL') || isempty(TOL)
74    if    exist('tol')
75       TOL = tol;
76    else
77       TOL = 1e-10;
78    end
79 end
80 %-------------------------------------------------------------------------
81
82
83 %-------------------------------------------------------------------------
84 % Initialization...
85 %-------------------------------------------------------------------------
86 i = 1;
87 Z =[];
88 s = INN_MAX + AUG_DIM;
89
90 zlen = 0;      % Holds the length of Z matrix. Not really necessary, but
```

```
91              % makes the code clearer
92
93 r1 = b - A*x;
94 RES_HST(1) = norm(r1);
95
96 fprintf(1,'\nInitializing LGMRES method\nParameters %i,%i',INN_MAX,AUG_DIM);
97 fprintf(1,', %e)\nProblem:%s\nDimension:%i',TOL,PROBLEM, dim);
98 fprintf(1,'\nNorm of residual:  %.10e',RES_HST(1));
99
100 %-------------------------------------------------------------------------
101
102
103 %-------------------------------------------------------------------------
104 % Main Loop
105 %   I will store Z backwards, because the algorithm always uses the
106 %   augmented vectors from the most recent to the oldest. It makes some
107 %   indices look different from what was proposed by Baker et. al, but they
108 %   are equivalent
109 %-------------------------------------------------------------------------
110 while (RES_HST(i)/RES_HST(1) > TOL) && (i <= OUT_MAX)
111
112    V = r1 / RES_HST(i);
113    %----------------------------------------------------------------------
114    % Inner orthogonalization
115    %----------------------------------------------------------------------
116    for j = 1:s
117
118       if j <= s - zlen
119          u = A*V(:,j);
120       else
121          aux = j -s + zlen;
122          u = A*Z(:,aux);
123       end
```

```
124
125         for l=1:j
126             H(l,j) = V(:,l)'*u;
127             u = u - H(l,j)*V(:,l);
128         end
129
130         H(j+1,j) = norm(u);
131         V(:,j+1) = u / H(j+1,j);
132
133     end
134     %-------------------------------------------------------------------------
135
136     W = [V(:,1:s-zlen) Z];
137     y = H \ (RES_HST(:,i) * eye(j+1,1));
138
139     % Adding new z vector
140     if zlen >= AUG_DIM
141         Z = [ W*y Z(:,1:AUG_DIM-1) ];
142         zlen = AUG_DIM;
143     else
144         Z = [ W*y Z];
145         zlen = zlen +1;
146     end
147
148     x = x + Z(:,1);
149     Z(:,1) = Z(:,1) / norm(Z(:,1));
150
151     r1 = b - A*x;
152     RES_HST(i+1) = norm(r1);
153     fprintf(1,'\b\b\b\b\b\b\b\b\b\b\b\b\b\b\b%.10e',RES_HST(i+1));
154
155     i = i+1;
156 end
```

```
157 %-----------------------------------------------------------------------
158
159 if ( i > OUT_MAX )
160     error('Algorithm failed to converge: number of maximum iterations exceeded');
161 else
162     fprintf('\n\nAlgorithm has converged.\nNumber of Iterations %i',i-1);
163     fprintf('\nNorm of True Residual %.10e\n',RES_HST(i));
164 end
165
166 R = struct('PROBLEM',PROBLEM,'INN_MAX',INN_MAX,'AUG_DIM',AUG_DIM,'OUT_MAX',...
167     OUT_MAX,'DIMENSION',dim,'DENSITY',density,'TOLERANCE',TOL,...
168     'ZERO_INIT_GUESS',ZERO_INIT_GUESS,'NUM_ITERATIONS',i-1,...
169     'TRUE_RESIDUAL_NORM',RES_HST(i));
170
171 end
```

Apêndice E

GCRO - Protótipo em MATLAB (Versão Econômica)

```
1  function [R RES] = GCRO_lite(problem,m)
2  format long; load(problem);
3
4
5  MAX_ITER = length(A); sol = init_guess; k = 0;
6  d = [];C = [];U = [];Z = []; N = [];
7
8  tol = 1e-10;
9  res = b - A * init_guess;
10 norm_res = norm(res);
11 RES = norm_res;
12
13 while  (norm_res/RES(1) > tol && k <= MAX_ITER)
14 %     fprintf(1,'\nITERATION %d\n', k);
15    H = []; B =zeros(k,m);
16    V = res / norm_res;
17    for j = 1:m
18       V(:,j+1) = A * V(:,j);
19       for i = 1:k
20          B(i,j) = (N(i,i)^2 * C(:,i)' ) * V(:,j+1);
21          V(:,j+1) = V(:,j+1) - B(i,j) * (C(:,i));
22       end
```

```
23        %Orthogonalize against other v_j
24        for i = 1:j
25            H(i,j) = V(:,i)' * V(:,j+1);
26            V(:,j+1) = V(:,j+1) - H(i,j) * V(:,i);
27        end
28        H(j+1,j) = norm( V(:,j+1) );
29        V(:,j+1) = V(:,j+1) / H(j+1,j);
30    end
31    y = H \ (norm_res * eye(j+1,1));
32    U(:,k+1) = V(:,1:m) * y;
33    C(:,k+1) = V * H * y;
34    Z(1:(k+1),k+1) = [ B*y ; 1 ];
35    N(k+1,k+1) = inv( norm( C(:,k+1) ) );
36    d(k+1,1) = ( C(:,k+1)' * res ) * N(k+1,k+1)^2;
37    res = res - d(k+1) * C(:,k+1);
38    norm_res = norm(res);
39     RES(k+1,1) = norm_res; %%%%%%%%%%
40
41     fprintf(1,'Iteration %d\tRes: %f\n', k,norm_res);
42     k = k + 1;
43 end
44
45 if k>= MAX_ITER
46     k = inf;
47 end
48
49 d(k,1) = 1;
50 sol = sol + U * inv( Z ) * d;
51 true_res = b - A*sol;
52 R = [k ; norm(true_res)];
53 end
```

Apêndice F

GCR - Cabeçalho em C (PETSc)

```
1  /*
2     Private data structure used by the GCR method.
3     The notation used here for the variables ressemble the one used
4     by de Sturler in "Nested Krylov Methods based on GCR", 1996
5  */
6
7  #include "private/kspimpl.h"
8
9  typedef struct{
10    /* Varibles Description
11    orthog_method     Points to the orthogonalization method to be used
12                      (namely Gram-Schmidt and Modified Gram-Schmidt)
13    delta_allocate    How many vectors must be allocated in each
14                      GCRGetNewVectors function call for U and C array
15                      of Vec
16    allocated      How many vectors are already allocated in C and U
17                      arrays of Vec (which should be equal)
18    it             Iteration counter. The main difference between this
19                      and ksp->it its hat the former is supposed to reset
20                      with restarts, and the later not
21    work_chunk        ------------
22    wc_allocated      The variable wc_allocated[i] holds how many
23                            vectors are allocated in work_chunk[i].
24                            Necessary to deallocate those vectors.
```

```
25    i_work_chunk      Stores the current (first) index of
26                            work_chunk bidimensional array
27    restart_point     When the number of iterations (ksp->its) is a
28                            multiple of this, then we restart
29                            (not implemented). Set 0 for no restart.
30    orthowork         ----------
31    */
32
33    Vec    *C, *U, res;
34    Vec    **work_chunk;
35
36    PetscInt    *wc_allocated;
37    PetscScalar *orthogwork;
38
39    PetscErrorCode (*orthog_method)(KSP);
40
41    PetscInt    i_work_chunk;
42    PetscInt    delta_allocate;
43    PetscInt    allocated;
44    PetscInt    it;
45    PetscInt    restart_point;
46
47 } KSP_GCR;
48
49 #define VEC_CK       gcr->C[gcr->it]
50 #define VEC_UK       gcr->U[gcr->it]
51 #define WORK_CHUNK_I gcr->work_chunk[gcr->i_work_chunk]
```

Apêndice G

GCR - Código Fonte em C (PETSc)

```
1  /*------------------------------------------------------------------------
2  This file implements GCR (Generalized Minimal Residual) method
3  Reference: Eisenstat, Elman and Schultz, 1983
4  ------------------------------------------------------------------------*/
5  #include "/Lago/Projetos/PETSc/gcr_p.h"
6
7  #define PETSCKSP_DLL
8  #define GCR_DELTA_ALLOCATE 10
9  #define GCR_DEFAULT_RESTART_POINT 30 /* 0 means no restart" */
10
11 static PetscErrorCode   GCRGetNewVectors(KSP);
12 PetscErrorCode PETSCKSP_DLLEXPORT KSPDestroy_GCR_Internal(KSP);
13 PetscErrorCode PETSCKSP_DLLEXPORT KSPGCRSetRestart_GCR(KSP, PetscInt);
14 PetscErrorCode PETSCKSP_DLLEXPORT KSPGCRSetRestart(KSP, PetscInt);
15
16 /*------------------------------------------------------------------------
17 KSPGCRModifiedGramSchmidtOrthogonalization
18 ------------------------------------------------------------------------*/
19 #undef __FUNCT__
20 #define __FUNCT__ "KSPGCRModifiedGramSchmidtOrthogonalization"
21 PetscErrorCode PETSCKSP_DLLEXPORT
22 KSPGCRModifiedGramSchmidtOrthogonalization(KSP
```

```
23  ksp){
24    KSP_GCR         *gcr = (KSP_GCR *)(ksp->data);
25    PetscErrorCode ierr;
26    PetscInt        j;
27    PetscScalar     alpha;
28
29    PetscFunctionBegin;
30    ierr = PetscLogEventBegin(KSP_GCROrthogonalization,ksp,0,0,0);CHKERRQ(
31           ierr);
32
33    for (j=0; j < gcr->it;j++){
34       /* alpha = cj'ck */     ierr = VecDot(VEC_CK, gcr->C[j], &alpha);
35       /* ck = ck - alpha.cj*/ ierr = VecAXPY(VEC_CK, -alpha, gcr->C[j]);
36       /* uk = uk - alpha.uj*/ ierr = VecAXPY(VEC_UK, -alpha, gcr->U[j]);
37    }
38
39    ierr = PetscLogEventEnd(KSP_GCROrthogonalization,ksp,0,0,0);CHKERRQ(ierr)
40           ;
41    PetscFunctionReturn(0);
42 }
43
44
45 /*------------------------------------------------------------------------
46 KSPGCRClassicalGramSchmidtOrthogonalization
47 ------------------------------------------------------------------------*/
48 #undef __FUNCT__
49 #define __FUNCT__ "KSPGCRClassicalGramSchmidtOrthogonalization"
50 PetscErrorCode PETSCKSP_DLLEXPORT
51 KSPGCRClassicalGramSchmidtOrthogonalization(KSP  ksp){
52    KSP_GCR         *gcr = (KSP_GCR *)(ksp->data);
53    PetscErrorCode ierr;
54    PetscInt        j;
55
```

```
56    PetscFunctionBegin;
57    ierr = PetscLogEventBegin(KSP_GCROrthogonalization,ksp,0,0,0);CHKERRQ(
58           ierr);
59
60    /* orthogwork = C'ck */       ierr = VecMDot( VEC_CK, gcr->it+1 ,gcr->C,
61 gcr->orthogwork );CHKERRQ(ierr);
62    /* Changes signal*/           for (j=0; j<=gcr->it; j++) gcr->orthogwork[
63                                        j] =
64 -gcr->orthogwork[j];
65
66    /* ck = ck - C.orthogwork */  ierr = VecMAXPY(VEC_CK, gcr->it+1,
67 gcr->orthogwork , gcr->C);CHKERRQ(ierr);
68    /* uk = uk - U.orthogwork */  ierr = VecMAXPY(VEC_UK, gcr->it+1,
69 gcr->orthogwork , gcr->U);CHKERRQ(ierr);
70
71    ierr = PetscLogEventEnd(KSP_GCROrthogonalization,ksp,0,0,0);CHKERRQ(ierr)
72           ;
73    PetscFunctionReturn(0);
74 }
75
76
77 /*------------------------------------------------------------------------
78 GCRGetNewVectors
79 Allocates new vectors if needed.
80 ------------------------------------------------------------------------*/
81 #undef __FUNCT__
82 #define __FUNCT__ "GCRGetNewVectors"
83 static PetscErrorCode GCRGetNewVectors(KSP ksp) {
84
85    PetscErrorCode ierr;
86    KSP_GCR   *gcr = (KSP_GCR *)(ksp->data);
87    PetscInt  n_alloc,j; /* Number of vectors to be allocated in this call*/
88
```

```
89    PetscFunctionBegin;
90    n_alloc = PetscMin(gcr->delta_allocate,gcr->restart_point - gcr->
91              allocated);
92
93    if (!n_alloc) PetscFunctionReturn(0);
94
95    ierr = KSPGetVecs(ksp, 2*n_alloc , &WORK_CHUNK_I, 0, PETSC_NULL);
96 CHKERRQ(ierr);
97    ierr = PetscLogObjectParents(ksp, 2*n_alloc, WORK_CHUNK_I); CHKERRQ(ierr)
98           ;
99
100    /* Append the new vectors to C and U. C receives the first n_alloc Vecs
101       while U receive the n_alloc last ones */
102    for (j=0; j < n_alloc; j++){
103       gcr->C[gcr->it +j] = WORK_CHUNK_I[j];
104       gcr->U[gcr->it +j] = WORK_CHUNK_I[n_alloc+j];
105    }
106
107    /* Update counters */
108    gcr->wc_allocated[gcr->i_work_chunk] = 2*n_alloc;
109    gcr->allocated += n_alloc;
110    gcr->i_work_chunk++;
111
112    PetscFunctionReturn(0);
113 }
114
115 /*------------------------------------------------------------------------
116 KSPSetUp_GCR
117 ------------------------------------------------------------------------*/
118 #undef __FUNCT__
119 #define __FUNCT__ "KSPSetUp_GCR"
120 PetscErrorCode KSPSetUp_GCR(KSP ksp) {
121
```

```
122     PetscErrorCode ierr;
123     KSP_GCR      *gcr = (KSP_GCR *)ksp->data;
124     Mat          Amat;
125     PetscInt     m,n,wc_alloc;
126
127     PetscFunctionBegin;
128     ierr = PCGetOperators(ksp->pc,&Amat,PETSC_NULL,PETSC_NULL); CHKERRQ(ierr)
129            ;
130     ierr =  MatGetSize(Amat, &m, &n);
131     if (m!=n) SETERRQ2(PETSC_ERR_ARG_SIZ, "This GCR implementation is not
132         intended
133 to solve rectangular linear systems (%d,%d)\n",m,n);CHKERRQ(ierr);
134
135     /* if no restart is set */
136     if (!gcr->restart_point) gcr->restart_point = m;
137
138     /* We will need to alloc things in work_chunk at most
139     gcr->restart_point /delta_allocate times. The + 0.5 is just for round
140     up purposes */
141     wc_alloc = (PetscInt) (((float) gcr->restart_point) / ((float)
142 gcr->delta_allocate) + 0.5);
143
144     /* This just "defines the size of the array", do not allocate anything
145     (except for gcr->wc_allocated) */
146     ierr = PetscMalloc((gcr->restart_point)*sizeof(void*),&gcr->C); CHKERRQ(
147            ierr);
148     ierr = PetscMalloc((gcr->restart_point)*sizeof(void*),&gcr->U); CHKERRQ(
149            ierr);
150     ierr = PetscMalloc((wc_alloc)*sizeof(void*),&gcr->work_chunk); CHKERRQ(
151            ierr);
152     ierr = PetscMalloc((wc_alloc)*sizeof(PetscInt),&gcr->wc_allocated);
153 CHKERRQ(ierr);
154     ierr = PetscLogObjectMemory(ksp, (2*gcr->restart_point +
```

```
155 wc_alloc)*sizeof(void*) + (wc_alloc)*sizeof(PetscInt) );CHKERRQ(ierr);
156
157    if (gcr->orthog_method == KSPGCRClassicalGramSchmidtOrthogonalization) {
158       ierr =
159 PetscMalloc((gcr->restart_point)*sizeof(PetscInt),&gcr->orthogwork);CHKERRQ(
160             ierr
161 );
162       ierr = PetscLogObjectMemory(ksp, gcr->restart_point*sizeof(PetscInt)
163 );CHKERRQ(ierr);
164    }
165
166    ierr = VecDuplicate(ksp->vec_rhs,&gcr->res);CHKERRQ(ierr);
167    ierr = PetscLogObjectParents(ksp,1,&gcr->res);CHKERRQ(ierr);
168
169    PetscFunctionReturn(0);
170 }
171
172 /*------------------------------------------------------------------------
173    GCRCycle
174    Perform one single GCR iteration
175 ------------------------------------------------------------------------*/
176 #undef __FUNCT__
177 #define __FUNCT__ "GCRCycle"
178 PetscErrorCode GCRCycle(KSP ksp) {
179
180    PetscErrorCode ierr;
181    PetscReal   alpha;
182    Mat         Amat;
183    KSP_GCR     *gcr = (KSP_GCR *)(ksp->data);
184
185    PetscFunctionBegin;
186
187    /* u = r */    ierr = VecCopy(gcr->res,VEC_UK); CHKERRQ(ierr);
```

```
188                     ierr = PCGetOperators(ksp->pc,&Amat,PETSC_NULL,PETSC_NULL)
189                            ;
190 CHKERRQ(ierr);
191     /* c = Au */   ierr = MatMult(Amat, VEC_UK, VEC_CK); CHKERRQ(ierr);
192
193     /* Ortogonalize C and updates U */
194     ierr = (*gcr->orthog_method)(ksp); CHKERRQ(ierr);
195     /* c = c / ||c|| */     ierr = VecNormalize( VEC_CK, &alpha ); CHKERRQ(
196             ierr);
197     /* u = u / ||c|| */     ierr = VecScale( VEC_UK, ((PetscScalar) 1/alpha)
198 );CHKERRQ(ierr);
199     /* alpha = c'.r */      ierr = VecDot( VEC_CK, gcr->res, &alpha );
200 CHKERRQ(ierr);
201     /* x = x + alpha.u */   ierr = VecAXPY( ksp->vec_sol , alpha , VEC_UK);
202 CHKERRQ(ierr);
203     /* r = r - alpha.c */   ierr = VecAXPY( gcr->res , -alpha , VEC_CK);
204 CHKERRQ(ierr);
205
206     PetscFunctionReturn(0);
207 }
208
209 /*-------------------------------------------------------------------------
210     KSPSolve_GCR
211     - Calculates initial residual
212     - Checks whether its needed to allocate more vectors
213     - Invoke the GCR iterations (GCRCycle function)
214     - Calculate and check the residual
215 -------------------------------------------------------------------------*/
216 #undef __FUNCT__
217 #define __FUNCT__ "KSPSolve_GCR"
218 PetscErrorCode KSPSolve_GCR(KSP ksp) {
219
220     PetscErrorCode ierr;
```

```
221   KSP_GCR      *gcr = (KSP_GCR *)ksp->data;
222
223   PetscFunctionBegin;
224
225   if (ksp->guess_zero){
226      ierr = VecCopy(ksp->vec_rhs,gcr->res); CHKERRQ(ierr);
227   }
228   //else... calc res
229
230   ksp->reason = KSP_CONVERGED_ITERATING;
231   ksp->its = 0;
232   gcr->it = 0;
233
234   while ( (!ksp->reason) && (ksp->its <= ksp->max_it) ) {
235
236      KSPLogResidualHistory(ksp,ksp->rnorm);
237      KSPMonitor(ksp,ksp->its,ksp->rnorm);
238
239      /* Allocate new vectors if necessary */
240      if (gcr->allocated <= gcr->it)
241         GCRGetNewVectors(ksp);
242
243      /* Execute one single iteration of GCR Method */
244      ierr     = GCRCycle(ksp);CHKERRQ(ierr);
245
246      /* Convergence Criteria */
247      ierr     = VecNorm(gcr->res, NORM_2, &ksp->rnorm); CHKERRQ(ierr);
248      if (ksp->rnorm < ksp->abstol) ksp->reason = KSP_CONVERGED_ATOL;
249
250      gcr->it++;      /* This counter resets with restart... */
251      ksp->its++;     /* ... but this one does not */
252      if (gcr->it >= gcr->restart_point) gcr->it = 0;
253   }
```

```
254
255     gcr->it--;
256     ksp->its--;
257     if (ksp->its > ksp->max_it) ksp->reason = KSP_DIVERGED_ITS;
258
259     PetscFunctionReturn(0);
260 }
261
262 /*------------------------------------------------------------------------
263     KSPSetFromOptions_GCR
264 ------------------------------------------------------------------------*/
265 #undef __FUNCT__
266 #define __FUNCT__ "KSPSetFromOptions_GCR"
267 PetscErrorCode KSPSetFromOptions_GCR(KSP ksp){
268     PetscErrorCode ierr;
269     PetscInt       new_restart_point;
270     KSP_GCR        *gcr = (KSP_GCR*)ksp->data;
271     PetscTruth     flg;
272
273     PetscFunctionBegin;
274     ierr = PetscOptionsHead("KSP GCR Options");CHKERRQ(ierr);
275     ierr = PetscOptionsInt("-ksp_gcr_restart", "Number of Krylov search
276 directions", "KSPGCRSetRestart", gcr->restart_point, &new_restart_point,
277 &flg);
278 CHKERRQ(ierr);
279     if (flg) { ierr = KSPGCRSetRestart(ksp,new_restart_point);CHKERRQ(ierr);
280         }
281
282     ierr = PetscOptionsTail(); CHKERRQ(ierr);
283     PetscFunctionReturn(0);
284 }
285
286 /*------------------------------------------------------------------------
```

```
287    KSPDestroy_GCR
288 ------------------------------------------------------------------------*/
289 #undef __FUNCT__
290 #define __FUNCT__ "KSPDestroy_GCR"
291 PetscErrorCode KSPDestroy_GCR(KSP ksp) {
292   PetscErrorCode ierr;
293
294   PetscFunctionBegin;
295   ierr = KSPDestroy_GCR_Internal(ksp);CHKERRQ(ierr);
296   ierr = PetscFree(ksp->data);CHKERRQ(ierr);
297
298 /* clear composed functions */
299   ierr =
300 PetscObjectComposeFunctionDynamic((PetscObject)ksp,"KSPGCRSetRestart_GCR",
301                                     "",
302 PETSC_NULL); CHKERRQ(ierr);
303
304 PetscFunctionReturn(0);
305 }
306
307 /*------------------------------------------------------------------------
308    KSPDestroy_GCR_Internal
309 ------------------------------------------------------------------------*/
310 #undef __FUNCT__
311 #define __FUNCT__ "KSPDestroy_GCR_Internal"
312 PetscErrorCode PETSCKSP_DLLEXPORT KSPDestroy_GCR_Internal(KSP ksp) {
313
314    KSP_GCR      *gcr = (KSP_GCR*)ksp->data;
315    PetscErrorCode ierr;
316    PetscInt       i;
317
318    PetscFunctionBegin;
319
```

```
320     ierr = PetscFree(gcr->C);CHKERRQ(ierr);
321     ierr = PetscFree(gcr->U);CHKERRQ(ierr);
322     ierr = VecDestroy(gcr->res);CHKERRQ(ierr);
323
324     if (gcr->orthog_method == KSPGCRClassicalGramSchmidtOrthogonalization) {
325       ierr = PetscFree(gcr->orthogwork);CHKERRQ(ierr);
326     }
327
328     for (i=0; i< gcr->i_work_chunk ; i++) {
329       ierr =
330 VecDestroyVecs(gcr->work_chunk[i],gcr->wc_allocated[i]);CHKERRQ(ierr);
331     }
332
333     ierr = PetscFree(gcr->wc_allocated);CHKERRQ(ierr);
334
335     gcr->allocated = 0;
336     gcr->i_work_chunk = 0;
337
338     PetscFunctionReturn(0);
339 }
340
341 /*-------------------------------------------------------------------------
342     KSPView_GCR
343  -------------------------------------------------------------------------*/
344 #undef __FUNCT__
345 #define __FUNCT__ "KSPView_GCR"
346 PetscErrorCode KSPView_GCR(KSP ksp,PetscViewer viewer) {
347     KSP_GCR         *gcr = (KSP_GCR *)ksp->data;
348     const char         *cstr;
349     PetscErrorCode     ierr;
350     PetscTruth         iascii,isstring;
351
352     PetscFunctionBegin;
```

```
353    ierr =
354 PetscTypeCompare((PetscObject)viewer,PETSC_VIEWER_ASCII,&iascii);CHKERRQ(
355                  ierr);
356    ierr =
357 PetscTypeCompare((PetscObject)viewer,PETSC_VIEWER_STRING,&isstring);CHKERRQ(
358                  ierr
359 );
360    if (gcr->orthog_method == KSPGCRClassicalGramSchmidtOrthogonalization) {
361       cstr = "Classical (unmodified) Gram-Schmidt Orthogonalization with no
362 iterative refinement";
363    } else if (gcr->orthog_method ==
364               KSPGCRModifiedGramSchmidtOrthogonalization) {
365       cstr = "Modified Gram-Schmidt Orthogonalization";
366    } else {
367       cstr = "Unknown orthogonalization";
368    }
369
370    if (iascii) {
371       ierr = PetscViewerASCIIPrintf(viewer,"  GCR: restart=%D, using
372 %s\n",gcr->restart_point,cstr);CHKERRQ(ierr);
373       ierr = PetscViewerASCIIPrintf(viewer,"  GCR: absolute tolerance
374 %G\n",ksp->abstol);CHKERRQ(ierr);
375    } else if (isstring) {
376       ierr = PetscViewerStringSPrintf(viewer,"%s restart
377 %D",cstr,gcr->restart_point);CHKERRQ(ierr);
378    } else {
379       SETERRQ1(PETSC_ERR_SUP,"Viewer type %s not supported for KSP
380 GCR",((PetscObject)viewer)->type_name);
381    }
382    PetscFunctionReturn(0);
383 }
384
385 /*-------------------------------------------------------------------------
```

```
386    KSPGCRSetRestart
387 -----------------------------------------------------------------------*/
388 #undef __FUNCT__
389 #define __FUNCT__ "KSPGCRSetRestart"
390 PetscErrorCode PETSCKSP_DLLEXPORT KSPGCRSetRestart(KSP ksp, PetscInt
391                                                    restart) {
392   PetscErrorCode ierr;
393
394   PetscFunctionBegin;
395   ierr =
396 PetscTryMethod(ksp,"KSPGCRSetRestart_C",(KSP,PetscInt),(ksp,restart));
397                CHKERRQ(
398 ierr);
399   PetscFunctionReturn(0);
400 }
401
402 /*-----------------------------------------------------------------------
403    KSPGCRSetRestart_GCR
404 -----------------------------------------------------------------------*/
405 EXTERN_C_BEGIN
406 #undef __FUNCT__
407 #define __FUNCT__ "KSPGCRSetRestart_GCR"
408 PetscErrorCode PETSCKSP_DLLEXPORT KSPGCRSetRestart_GCR(KSP ksp, PetscInt
409 new_restart_point) {
410    KSP_GCR        *gcr = (KSP_GCR *) ksp->data;
411    PetscErrorCode ierr;
412
413    PetscFunctionBegin;
414    if (new_restart_point < 0) SETERRQ(PETSC_ERR_ARG_OUTOFRANGE,"Restart
415        must be
416 positive");
417
418    /* If the vectors were already created, destroy and create again */
```

```
419     if ( (ksp->setupcalled) && (gcr->restart_point != new_restart_point) ) {
420       ierr = KSPDestroy_GCR_Internal(ksp);CHKERRQ(ierr);
421       ksp->setupcalled = 0;
422     }
423   gcr->restart_point = new_restart_point;
424   PetscFunctionReturn(0);
425 }
426 EXTERN_C_END
427
428 /*------------------------------------------------------------------------
429    KSPCreate_GCR
430 ------------------------------------------------------------------------*/
431 EXTERN_C_BEGIN
432 #undef __FUNCT__
433 #define __FUNCT__ "KSPCreate_GCR"
434 PetscErrorCode PETSCKSP_DLLEXPORT KSPCreate_GCR(KSP ksp) {
435    KSP_GCR  *gcr;
436    PetscErrorCode ierr;
437
438    PetscFunctionBegin;
439    ierr = PetscNewLog(ksp,KSP_GCR,&gcr); CHKERRQ(ierr);
440    ksp->data                = (void *)gcr;
441
442    ksp->normtype            = KSP_NORM_PRECONDITIONED;
443    ksp->pc_side             = PC_LEFT;
444
445    ksp->abstol              = 1.0e-7;   /* Absolute convergence */
446    ksp->guess_zero          = PETSC_TRUE;
447
448    ksp->ops->setup          = KSPSetUp_GCR;
449    ksp->ops->solve          = KSPSolve_GCR;
450    ksp->ops->destroy        = KSPDestroy_GCR;
451    ksp->ops->view           = KSPView_GCR;
```

```
452    ksp->ops->setfromoptions = KSPSetFromOptions_GCR;
453
454   ierr = PetscObjectComposeFunctionDynamic((PetscObject)ksp,
455          "KSPGCRSetRestart_C",
456          "KSPGCRSetRestart_GCR",
457          KSPGCRSetRestart_GCR);CHKERRQ(ierr);
458
459    gcr->orthog_method      =  KSPGCRModifiedGramSchmidtOrthogonalization;
460    gcr->allocated          = 0;
461    gcr->i_work_chunk       = 0;
462    gcr->delta_allocate     = GCR_DELTA_ALLOCATE;
463    gcr->restart_point      = GCR_DEFAULT_RESTART_POINT;
464
465    PetscFunctionReturn(0);
466 }
467 EXTERN_C_END
```

Apêndice H

GCR - Exemplo (PETSc)

```
1
2  static char help[] = "Reads a linear system from file and then solves it";
3  #include "petscksp.h"
4
5  #undef __FUNCT__
6  #define __FUNCT__ "main"
7  int main(int argc,char **args)
8  {
9     KSP            ksp;
10    PetscMPIInt    rank=1;
11    PetscViewer    fd;      /* File descriptor */
12    PetscInt       m,n;
13    PetscReal      norm;
14    PetscTruth     flg;
15    PetscErrorCode ierr;
16    char           file[PETSC_MAX_PATH_LEN];
17
18    Mat            A,I;
19    Vec            b,x,exact_sol;
20
21    PetscInitialize(&argc,&args,(char *)0,help);
22    ierr = MPI_Comm_rank(PETSC_COMM_WORLD,&rank);CHKERRQ(ierr);
23
24    /* Verifies if the file name argument is provided */
```

```
25    ierr = PetscOptionsGetString(PETSC_NULL,"-f",file,PETSC_MAX_PATH_LEN-1,
26           &flg);CHKERRQ(ierr);
27    if (!flg) SETERRQ(1,"Must indicate binary file with the -f option");
28
29    /* Read from file */
30    ierr = PetscViewerBinaryOpen(PETSC_COMM_WORLD, file, FILE_MODE_READ, &fd)
31           ;CHKERRQ(ierr);
32    ierr = MatLoad(fd,MATAIJ,&A);CHKERRQ(ierr);
33
34    ierr = MatGetLocalSize(A,&m,&n); CHKERRQ(ierr);
35    if (m != n) SETERRQ2(PETSC_ERR_ARG_SIZ, "This example is not intended
36        for rectangular matrices (%d, %d)", m, n);
37
38    /* - Creates all vectors
39       - exact_sol all one
40       - Set right hand side */
41    ierr = VecCreate(PETSC_COMM_WORLD,&exact_sol); CHKERRQ(ierr);
42    ierr = VecSetSizes(exact_sol,m,PETSC_DECIDE); CHKERRQ(ierr);
43    ierr = VecSetFromOptions(exact_sol); CHKERRQ(ierr);
44    ierr = VecDuplicate(exact_sol,&b);CHKERRQ(ierr);
45    ierr = VecDuplicate(exact_sol,&x);CHKERRQ(ierr);
46    ierr = VecSet(exact_sol, 1.0); CHKERRQ(ierr);
47    ierr = MatMult(A,exact_sol,b);CHKERRQ(ierr);
48
49    /* Created the Identity Matrix for preconditioning */
50    ierr = MatDuplicate(A, MAT_DO_NOT_COPY_VALUES, &I); CHKERRQ(ierr);
51    ierr = MatDiagonalSet(I, exact_sol, INSERT_VALUES); CHKERRQ(ierr)
52
53    /* Creates and set KSP context, then solve the system */
54    ierr = KSPCreate(PETSC_COMM_WORLD,&ksp);CHKERRQ(ierr);
55    ierr = KSPSetOperators(ksp,A,I,SAME_PRECONDITIONER);CHKERRQ(ierr);
56    ierr = KSPSetTolerances(ksp,1.e-7,PETSC_DEFAULT,PETSC_DEFAULT,
57           PETSC_DEFAULT);CHKERRQ(ierr);
```

```
58    ierr = KSPSetFromOptions(ksp);CHKERRQ(ierr);
59    ierr = KSPSolve(ksp,b,x);CHKERRQ(ierr);
60
61    ierr = KSPView(ksp,PETSC_VIEWER_STDOUT_WORLD);CHKERRQ(ierr);
62
63    /* Check the error */
64    ierr = VecAXPY(x,-1.0,exact_sol); CHKERRQ(ierr);
65    ierr = VecNorm(x,NORM_2,&norm); CHKERRQ(ierr);
66    ierr = KSPGetIterationNumber(ksp,&n); CHKERRQ(ierr);
67    ierr = PetscPrintf(PETSC_COMM_WORLD, "Norm of error %g, Iterations %D\n",
68           norm, n); CHKERRQ(ierr);
69
70    /* Free work space.  All PETSc objects should be destroyed when they
71       are no longer needed. */
72    ierr = VecDestroy(x);CHKERRQ(ierr); ierr = VecDestroy(exact_sol);CHKERRQ(
73           ierr);
74    ierr = VecDestroy(b);CHKERRQ(ierr); ierr = MatDestroy(A);CHKERRQ(ierr);
75    ierr = MatDestroy(I);CHKERRQ(ierr); ierr = KSPDestroy(ksp);CHKERRQ(ierr);
76
77    ierr = PetscFinalize();CHKERRQ(ierr);
78    return 0;
79 }
```

Apêndice I

GCR - Saída (PETSc)

```
  1 KSP preconditioned resid norm 5.565500753978e+00 true resid norm
  5.565500753978e+00 ||Ae||/||Ax|| 3.521497198213e-01
  2 KSP preconditioned resid norm 3.938792642989e+00 true resid norm
  3.938792642989e+00 ||Ae||/||Ax|| 2.492219095778e-01
  3 KSP preconditioned resid norm 2.262355376865e+00 true resid norm
  2.262355376865e+00 ||Ae||/||Ax|| 1.431475526313e-01
  4 KSP preconditioned resid norm 1.475057028466e+00 true resid norm
  1.475057028466e+00 ||Ae||/||Ax|| 9.333228801085e-02
  5 KSP preconditioned resid norm 3.208603385589e-01 true resid norm
  3.208603385589e-01 ||Ae||/||Ax|| 2.030201473687e-02
  6 KSP preconditioned resid norm 6.979483157150e-02 true resid norm
  6.979483157150e-02 ||Ae||/||Ax|| 4.416175914686e-03
  7 KSP preconditioned resid norm 4.550627099213e-02 true resid norm
  4.550627099214e-02 ||Ae||/||Ax|| 2.879349278417e-03
  8 KSP preconditioned resid norm 2.613779556113e-02 true resid norm
  2.613779556113e-02 ||Ae||/||Ax|| 1.653834540768e-03
  9 KSP preconditioned resid norm 1.849813007150e-02 true resid norm
  1.849813007150e-02 |||Ae||/||Ax|| 1.170444782932e-03
 10 KSP preconditioned resid norm 6.514111321898e-03 true resid norm
6.514111321897e-03 ||Ae||/||Ax|| 4.121718023759e-04
 11 KSP preconditioned resid norm 2.293942476892e-03 true resid norm
2.293942476891e-03 ||Ae||/||Ax|| 1.451461847249e-04
 12 KSP preconditioned resid norm 1.623459262845e-03 true resid norm
1.623459262846e-03 ||Ae||/||Ax|| 1.027222436623e-04
 13 KSP preconditioned resid norm 9.324790932971e-04 true resid norm
9.324790932977e-04 ||Ae||/||Ax|| 5.900138477375e-05
 14 KSP preconditioned resid norm 6.079769140299e-04 true resid norm
6.079769140314e-04 ||Ae||/||Ax|| 3.846893736938e-05
 15 KSP preconditioned resid norm 1.322495840547e-04 true resid norm
```

```
 1.322495840578e-04 ||Ae||/||Ax|| 8.367918006148e-06
 16 KSP preconditioned resid norm 2.876746152533e-05 true resid norm
 2.876746152548e-05 ||Ae||/||Ax|| 1.820223186374e-06
 17 KSP preconditioned resid norm 1.875640173423e-05 true resid norm
 1.875640173570e-05 ||Ae||/||Ax|| 1.186786581847e-06
 18 KSP preconditioned resid norm 1.077326230656e-05 true resid norm
 1.077326230681e-05 ||Ae||/||Ax|| 6.816639635153e-07
 19 KSP preconditioned resid norm 7.624407612154e-06 true resid norm
 7.624407611938e-06 ||Ae||/||Ax|| 4.824243357489e-07
 20 KSP preconditioned resid norm 2.684933004424e-06 true resid norm
 2.684933006118e-06 ||Ae||/||Ax|| 1.698855947810e-07
 21 KSP preconditioned resid norm 9.454984052468e-07 true resid norm
 9.454984050117e-07 ||Ae||/||Ax|| 5.982516455118e-08
 22 KSP preconditioned resid norm 6.691441304510e-07 true resid norm
 6.691441303593e-07 ||Ae||/||Ax|| 4.233921230856e-08
 23 KSP preconditioned resid norm 3.843415885622e-07 true resid norm
 3.843415884478e-07 ||Ae||/||Ax|| 2.431870709763e-08
 24 KSP preconditioned resid norm 2.505909404587e-07 true resid norm
 2.505909400532e-07 ||Ae||/||Ax|| 1.585581122533e-08


KSP Object:
  type: gcr
    GCR: restart=5, using Modified Gram-Schmidt Orthogonalization
    GCR: absolute tolerance 1e-07
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-07, absolute=1e-07, divergence=10000
  left preconditioning
PC Object:
  type: ilu
    ILU: 0 levels of fill
    ILU: factor fill ratio allocated 1
    ILU: tolerance for zero pivot 1e-12
    ILU: using diagonal shift to prevent zero pivot
    ILU: using diagonal shift on blocks to prevent zero pivot
         out-of-place factorization
         matrix ordering: natural
    ILU: factor fill ratio needed 1
         Factored matrix follows
        Matrix Object:
          type=seqaij, rows=50, cols=50
          package used to perform factorization: petsc
```

```
          total: nonzeros=708, allocated nonzeros=708
            using I-node routines: found 25 nodes, limit used is 5
  linear system matrix followed by preconditioner matrix:
  Matrix Object:
    type=seqaij, rows=50, cols=50
    total: nonzeros=708, allocated nonzeros=708
      using I-node routines: found 25 nodes, limit used is 5
  Matrix Object:
    type=seqaij, rows=50, cols=50
    total: nonzeros=708, allocated nonzeros=708
      using I-node routines: found 25 nodes, limit used is 5
Norm of error 2.01392e-08, Iterations 25
```