UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Estatística

Arthur Mota Silva Dantés Macedo

Métodos de Krylov aplicados na Análise de Regressão Linear de Big Data

Belo Horizonte
2025

Arthur Mota Silva Dantés Macedo

Métodos de Krylov aplicados na Análise de Regressão Linear de Big Data

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Estatística da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Estatística.

Orientador: Thiago Rezende dos Santos

Belo Horizonte
2025

Macedo, Arthur Mota Silva Dantés.

M141m Métodos de Krylov aplicados na análise de regressão linear de Big Data [recurso eletrônico] / Arthur Mota Silva Dantés Macedo – 2025.
1 recurso online (81 f. il., color.) : pdf.

Orientador: Thiago Rezende dos Santos.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Estatística.
Referências: f. 63-66

1. Estatística – Teses. 2. Análise de regressão – Teses. 3. Álgebra linear - Teses. 4. Big data – Teses. 5. Mínimos quadrados – Processamento de dados – Teses. I. Santos, Thiago Rezende dos. I. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Estatística. III. Título.

CDU 519.2(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

FOLHA DE APROVAÇÃO

"Métodos de Krylov aplicados na Análise de Regressão Linear de Big Data"

ARTHUR MOTA SILVA DANTÉS MACEDO

Dissertação submetida à Banca Examinadora, designada pelo Colegiado do Programa de Pós-Graduação em ESTATÍSTICA, como requisito para obtenção do grau de Mestre em ESTATÍSTICA, área de concentração ESTATÍSTICA E PROBABILIDADE.

Aprovada em 07 de março de 2025, pela banca constituída pelos membros:

Prof. Thiago Rezende dos Santos - Orientador
DEST/UFMG

gov.br
Documento assinado digitalmente
CRISTIANO DE CARVALHO SANTOS
Data: 07/03/2025 12:51:35-0300
Verifique em https://validar.iti.gov.br

Prof. Cristiano de Carvalho Santos
DEST/UFMG

Prof. Gilvan Ramalho Guedes
DEMOGRAFIA/UFMG

Belo Horizonte, 7 de março de 2025.

Dedico este trabalho aos meu pais, Renato e Luciana.

Agradecimentos

Agradeço aos meus pais, Renato e Luciana, por terem viabilizado e incentivado minha educação, o que me permitiu estar na posição que estou hoje.

Aos meus amigos, à minha família e à minha parceira, Luisa, pelo apoio e incentivo que me deram por todo esse tempo.

Ao meu orientador Thiago, que, com muita paciência, conhecimento e experiência, me conduziu na elaboração deste trabalho.

À UFMG, por oferecer um dos melhores ensinos públicos de nível superior do Brasil e do mundo.

Às organizações de pesquisa e fomento que apoiam a educação: CAPES, CAPEMIG, CNPq e, em especial, a FAPEMIG, que durante meu mestrado disponibilizou bolsa por mais de um ano e que acreditou no meu potencial como pesquisador.

Resumo

O crescente uso de bases de dados enormes e complexas, chamadas de Big Data, torna necessária uma otimização dos métodos tradicionais de análise de dados para viabilizar sua aplicação nesse tipo de dados. Mesmo métodos de análise estatística considerados simples, como a regressão linear, são ineficientes quando aplicados na sua forma tradicional a Big Data, devido ao seu alto custo computacional. Por consequência, exigem adaptações. O presente trabalho trata da aplicação em dados considerados Big Data de uma classe de algoritmos, os Métodos de Krylov, a fim de se estimar os parâmetros da regressão linear eficientemente. Esses métodos retornam uma aproximação da solução do problema dos mínimos quadrados a cada iteração, sendo computacionalmente mais econômicos para se obter uma estimativa satisfatória da solução quando comparados a métodos tradicionais, como o método da decomposição QR aplicado ao problema dos mínimos quadrados. Dois métodos de Krylov são apresentados e estudados no texto: o Generalized Minimum Residuals (GMRES) e o LSMR, com um grande foco no último.

Por fim, para avaliar o desempenho do LSMR, são apresentados vários estudos de simulações em bases de dados de diferentes dimensionalidades, todas consideradas Big Data, junto com aplicações em conjuntos de dados reais, também considerados Big Data. O desempenho foi medido comparando algumas métricas resultantes dos estudos do LSMR, como o tempo de execução do algoritmo, com as métricas resultantes do método direto da decomposição QR, aplicado ao problema dos mínimos quadrados. Além disso, as mesmas métricas foram usadas para comparar o LSMR com outros dois métodos de Krylov, o LSQR e o método dos Gradientes Conjugados. No geral, observou-se um melhor desempenho do LSMR quanto ao tempo de execução, fornecendo estimativas de soluções equivalentes ou, em alguns casos, melhores às dos demais métodos avaliados.

Palavras-chave: regressão linear; métodos de Krylov; big data; álgebra linear numérica.

Abstract

The increasing use of massive and complex databases, known as Big Data, renders the optimization of traditional data analysis methods necessary to enable their application to this type of information. Even statistical analysis methods considered simple, such as linear regression, are inefficient when applied in their traditional form to Big Data due to their high computational cost. Consequently, they require adaptations. This paper addresses the application of the Krylov Methods, a class of algorithms, to Big Data to efficiently estimate parameters for linear regression. These methods return an approximation of the solution to the least squares problem at each iteration, being computationally more economical to obtain a satisfactory estimate of the solution compared to other traditional methods, such as the QR decomposition method applied to the least squares problem. Two Krylov methods are presented and studied in the text: the Generalized Minimum Residuals (GMRES) and LSMR, with a strong focus on the latter.

Finally, to evaluate the performance of LSMR, several simulation studies are presented in databases of different dimensionalities, along with applications in real datasets, all considered Big Data. Performance was measured by comparing metrics resulting from the LSMR studies, such as the execution time of the algorithm, with those resulting from the direct method of QR decomposition applied to the least squares problem. In addition, the same approach was employed to correlate LSMR with two other Krylov methods, LSQR and the Conjugate Gradient method. Overall, LSMR performed better regarding execution time, providing estimates of solutions similar to or even better than the other methods evaluated.

Keywords: linear regression; Krylov methods; big data; numerical linear algebra.

Lista de Figuras

3.1 Teste GMRES com $n = 100$ 33
3.2 Teste GMRES com $n = 1000$ 34
3.3 Teste GMRES com $n = 5000$ 34

5.1 Gráficos das seis simulações, comparando o EQM dos $\hat{\beta}$ de cada iteração dos algoritmos LSMR, LSQR e CG. 55

6.1 EQM dos preditores obtidos dos método iterativos por iteração. 58
6.2 Uma simples análise exploratória da base de dados das bicicletas compartilhadas. 59

A.1 Transformação de Householder do vetor x em respeito ao subespaço $S \coloneqq \langle v \rangle^{\perp}$. 72

Lista de Tabelas

5.1 Cinco simulações com $n = 10^6$ observações cada, 2 parâmetros β_0 e β_1 e uma covariável X 51
5.2 Dez simulações com $n = 10^6$ observações cada e 2 parâmetros β_0 e β_1. 52
5.3 Erros Quadráticos Médios das estimativas $\hat{\beta}_0$ e $\hat{\beta}_1$ retornadas pelo LSMR e pela método direto, junto a média dos EQMs das regressões. 52
5.4 Dez estudos de simulação com diferentes valores de observações n e covariáveis m 53
5.5 Resultados de seis simulações com diferentes valores de observações n e covariáveis m, comparando três métodos de Krylov: LSMR, LSQR e CG. 55

6.1 Comparação entre os algoritmos iterativos LSMR, LSQR e CG, junto com o método direto da decomposição QR, aplicados na análise de regressão linear da primeira base de dados real. 58
6.2 Comparação entre os algoritmos iterativos LSMR, LSQR e CG, junto com o método direto da decomposição QR, aplicados na análise de regressão linear da segunda base de dados real. 60

Lista de Algoritmos

1 Método de Arnoldi 25
2 Método de Lanczos 27
3 GMRES 30
4 Arnoldi Modificado 31
5 Fatoração QR de $\hat{H}_k$ 31
6 GMRES Otimizado 32
7 LSMR 44
8 Cálculo de $\|r_k\|_2$ 47

Lista de Símbolos

Símbolo	**Descrição**
$\langle \cdot, \cdot \rangle$	Produto interno
$\perp$	Ortogonalidade
$\|\cdot\|_p$	p-Norma
$>>$	Muito maior que
A^T	Matriz transposta
$\mathcal{O}(\cdot)$	Notação big-O
I_n	Matriz Identidade $n \times n$
$span(\mathcal{S})$	Subespaço gerado por $\mathcal{S}$
$im(A)$	Imagem da matriz A
$ker(A)$	Núcleo da matriz A
e_j^k	j-ésima coluna da matriz identidade I_k
$\kappa(A)$	Número de condicionamento da matriz A

Sumário

1 Introdução **14**

1.1 Considerações Preliminares . . . 15
1.2 Objetivos . . . 16
1.3 Organização do Trabalho . . . 16

2 Mínimos Quadrados **17**

2.1 Regressão Linear . . . 17
2.2 Resíduo . . . 19
2.3 Melhor Aproximação . . . 19
2.4 Aplicação . . . 20

3 Métodos de Krylov **21**

3.1 Subespaços de Krylov . . . 21
3.2 Ideia Geral . . . 22
3.3 Método de Arnoldi . . . 22
3.3.1 Decomposição de Arnoldi . . . 23
3.3.2 Algoritmo . . . 24
3.4 Método de Lanczos . . . 25
3.5 GMRES . . . 27
3.5.1 Teoria . . . 28
3.5.2 Algoritmo . . . 30
3.5.3 Estudo com Dados Simulados . . . 33

4 LSMR **35**

4.1 Bidiagonalização de Golub-Kahan . . . 35
4.2 Teoria . . . 37
4.2.1 Decomposições QR . . . 38
4.3 Recorrências . . . 39
4.3.1 Primeira decomposição QR . . . 40
4.3.2 Segunda decomposição QR . . . 42
4.4 Algoritmo . . . 43
4.5 Normas e condição de parada . . . 44
4.5.1 Cálculo de $\|r_k\|_2$ . . . 45
4.5.2 Cálculo de $\|x_k\|_2$ . . . 47

4.5.3 Cálculo de $\|A\|_F$ e $\kappa(A)$ . . . 48
4.5.4 Condições de Parada . . . 48

5 Estudo de Simulação **50**
5.1 Comparação com Método Direto . . . 50
5.2 Comparação com outros Métodos de Krylov . . . 54

6 Aplicação **57**
6.1 Base de dados: Timbres Musicais . . . 57
6.2 Base de dados: Bicicletas Compartilhadas . . . 59

7 Conclusões **61**
7.1 Observações Finais . . . 61
7.2 Tópicos para Trabalhos Futuros . . . 62

Referências **63**

Apêndice A Decomposição QR **67**
A.1 Processo de Gram-Schmidt . . . 67
A.1.1 PGS em Conjunto Linearmente Dependente . . . 69
A.2 Decomposição QR Reduzida . . . 70
A.3 Decomposição QR Completa . . . 70
A.4 Refletores de Householder . . . 71

Apêndice B Polinômios Mínimo e Característico de uma matriz **74**
B.1 Polinômio Mínimo . . . 74
B.2 Polinômio Característico . . . 75

Apêndice C Prova do Teorema 3.3.1 **76**

Apêndice D Normas **78**
D.1 Norma de Vetores . . . 78
D.2 Norma de Matrizes . . . 79
D.3 Número de Condicionamento . . . 80

Capítulo 1

Introdução

Muitas vezes que lidamos com análise de dados, temos que nos preocupar com o quanto aqueles dados demandam computacionalmente para uma máquina realizar sua análise. O tamanho dos dados, seu tipo, e os algoritmos utilizados são fatores que influenciam na carga computacional de sua análise. Neste texto, trabalharemos com coleções de dados classificadas como Big Data. Rigorosamente, definimos Big Data como uma coleção de dados que não pode ser controlada por meio de bases de dados tradicionais, dado seu enorme tamanho [23]. Levando em consideração o volume desse tipo de coleção, um dos desafios de se trabalhar com Big Data é aplicar uma análise estatística em toda a base de dados, já que existe uma limitação computacional e de tempo para fazê-lo.

Analisaremos os dados através do método de Regressão Linear, levando em consideração que é um modelo popular, simples e eficiente para o estudo dos dados. Apesar disso, o método de regressão tem uma carga computacional alta, como demonstraremos. Outra questão importante, digna de atenção, é ajustar o modelo de regressão a dados imperfeitos, que foram sujeitos a erros. Estudaremos também como os algoritmos tratados no texto buscam contornar esse desafio. Além disso, é fato que quanto mais observações forem fornecidas, mais precisamente os parâmetros do modelo serão estimados. O desafio de "resolver" um sistema de equações linear sobredeterminado resulta disso. Mostraremos que, nesse caso, a solução ótima é aquela que minimiza a soma dos quadrados do resíduo. Ou seja, no fim, a análise de dados utilizando a regressão linear se resume ao problema dos mínimos quadrados. Por curiosidade, quando Gauss tinha apenas dezoito anos, em 1795, ele afirmou ter descoberto o método dos mínimos quadrados [3], dando a este problema pelo menos dois séculos de existência.

Desde então, uma vasta literatura surgiu para estudar esse problema, principalmente após o advento dos computadores. Junto com esse surgimento, houve também um grande avanço na área da Álgebra Linear Numérica, que pode ser considerada a interseção da matemática e da computação para a resolução de problemas na álgebra linear. Há uma vasta literatura nessa área. Podemos dar como exemplo o livro de Golub e Van Loan [12], considerado por Trefethen e Bau [38] como a enciclopédia da álgebra linear numérica. Nesse e em livros-textos semelhantes, são tratados diversos tipos de fatorações de matrizes, como a decomposição QR, e algoritmos usados na resolução de sistemas de equações

lineares do tipo $Ax = b$, em que A é uma matriz e x, b são vetores. Esses algoritmos podem ser classificados de duas formas: os diretos e os iterativos.

Os métodos diretos são aqueles que exploram diferentes decomposições da matriz A, como as decomposições Cholesky [38, Seção 23], QR[1], ou LU [12, Seção 3.2], para simplificar o sistema e facilitar sua resolução. Os algoritmos retornam a solução exata do sistema. Apesar de parecerem ótimos na teoria, em casos onde a matriz A tem alta dimensionalidade, ou seja, os dados são volumosos, esses algoritmos muitas vezes não são viáveis na prática. Como exemplo, para a resolução das Equações Normais (Seção 2.4), um método direto tem, em média, um custo computacional [31] de $\mathcal{O}(n^3)$ flops (operações de ponto flutuante) [39].

Em contrapartida, os chamados métodos iterativos, onde se enquadram os métodos de Krylov, no mesmo exemplo, em geral, têm custo de $\mathcal{O}(n^2)$ flops ou menos [39]. Além disso, geralmente, não é necessário calcular o produto A^TA explicitamente, tornando a utilização de métodos iterativos para problemas de mínimos quadrados bastante atraente. Vagamente falando, podemos dizer que esses são métodos que retornam uma solução aproximada do sistema linear a cada iteração, tornando-a cada vez melhor à medida que o algoritmo vai atuando. Uma definição rigorosa é dada no início do Capítulo 3.

Como exemplo de trabalhos que estudam e exploram as propriedades dos métodos de Krylov pode-se citar [20], [10] e [13]. Brown e Saad exploram em [4] versões modificadas do Método de Arnoldi (Seção 3.3) e do GMRES (Seção 3.5) com o objetivo de solucionar aproximadamente sistemas de equações não lineares, mostrando uma outra área de aplicação desses algoritmos. Nenhum artigo sobre a aplicação dos métodos de Krylov na análise de regressão linear foi encontrado.

1.1 Considerações Preliminares

Para se ler o texto, é recomendado um conhecimento preliminar de conceitos básicos da Álgebra Linear como subespaços, produto interno, tipos de matrizes, multiplicação matriz/matriz e vetor/matriz. Além disso, um conhecimento básico em relação à complexidade de tempo de algoritmos irá facilitar a leitura.

Caso não seja dito o contrário, matrizes serão definidas por letras maiúsculas (A, B, ...), vetores serão definidos por letras minúsculas (v, u, ...) e constantes serão definidas por letras gregas (ρ, α, ...).

[1] A decomposição QR será bastante explorada no texto. Veja o Apêndice A.

1.2 Objetivos

Neste trabalho, estudaremos e aplicaremos na análise de regressão linear e em coleções de dados reais considerados Big Data métodos iterativos no chamado subespaço de Krylov (Seção 3.1) a fim de se obter soluções satisfatórias das equações normais, em um tempo viável e não necessitando de um grande armazenamento. A análise de Big Data é visada por grandes empresas de tecnologia e é usada em várias áreas, como, por exemplo, em aplicações de inteligência artificial. Buscamos testar e comparar o desempenho desses métodos para aprofundar no estudo de dados volumosos simulados e reais, com o objetivo de facilitar sua análise em trabalhos seguintes.

1.3 Organização do Trabalho

Começamos no Capítulo 2 revisando a definição de análise de regressão linear e introduzindo o conceito de mínimos quadrados, e como eles se relacionam. Em seguida, no Capítulo 3, definimos os subespaços de Krylov e apresentamos dois algoritmos, o método de Arnoldi e o método de Lanczos, que serão necessários para o funcionamento de alguns métodos iterativos. No fim do Capítulo 3 , introduzimos o algoritmo GMRES, explicando a teoria por trás e realizando alguns testes. O Capítulo 4 será dedicado exclusivamente ao algoritmo LSMR[2], que será o foco do texto. Vamos introduzir a bidiagonalização de Golub-Kahan e a teoria por trás desse algoritmo. Por fim, vamos estimar algumas normas relacionadas aos mínimos quadrados e analisaremos algumas condições de parada do algoritmo. Nos Capítulos 5 e 6, comparamos o LSMR com outros métodos de Krylov e com o método direto da decomposição QR, através de estudos de simulação e aplicações em bases de dados reais. As conclusões do texto estão no Capítulo 7.

[2]LSMR é o nome original do algoritmo, não sendo considerado uma sigla.

Capítulo 2

Mínimos Quadrados

O método dos Mínimos Quadrados será a ferramenta essencial para encontrar o vetor que minimiza o resíduo, que será definido na Seção 2.2, do sistema linear sobredeterminado $Ax = b$, em que $A \in \mathbb{R}^{n\times m}$ e $b \in \mathbb{R}^n$. Um sistema sobredeterminado é aquele que a matriz A possui mais linhas do que colunas, ou seja, $n > m$.

2.1 Regressão Linear

Suponha que tenhamos uma coleção de dados, com mais de uma variável e mais observações do que variáveis. Pesquisadores normalmente estão interessados em saber a relação de uma variável com parte das outras, ou até sua relação com todas as outras. A análise de regressão é um método usado para compreender essa relação.

Existem alguns tipos de regressão [44], dentre elas a regressão linear múltipla, que será estudada neste texto. Esse modelo assume que uma variável, chamada de responsiva, é função linear de outras variáveis, chamadas de independentes, que são escolhidas de acordo com o objetivo do estudo. A forma geral da regressão linear múltipla para uma observação é

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_{m-1} X_{m-1} + \epsilon \tag{2.1}$$

onde Y é a variável responsiva, β_i são os coeficientes da regressão e cada X_i representa uma das $m-1$ variáveis independentes do modelo. A variável aleatória ϵ representa o erro do modelo. Normalmente, ϵ tem distribuição normal de média 0 e variância constante σ^2. A partir dessa equação, conseguimos fazer uma predição do valor de Y baseada nos valores das variáveis X_i.

O experimento comum da regressão linear múltipla é obter n observações $(X_1^i, X_2^i, \ldots, X_{m-1}^i, Y_i)$[1] e inseri-los no modelo na forma

$$Y_i = \beta_0 + \beta_1 X_1^i + \cdots + \beta_{m-1} X_{m-1}^i + \epsilon_i, \text{ para } i = 1, 2, \ldots, n \tag{2.2}$$

com a esperança de ϵ_i sendo igual a 0 e a variância σ^2 desconhecida. Os erros aleatórios ϵ_i são independentes.

O princípio dos **Mínimos Quadrados** surge aqui. Buscamos estimar os coeficientes $\beta_0, \ldots, \beta_{m-1}$ de forma que a soma dos quadrados da distância euclidiana das variáveis Y_i e da variável estimada $\hat{Y}_i = \beta_0 + \beta_1 X_1^i + \cdots + \beta_{m-1} X_{m-1}^i$ seja a menor dentre todas as possíveis escolhas dos coeficientes, ou seja,

$$(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_{m-1}) = \underset{(\beta_0, \beta_1, \ldots, \beta_{m-1})}{\arg\min} \sum_{i=1}^{n} [Y_i - (\beta_0 + \beta_1 X_1^i + \cdots + \beta_{m-1} X_{m-1}^i)]^2. \tag{2.3}$$

Podemos definir

$$b \coloneqq \begin{bmatrix} Y_1 & Y_2 & \ldots & Y_n \end{bmatrix}^T, \quad A \coloneqq \begin{bmatrix} 1 & X_1^1 & X_2^1 & \ldots & X_{m-1}^1 \\ 1 & X_1^2 & X_2^2 & \ldots & X_{m-1}^2 \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & X_1^n & X_2^n & \ldots & X_{m-1}^n \end{bmatrix} \quad \text{e} \quad \beta \coloneqq \begin{bmatrix} \beta_0 & \beta_1 & \ldots & \beta_{m-1} \end{bmatrix}^T$$

e, assim, a equação (2.3) se torna[2]

$$\hat{\boldsymbol{\beta}} \coloneqq (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_{m-1}) = \arg\min_{\beta} \|b - A\beta\|_2^2$$

que é equivalente a

$$\hat{\boldsymbol{\beta}} = \arg\min_{x} \|b - Ax\|_2 . \tag{2.4}$$

Logo, para se estimar os coeficientes da equação (2.1), basta resolver a equação acima. O resto do texto vai se basear no tratamento desse problema.

[1] Abuso de notação. O índice i representa o i-ésimo exemplo, e não o expoente de X_j, $j = 1, \ldots, p$. O expoente em todos os casos será igual a 1.

[2] Veja o Apêndice D

2.2 Resíduo

Vamos introduzir essa seção com um importante teorema da álgebra linear, sem antes fazer a seguinte definição:

Definição 2.2.1. *Seja $A \in \mathbb{R}^{n\times m}$ e $b \in \mathbb{R}^n$. Chamamos de matriz aumentada $[A|b]$ a matriz de dimensão $n \times (m+1)$ obtida ao concatenar o vetor b na matriz A.*

Agora, podemos apresentar o teorema.

Teorema 2.2.1 (Teorema de Kronecker-Capelli, [45], Página 56)**.** *O sistema de equações $Ax = b$ tem solução se e somente se o posto[3] da matriz A é igual ao posto da matriz aumentada $[A|b]$.*

A partir do teorema conclui-se que o sistema linear $Ax = b$ tem solução somente se $[A|b]$ tem posto deficiente, o que é bastante improvável. Com efeito, o conjunto das matrizes de posto deficiente tem medida de Lebesgue igual a 0 [6]. Dessa forma, como o sistema muito provavelmente não tem solução, vamos chamar de solução do sistema sobredeterminado o vetor $\hat{x}$ tal que o vetor $b - A\hat{x}$ tenha a menor 2-norma possível. Chamaremos esse vetor de *resíduo*. Rigorosamente, procuramos encontrar o vetor $\hat{x}$ que soluciona a equação (2.4). Obviamente, caso o sistema tenha uma solução exata, o vetor $\hat{x}$ será a solução, já que a norma do resíduo é zero.

2.3 Melhor Aproximação

Seja V um espaço vetorial de dimensão m, $v \in V$, $\mathcal{U} = \{u_1, \ldots, u_k\}$, $k < m$, um conjunto ortonormal de V e $S = span(\mathcal{U})$. Defina

$$\hat{v} = \langle v, u_1\rangle u_1 + \cdots + \langle v, u_k\rangle u_k.$$

O vetor $\hat{v}$ possui duas propriedades úteis. São elas [27]:
1) $\hat{v}$ é o único vetor $s \in S$ tal que $(v - s) \perp S$
2) $\hat{v}$ é a **melhor aproximação** de v em S, ou seja, $\hat{v}$ é o vetor $s \in S$ mais próximo a v, de modo que

$$\|v - \hat{v}\|_2 < \|v - s\|_2$$

para todo $s \in S - \{\hat{v}\}$.

[3] Veja a Seção 2 de [27].

2.4 Aplicação

Queremos encontrar o vetor $A\hat{x} = \hat{v} \in im(A)$[4] que melhor se aproxima do vetor b no sistema linear $Ax = b$, ou seja, o vetor $A\hat{x}$ que satisfaz

$$\|b - A\hat{x}\|_2 < \|b - Ax\|_2$$

para todo $Ax \in im(A) - \{A\hat{x}\}$.

Como vimos na última seção, esse vetor $A\hat{x}$ existe, é único e

$$(b - A\hat{x}) = r \perp im(A)$$

ou seja, o resíduo $r \in im(A)^{\perp}$. Temos que $im(A)^{\perp} = ker(A^T)$ [27], logo

$$\begin{aligned} r \in ker(A^T) &\iff A^T r = 0 \\ &\iff A^T(b - A\hat{x}) = 0 \\ &\iff A^T A\hat{x} = A^T b. \end{aligned}$$

O sistema linear $m \times m$

$$A^T A\hat{x} = A^T b \tag{2.5}$$

é chamado de **Equações Normais** [27].

Tem-se que $A^T A$ é não singular se, e somente se, A tem posto completo [39]. Neste trabalho vamos assumir que isso é verdade, já que $n >> m$, o que torna bastante improvável que a matriz A tenha posto deficiente.

Dessa forma, o sistema (2.5) tem solução única, que é o vetor solução do sistema sobredeterminado $Ax = b$.

[4] Veja a Seção 2 de [27].

Capítulo 3

Métodos de Krylov

Nesta seção explicaremos por que os métodos iterativos nos subespaços de Krylov são vastamente utilizados para a resolução do sistema linear $Ax = b$, onde A é quadrada e grande. Não existe uma definição universal, mas podemos dizer que são métodos que retornam a cada iteração uma solução aproximada do sistema linear $Ax = b$ após repetidas operações matriz/vetor envolvendo a matriz A [29], excluindo assim métodos que envolvem essas operações na matriz transposta A^T.

3.1 Subespaços de Krylov

Seja A uma matriz de tamanho n definida como na introdução desta seção e $v \neq 0$, $v \in \mathbb{R}^n$. Como em [34], definimos a sequência de Krylov de ordem k gerada por A e v como os vetores $\{v, Av, A^2v, \ldots, A^{k-1}v\}$. A partir desse conceito, definimos o subespaço de Krylov de ordem k gerado por A e v como o subespaço gerado[1] por essa sequência, ou seja,

$$\mathcal{K}_k(A, v) = span\{v, Av, A^2v, \ldots, A^{k-1}v\} \tag{3.1}$$

e a matriz de Krylov associada como

$$K_k(A, v) = [v \,|\, Av \,|\, A^2v \,|\, \ldots \,|\, A^{k-1}v\,] \in \mathbb{R}^{n\times k}. \tag{3.2}$$

Podemos relacionar as duas definições percebendo que $\mathcal{K}_k(A, v) = im(K_k(A, v))$.

Vamos supor nesta seção e no resto do texto que os vetores $\{v, Av, A^2v, \ldots, A^{k-1}v\}$ são linearmente independentes para $k \leq n$.

[1] Para compreender o conceito de subespaço gerado, veja a Seção 1 de [27].

3.2 Ideia Geral

Considere o sistema linear $Ax = b$, onde A é quadrada, grande e esparsa, como na seção anterior. Além disso, seja A uma matriz de grande dimensão e esparsa. Os chamados Métodos de Krylov [29] são algoritmos iterativos que buscam uma solução x_k no subespaço (3.1) (gerado por A e b) a cada iteração que minimiza a norma residual, ou seja,

$$x_k = \underset{z \in \mathcal{K}_k(A,b)}{\arg\min} \|b - Az\|_2 . \tag{3.3}$$

Para demonstrar o motivo pelo qual esses subespaços são bons espaços de busca de soluções, vamos primeiro supor que A é não singular. Dessa forma, a solução do sistema linear $Ax = b$ é $x = A^{-1}b$. A matriz A^{-1} é computacionalmente cara de ser calculada, tendo complexidade de tempo igual a $\mathcal{O}(n^3)$ no clássico algoritmo de eliminação Gaussiana [36] e, no melhor caso, usando uma modificação do algoritmo de Coppersmith-Winograd [43], tendo complexidade igual a $\mathcal{O}(n^{2.373})$. Por esse motivo, recorremos a métodos computacionalmente mais viáveis, como os métodos de Krylov, que, apesar de não retornarem as soluções exatas, muitas vezes as aproximações são satisfatórias [41].

Precisamos compreender o que é o *polinômio mínimo* e o *polinômio característico* de uma matriz A para entender a lógica por trás dos métodos de Krylov. Veja o Apêndice B para uma revisão detalhada sobre esses polinômios.

Seja m igual à soma das multiplicidades dos autovalores de A, como visto no Apêndice B. Como $x = A^{-1}b$, então $x \in \mathcal{K}_m(A, b)$. Observe que $m \leq n$, então a solução sempre será encontrada em menos de n iterações, dado que os algoritmos iterativos de Krylov na k-ésima iteração buscam a solução em $\mathcal{K}_k(A, b)$. Esse fato nos faz acreditar que os subespaços de Krylov são bons espaços para se procurar a solução do sistema $Ax = b$, em que A é não singular. Quando A é singular, esses espaços continuam bons espaços de procura, mas não vamos trabalhar com esse caso neste texto. Para uma abordagem mais aprofundada, veja [20].

3.3 Método de Arnoldi

Como os vetores $\{b, Ab, A^2b, \ldots, A^{k-1}b\}$ são linearmente independentes e geram o subespaço $\mathcal{K}_k(A, b)$, eles formam uma base desse subespaço. Essa base não é uma base apropriada para os subespaços de Krylov do ponto de vista numérico, já que o vetor A^kb converge para o autovetor dominante [38], fazendo com que os vetores progressivamente

se tornem linearmente dependentes na precisão do computador. Dessa forma, a base é bastante mal-condicionada [38], o que torna o uso do algoritmo de Gram-Schmidt inviável para sua ortogonalização. Uma alternativa para a ortogonalização dessa base é o **método de Arnoldi** [3, 34, 39]. Esse algoritmo é equivalente ao processo de Gram-Schmidt Modificado[2] [38] aplicado à sequência de Krylov de ordem n gerada por A e b. Apesar disso, seu desenvolvimento teórico, baseado nos subespaços de Krylov, será útil no decorrer do texto.

3.3.1 Decomposição de Arnoldi

Para introduzir esta seção, vamos definir uma matriz semelhante a uma matriz triangular e que é bastante encontrada em decomposições:

Definição 3.3.1 ([12]). *Chamamos uma matriz H de Hessenberg Superior se $h_{ij} = 0$ caso $i > j+1$. Se $h_{ij} \neq 0$ para $i = j+1$, então chamamos H de Hessenberg Superior não reduzida. Da mesma forma, se $h_{ij} = 0$ caso $j > i+1$, então chamamos H de Hessenberg Inferior e se $h_{ij} \neq 0$ para $j = i+1$, então chamamos H de Hessenberg Inferior não reduzida.*

Seja $K_{k+1} := K_{k+1}(A, b)$ definida como em (3.2) e seja

$$K_{k+1} = Q_{k+1}R_{k+1} \tag{3.4}$$

a fatoração QR de K_{k+1}. Como K_{k+1} é invertível, as colunas de Q_{k+1} formam uma base ortonormal de $\mathcal{K}_{k+1}(A, b)$ [38]. A partir dessa matriz, podemos reduzir A a uma matriz de Hessenberg, como afirma o teorema a seguir:

Teorema 3.3.1 ([34, página 298]). *Seja Q_{k+1} o fator-Q de K_{k+1}. Então existe uma matriz $(k+1) \times k$ não reduzida de Hessenberg $\hat{H}_k$ tal que*

$$AQ_k = Q_{k+1}\hat{H}_k. \tag{3.5}$$

Reciprocamente, se Q_{k+1} é uma matriz ortonormal com primeira coluna b e satisfaz (3.5), onde $\hat{H}_k$ é uma matriz $(k+1) \times k$ superior não reduzida de Hessenberg, então Q_{k+1} é o fator-Q de K_{k+1}.

Ao leitor interessado, veja a prova do teorema no Apêndice C. Podemos, a partir do teorema, elaborar a seguinte definição:

[2] Veja o Apêndice A.

Definição 3.3.2 ([34, página 299]). *Seja $U_{k+1} \in \mathbb{R}^{n\times(k+1)}$ uma matriz ortogonal e seja U_k a matriz que consiste das k primeiras colunas de U_{k+1}. Se existe uma matriz $(k+1)\times k$ superior não reduzida de Hessenberg $\hat{H}_k$ tal que*

$$AU_k = U_{k+1}\hat{H}_k \tag{3.6}$$

*então chamamos (3.6) de **Decomposição de Arnoldi de ordem k**. Se $\hat{H}_k$ é reduzida, falamos que a decomposição de Arnoldi é **Reduzida**.*

Podemos reescrever a decomposição de Arnoldi de uma forma que irá ser útil no que se segue. Particionamos

$$\hat{H}_k = \begin{bmatrix} H_k \\ h_{k+1,k}e_k^{k^T} \end{bmatrix},$$

onde H_k é a matriz $k\times k$ de Hessenberg composta pelas primeiras k linhas de $\hat{H}_k$ e $h_{k+1,k}$ é o elemento da $(k+1)$-ésima linha e k-ésima coluna de $\hat{H}_k$. Seja $\omega_k = h_{k+1,k}$. Então a decomposição de Arnoldi (3.6) é equivalente à relação

$$AU_k = U_kH_k + \omega_k u_{k+1}e_k^{k^T}. \tag{3.7}$$

Multiplicando ambos os lados pela direita por Q_k^T temos:

$$U_k^T AU_k = H_k + \omega_k U_k^T u_{k+1}e_k^{k^T}$$

e, como U_{k+1} é ortonormal, $\omega_k U_k^T u_{k+1}e_k^{k^T} = 0$, assim

$$U_k^T AU_k = H_k. \tag{3.8}$$

O seguinte teorema, apresentado sem prova, demonstra a unicidade da decomposição de Arnoldi quando a sequência de Krylov não termina em $k+1$, ou seja, $\mathcal{K}_{k+1} \neq \mathcal{K}_{k+2}$:

Teorema 3.3.2 ([34, página 300]). *Suponha que $\mathcal{K}_{k+1} \neq \mathcal{K}_{k+2}$. Então, a não ser por sinais das colunas de Q_{k+1}, a decomposição de Arnoldi de K_{k+1} é única.*

Esse teorema será essencial para provar que a bidiagonalização de Golub-Kahan (Seção 4.1) é equivalente ao método de Lanczos (Seção 3.4) aplicado às matrizes simétricas A^TA e AA^T, dados certos vetores iniciais.

3.3.2 Algoritmo

Os teoremas 3.3.1 e 3.3.2 sugerem um algoritmo para se calcular q_{k+1} a partir de $q_1, q_2, \ldots, q_k$, ou seja, uma base ortonormal para $\mathcal{K}_{k+1}$. Escrevemos a k-ésima coluna de (3.7) como

$$Aq_k = Q_kh_k + \omega_k q_{k+1}.$$

Então, pela ortogonalidade de Q_{k+1}, obtemos

$$Q_k^T A q_k = h_k.$$

Além disso, já que $\omega_k q_{k+1} = Aq_k - Q_k h_k$ e $\|q_{k+1}\|_2 = 1$, devemos ter

$$\omega_k = \|Aq_k - Q_k h_k\|_2$$

e

$$q_{k+1} = \frac{Aq_k - Q_k h_k}{\omega_k}.$$

Essas equações nos levam ao seguinte algoritmo, o método de Arnoldi [34]:

Algoritmo 1: Método de Arnoldi

Arnoldi (A, b)
$n = size(A)$;
$q_1 = \frac{b}{\|b\|_2}$;
Inicialize $\hat{H}_1$ e Q_1;
for $k = 1$ *até* $n - 1$ **do**
  $h_k = Q_k^T A q_k$;
  $v = Aq_k - Q_k h_k$;
  $\omega_k = h_{k+1,k} = \|v\|_2$;
  $q_{k+1} = \frac{v}{\omega_k}$;
  $\hat{H}_k = \begin{bmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{bmatrix}$;
end
return $Q_n, \hat{H}_{n-1}$

O algoritmo retorna a matriz ortogonal Q_n, onde as colunas formam uma base ortonormal de $\mathcal{K}_n$, que será útil no algoritmo que veremos a seguir, o GMRES (Seção 3.5). Além disso, o método de Arnoldi retorna também a matriz de Hessenberg H_n.

3.4 Método de Lanczos

No caso particular em que A é uma matriz simétrica[3], então, por (3.8), H_k também é simétrica, visto que

$$H_k^T = (U_k^T A U_k)^T = U_k^T A^T U_k = U_k^T A U_k = H_k.$$

[3] O que ocorre, por exemplo, nas Equações Normais.

Note que isso só é possível se H_k for tridiagonal, ou seja, $h_{ij} = 0$ quando $i > j+1$ e quando $j > i+1$, onde h_{ij} são os elementos de H_k. Obviamente, o método de Arnoldi funcionaria perfeitamente nesse caso, mas podemos abusar das propriedades desse tipo de matriz e modificá-lo para necessitar de menos processamento de máquina e armazenamento de dados. Esse é o chamado **método de Lanczos** [3, 34, 39].

Primeiro, definimos

$$\nu_i = h_{ii} \quad e \quad \omega_j = h_{j-1,j}$$

para $i = 1, 2, \ldots, k$ e $j = 1, 2, \ldots, k+1$. A matriz de Hessenberg tridiagonal terá a forma

$$\hat{H}_k = \begin{bmatrix} \nu_1 & \omega_2 & & & \\ \omega_2 & \nu_2 & \omega_3 & & \\ & \omega_3 & \nu_3 & \ddots & \\ & & \ddots & \ddots & \omega_k \\ & & & \omega_k & \nu_k \\ & & & & \omega_{k+1} \end{bmatrix} = \begin{bmatrix} H_k \\ \omega_{k+1} e_k^T \end{bmatrix} \tag{3.9}$$

onde H_k é a matriz de Hessenberg obtida ao excluir a última linha de $\hat{H}_k$.

De (3.7), como $q_1 \coloneqq \frac{b}{\|b\|_2}$, temos que q_2 é definido pela fórmula

$$Aq_1 = \nu_1 q_1 + \omega_2 q_2,$$

e, da ortogonalidade de q_1 e q_2, multiplicando a equação acima por q_1^T pela direita temos

$$q_1^T A q_1 = \nu_1$$

e, já que $\|q_2\|_2 = 1$,

$$\omega_2 = \|Aq_1 - \nu_1 q_1\|_2 .$$

Para a k-ésima coluna vamos ter

$$Aq_k = \omega_k q_{k-1} + \nu_k q_k + \omega_{k+1} q_{k+1}, \tag{3.10}$$

e, usando a mesma lógica que usamos na primeira coluna, obtemos

$$q_k^T A q_k = \nu_k \quad e \quad \omega_{k+1} = \|Aq_k - \nu_k q_k - \omega_k q_{k-1}\|_2 .$$

De (3.10) podemos obter q_{k+1} da seguinte forma:

$$q_{k+1} = \frac{Aq_k - \omega_k q_{k-1} - \nu_k q_k}{\omega_{k+1}}. \tag{3.11}$$

Assim, podemos definir o método de Lanczos como o seguinte algoritmo [39]

Algoritmo 2: Método de Lanczos

Lanczos (A, b)
$n = size(A)$;
$q_1 = \frac{b}{\|b\|_2}$;
Inicialize as matrizes $\hat{H}_n$ como em (3.9) e Q_n;
$\nu_1 = q_1^T A q_1$;
$\omega_2 = \|Aq_1 - \nu_1 q_1\|_2$;
$q_2 = \frac{Aq_1 - \nu_1 q_1}{\omega_2}$;
for $k = 2$ *até* $n - 1$ **do**
  $t = Aq_k$;
  $\nu_k = q_k^T t$;
  $t = t - \omega_k q_{k-1} - \nu_k q_k$;
  $\omega_{k+1} = \|t\|_2$;
  $q_{k+1} = \frac{t}{\omega_{k+1}}$;
end
return Q_n, H_n

Os vetores que formam a matriz Q_n são chamados de vetores de Lanczos [39].

3.5 GMRES

Nesta seção, vamos introduzir o algoritmo idealizado por Saad e Schultz [30] chamado de *Generalized Minimum Residuals* (GMRES), um método de Krylov usado na resolução de sistemas lineares do tipo $Ax = b$, onde A é uma matriz quadrada. A simplicidade do algoritmo o torna um bom exemplo para demonstrar a utilidade do método de Arnoldi na resolução de sistemas lineares nos subespaços de Krylov.

Uma vantagem desse método comparado a outros, como o método do Gradiente Conjugado, criado por Stiefel e Hestenes [15], é que basta a matriz A ser quadrada não singular, ignorando outras condições mais fortes como simetria e positividade. A ideia principal do GMRES é reescrever o resíduo do sistema de forma a facilitar sua resolução a cada iteração, ou seja, a cada passo do algoritmo vamos aproximar a solução exata $x = A^{-1}b$ do sistema por uma aproximação $x_k \in \mathcal{K}_k(A, b)$ que minimiza uma forma mais simples do resíduo

$$\|r_k\|_2 = \|b - Ax_k\|_2 .$$

Vamos agora entender como resolver esse problema.

3.5.1 Teoria

Seja $A \in \mathbb{R}^{n\times n}$. O vetor $x_k \in \mathcal{K}_k(A,b)$ pode ser escrito como $x_k = K_k y_k$ para algum $y_k \in \mathbb{R}^n$. Dessa forma

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \|b - AK_k y_k\|_2 .$$

Entretanto, como vimos, esse problema continua sendo bastante instável.

Assim, vamos recorrer ao método de Arnoldi, visto que uma base ortonormal facilitaria bastante a resolução desse problema. Seja $Q_k = \begin{bmatrix} q_1 & q_2 & \dots & q_k \end{bmatrix}$ a matriz ortonormal retornada pelo Algoritmo 1. Dessa vez teremos $x_k = Q_k y_k$ para algum $y_k \in \mathbb{R}^n$. Assim,

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \|b - AQ_k y_k\|_2 ,$$

e, por (3.5),

$$\|r_k\|_2 = \|b - AQ_k y_k\|_2 = \left\|b - Q_{k+1}\hat{H}_k y_k\right\|_2 .$$

onde $\hat{H}_k$ é uma matriz de Hessenberg não reduzida superior.
Como Q_{k+1} é ortonormal,

$$\begin{aligned}
\|r_k\|_2 &= \left\|b - Q_{k+1}\hat{H}_k y_k\right\|_2 \\
&= \left\|Q_{k+1}^T(b - Q_{k+1}\hat{H}_k y_k)\right\|_2 \\
&= \left\|Q_{k+1}^T b - Q_{k+1}^T Q_{k+1}\hat{H}_k y_k\right\|_2 \\
&= \left\|Q_{k+1}^T b - \hat{H}_k y_k\right\|_2 .
\end{aligned}$$

Temos que $Q_{k+1}^T b = \begin{bmatrix} q_1^T b & q_2^T b & \dots & q_{k+1}^T b \end{bmatrix}$ e, por construção, que $q_1 = \frac{b}{\|b\|_2}$, logo $q_1^T b = \|b\|_2$ e $q_i^T b = 0$ para $i = 2, \dots, k+1$. Assim,

$$\|r_k\|_2 = \left\|Q_{k+1}^T b - \hat{H}_k y_k\right\|_2 = \left\|e_1\|b\|_2 - \hat{H}_k y_k\right\|_2 .$$

Dessa forma, o problema passa a ser encontrar o mínimo do resíduo

$$\|r_k\|_2 = \left\|e_1\|b\|_2 - \hat{H}_k y_k\right\|_2 . \tag{3.12}$$

Vale destacar que a matriz $\hat{H}$ tem dimensões $(k+1)\times k$, ou seja, menor que n, o que permite uma solução mais eficiente.

A questão agora é como encontrar o mínimo do resíduo (3.12). A resposta é aplicar a decomposição QR na matriz $\hat{H}_k$. As propriedades de uma matriz de Hessenberg facilitam na realização desse procedimento a cada iteração. Para ver isso, seja

$$\hat{H}_k = \hat{Q}_{k+1}\hat{R}_k \implies \hat{Q}_{k+1}^T\hat{H}_k = \hat{R}_k$$

a decomposição QR de $\hat{H}_k$, que é realizada na k-ésima iteração do algoritmo. Como $\hat{H}_k$ tem dimensões $(k+1)\times k$, então a matriz ortonormal $\hat{Q}_{k+1}$ tem dimensões $(k+1)\times(k+1)$ e a matriz triangular $\hat{R}_k$ tem dimensões $(k+1)\times k$. A última linha de $\hat{R}_k$ é composta de zeros, ou seja

$$\hat{R}_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix},$$

onde R_k é uma matriz triangular de tamanho $k \times k$. Se particionarmos

$$\hat{H}_{k+1} = \begin{bmatrix} \hat{H}_k & h_{k+1} \\ 0 & h_{k+2,k+1} \end{bmatrix},$$

vemos que apenas a última linha e coluna da matriz de Hessenberg são atualizadas de uma iteração para a próxima. Note que se multiplicarmos pela esquerda a matriz $\hat{H}_{k+1}$ por uma matriz formada por $\hat{Q}_{k+1}$ e o vetor e_{k+1} como linha e coluna adicionais, então vamos quase ter uma matriz triangular superior. Matematicamente, teremos

$$\begin{bmatrix} \hat{Q}_{k+1}^T & 0 \\ 0 & 1 \end{bmatrix} \hat{H}_{k+1} = \begin{bmatrix} R_k & r_{k+1} \\ 0 & \rho \\ 0 & \sigma \end{bmatrix},$$

onde as constantes ρ e σ são elementos da matriz resultante da multiplicação matricial do lado esquerdo e r_{k+1} é um vetor. A matriz da direita é triangular se, e somente se, $\sigma = 0$. Para resolver isso, basta aplicar a *rotação de Givens*

$$P(k+1,k+2) := P_k = \begin{bmatrix} I_k & 0 & 0 \\ 0 & c_k & s_k \\ 0 & -s_k & c_k \end{bmatrix},$$

onde

$$c_k = \frac{\rho}{\sqrt{\rho^2+\sigma^2}} \quad \text{e} \quad s_k = \frac{\sigma}{\sqrt{\rho^2+\sigma^2}}.$$

Dessa forma,

$$\hat{Q}_{k+2}^T = P_k \begin{bmatrix} \hat{Q}_{k+1}^T & 0 \\ 0 & 1 \end{bmatrix}.$$

Logo,

$$\hat{Q}_{k+2}^T \hat{H}_{k+1} = \begin{bmatrix} R_k & r_{k+1} \\ 0 & r_{k+1,k+1} \\ 0 & 0 \end{bmatrix} \tag{3.13}$$

é uma matriz triangular com $r_{k+1,k+1} = \sqrt{\rho^2+\sigma^2}$.

Podemos agora modificar o resíduo (3.12) da seguinte forma:

$$\begin{aligned}
\|r_k\|_2 &= \left\| e_1\|b\|_2 - \hat{H}_k y_k \right\|_2 \\
&= \left\| \hat{Q}_{k+1}^T (e_1\|b\|_2 - \hat{H}_k y_k) \right\|_2 \\
&= \left\| \hat{Q}_{k+1}^T e_1\|b\|_2 - \hat{Q}_{k+1}^T \hat{H}_k y_k) \right\|_2 \\
&= \left\| \hat{Q}_{k+1}^T e_1\|b\|_2 - \hat{R}_k y_k \right\|_2 \\
&= \left\| \hat{g}_k - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k \right\|_2,
\end{aligned}$$

onde $\hat{g}_k := \hat{Q}_{k+1}^T e_1\|b\|_2 = \begin{bmatrix} g_k \\ \omega_k \end{bmatrix}$.

Observe que, para minimizar a norma acima, basta resolver o sistema $R_k y_k = g_k$. Assim,

$$y_k = R_k^{-1} g_k. \tag{3.14}$$

3.5.2 Algoritmo

Podemos agora definir o algoritmo GMRES:

Algoritmo 3: GMRES

GMRES (A, b)
$q_1 = \frac{b}{\|b\|_2}$;
for $k = 2, 3, \ldots$ **do**
  Realize a Decomposição de Arnoldi (1) com $n = k$;
  Encontre y_k que minimiza (3.12);
  Faça $x_k = Q_k y_k$;
end
return x_k

Para otimizar a implementação acima, podemos definir novas funções que nos ajudarão no processo. Realizar a Decomposição de Arnoldi na iteração k faz com que todas as colunas das matrizes Q_k e $\hat{H}_{k-1}$ sejam calculadas novamente. A fim de evitar essas operações desnecessárias, podemos criar uma nova função com apenas as operações do laço do Algoritmo 1 que calcula as novas colunas q_{k+1} e h_k de Q_{k+1} e $\hat{H}_k$, respectivamente. Note-se que apenas as matrizes A, Q_k e $\hat{H}_{k-1}$ serão necessárias como

parâmetros da função. Dessa forma, podemos implementar o Algoritmo auxiliar 4 como alternativa à decomposição de Arnoldi.

Algoritmo 4: Arnoldi Modificado

ArnoldiMod $(A,\ Q_k,\ \hat{H}_{k-1})$

$h_k := Q_k^T A q_k$;

$v := Aq_k - Q_k h_k$;

$\omega_k = h_{k+1,k} := \|v\|_2$;

$q_{k+1} := \frac{v}{\omega_k}$;

$\hat{H}_k := \begin{bmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{bmatrix}$;

$Q_{k+1} := \begin{bmatrix} Q_k & q_{k+1} \end{bmatrix}$;

return $\hat{H}_k, Q_{k+1}$

Na seção anterior, demonstramos como encontrar um y_k que minimiza (3.12). Antes disso, será necessário realizar a decomposição QR na matriz $\hat{H}_k$. Para isso, temos o seguinte algoritmo:

Algoritmo 5: Fatoração QR de $\hat{H}_k$

QR $(\hat{H}_k,\ \hat{Q}_k,\ )$

Calcular ρ e σ:

$\rho := \begin{bmatrix} \hat{q}_{k+1} & 0 \end{bmatrix} \begin{bmatrix} h_{k+1} \\ h_{k+2,k+1} \end{bmatrix}$;

$\sigma := \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} h_{k+1} \\ h_{k+2,k+1} \end{bmatrix}$;

Calcular a matriz de Givens G_k (3.13);

$Q_{aux} := \begin{bmatrix} \hat{Q}_{k+1}^T & 0 \\ 0 & 1 \end{bmatrix}$;

$\hat{Q}_{k+1} := G_k Q_{aux}$;

$\hat{R}_k := \hat{Q}_{k+1} \hat{H}_k$;

return $\hat{R}_k, \hat{Q}_{k+1}$

Note que não foi necessário realizar a multiplicação $Q_{aux}\hat{H}_k$ para obter ρ e σ, bastando apenas realizar duas multiplicações de vetores.

Com essas duas funções conseguimos implementar uma versão otimizada e completa do Algoritmo 6:

Algoritmo 6: GMRES Otimizado

GMRESOti $(A,\ b,\ m)$
(Inicializando as matrizes Q_1 e $\hat{H}_1$)
$q_1 := \frac{b}{\|b\|_2}, \quad Q_1 := q_1;$
$h_1 := Q_1^T A q_1, \quad \hat{H}_1 := h_1;$
$v := Aq_1 - Q_1 h_1, \quad \omega_1 = h_{2,1} := \|v\|_2;$
$q_2 := \frac{v}{\omega_1}, \quad Q_2 := \begin{bmatrix} q_1 & q_2 \end{bmatrix}, \quad \hat{Q}_2 := \begin{bmatrix} \frac{h_1}{\|h_1\|_2} & \hat{q}_2 \end{bmatrix}$, onde $\hat{q}_2$ pode ser obtido pelo processo de Gram-Schmidt;
for $k = 2, 3, \ldots, m$ **do**
  Definir $\hat{H}_k, Q_{k+1} :=$ ArnoldiRed$(A,\ \hat{H}_{k-1},\ Q_k)$;
  Definir $\hat{R}_k, \hat{Q}_{k+1} :=$ QR$(\hat{H}_k,\ \hat{Q}_k)$;
end
Fazer $\hat{g} := \|b\|_2 \hat{Q}_{m+1}^T e_1 = \begin{bmatrix} g & \omega \end{bmatrix}^T;$
Seja $\hat{R}_m = \begin{bmatrix} R_m & 0 \end{bmatrix}^T;$
Defina $y := R_m^{-1} g;$
Faça $x = Q_m y;$
return x

Os passos anteriores ao laço são necessários para inicializar as matrizes $\hat{H}_1$, Q_2 e $\hat{Q}_2$. No laço, usamos as funções que definimos anteriormente para obter a Q_{k+1}, $\hat{H}_k$ e sua decomposição QR, $\hat{Q}_{k+1}$ e $\hat{R}_k$. O valor m que limita a iteração é arbitrário, podendo ser um inteiro ou uma condição, como por exemplo $\frac{\|b - Ax_k\|_2}{\|b\|_2} < \epsilon$, onde ϵ é um valor pequeno arbitrário. Veja que o lado esquerdo da desigualdade representa o erro relativo do resíduo, fazendo com que, ao usar essa condição de parada, o algoritmo pare quando esse erro for menor que ϵ. Note em

$$\|r_k\|_2 = \left\| \begin{bmatrix} g_k \\ \omega_k \end{bmatrix} - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k \right\|_2$$

que a norma mínima do resíduo é igual a ω, logo podemos usá-lo como condição do laço ao invés de calcular x_k a cada iteração. Dessa forma, precisamos calcular y_k e x_k uma única vez, após a última iteração.

O algoritmo converge em, no máximo, n iterações, como vimos no Apêndice B. Além disso, o resíduo é não crescente, ou seja, $\|r_{k+1}\|_2 \leq \|r_k\|_2$. Isso é verdade pois r_k é minimizado no subespaço $\mathcal{K}_k$, e $\mathcal{K}_{k+1} \supset \mathcal{K}_k$.

3.5.3 Estudo com Dados Simulados

Realizamos três testes envolvendo o GMRES, todos aplicando esse algoritmo em uma matriz e um vetor de tamanho n criados aleatoriamente a partir de uma distribuição uniforme no intervalo $[0, 1]$ usando a biblioteca *Scipy* [40] do Python. Além disso, usamos as bibliotecas *Numpy* [14] para realizar operações entre matrizes e/ou vetores, e a *Matplotlib* [18] para plotar os gráficos.

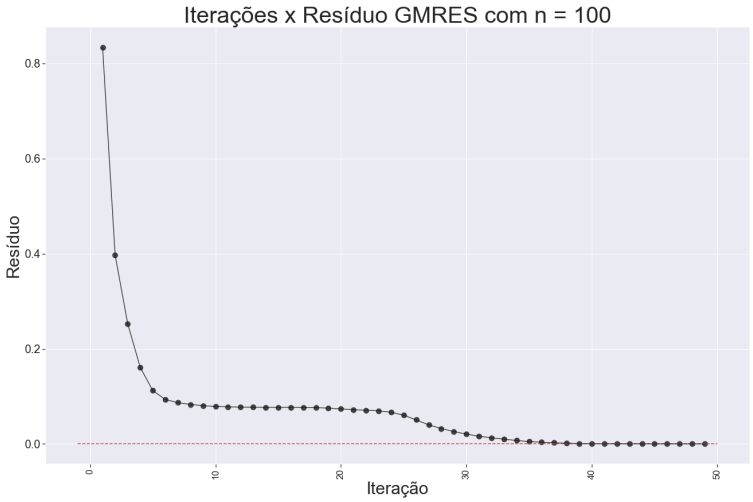
Primeiro, testamos o GMRES no sistema $Ax = b$ de tamanho $n = 100$. O algoritmo foi interrompido na iteração em que $\|b - Ax_k\|_2 < \|b\|_2 \epsilon$, onde definimos $\epsilon = 10^{-6}$. Registramos a relação entre a iteração e a norma do resíduo $\|b - Ax_k\|_2$, onde x_k é a solução aproximada retornada pelo algoritmo. A Figura 3.1 mostra essa relação. Observe que bastou apenas 49 iterações para que conseguíssemos obter uma aproximação satisfatória, menos da metade do limite superior, que é igual a 100.

Para o segundo teste, usamos $n = 1000$. O resultado está na Figura 3.2. Novamente, note que, aproximadamente, 330 iterações foram necessárias para obter a solução aproximada, um terço do limite superior de 1000 iterações.

Por último, a Figura 3.3 representa o teste com $n = 5000$.

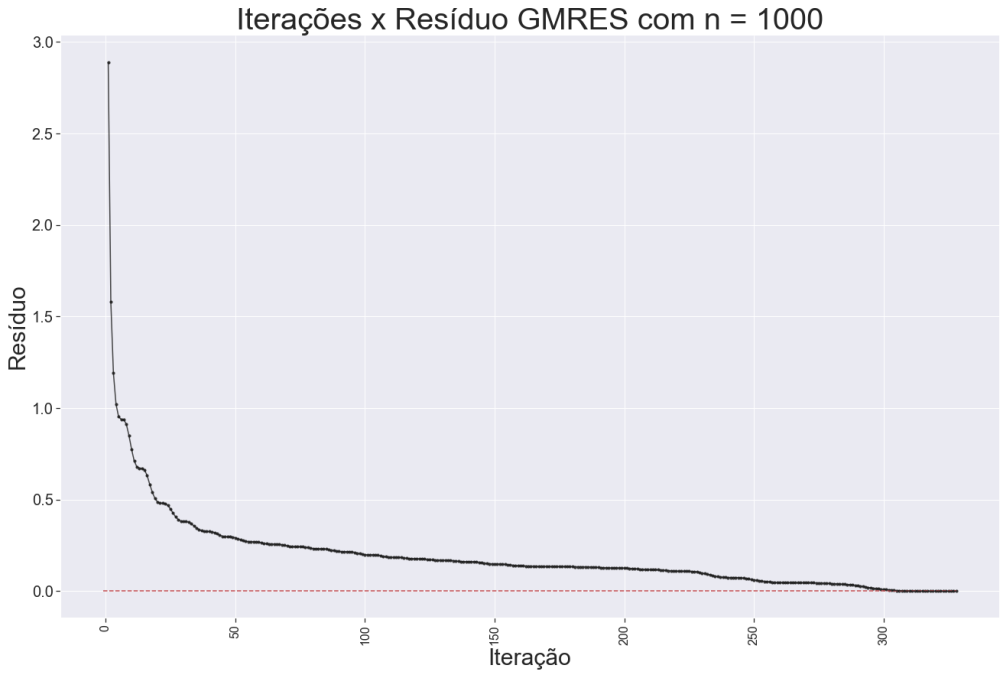
A partir dos testes, podemos observar o fato de que a sequência dos resíduos $\|r_k\|_2 = \|b - Ax_k\|_2$ é monótona não-crescente, com limite igual a 0. Também nota-se que, para a tolerância $\epsilon = 10^{-6}$, o número de iterações necessárias para o GMRES retornar a solução aproximada foi de no máximo, aproximadamente, $n/2$, com n sendo o tamanho da matriz.

Figura 3.1: Teste GMRES com $n = 100$



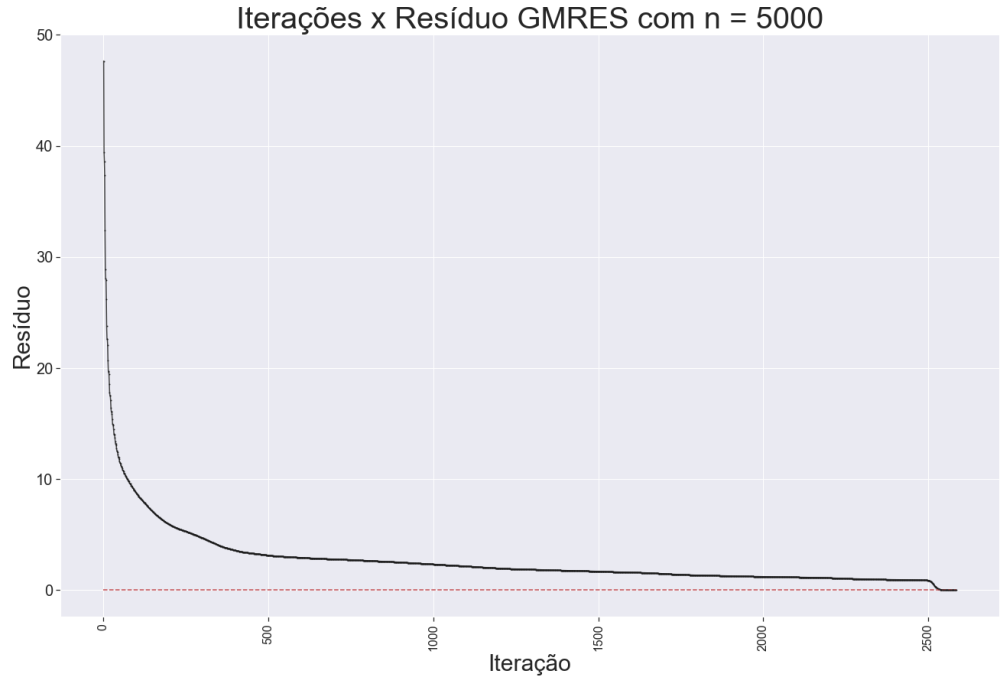
Fonte: Elaborado pelo autor.

Figura 3.2: Teste GMRES com $n = 1000$



Fonte: Elaborado pelo autor.

Figura 3.3: Teste GMRES com $n = 5000$



Fonte: Elaborado pelo autor.

Capítulo 4

LSMR

O GMRES, analisado no último capítulo, busca aproximar a solução exata de um sistema linear $Ax = b$, onde A é uma matriz quadrada. O algoritmo não é apropriado para equações normais, já que o produto A^TA é caro de se calcular, necessitando de um tempo igual a $\mathcal{O}(n^2m)$ [32], onde n e m são as dimensões da matriz A. Dessa forma, vamos analisar um algoritmo específico para esse caso, o LSMR [8], proposto por Fong e Saunders em 2011. Com melhor performance se a matriz A for esparsa, esse algoritmo iterativo busca a solução do problema dos mínimos quadrados $\arg\min_x \|b - Ax\|_2$ nos subespaços de Krylov. Fong e Saunders se basearam no processo de bidiagonalização de Golub-Kahan [11], idealizado por G. Golub e W. Kahan em 1965.

4.1 Bidiagonalização de Golub-Kahan

Seja $A \in \mathbb{R}^{m\times n}$. Sem perda de generalidade, vamos supor que $m \geq n$[1]. O objetivo do processo de bidiagonalização de Golub-Kahan é decompor a matriz A na forma $A = UBV^T$, onde $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$ são ortogonais e $B \in \mathbb{R}^{m\times n}$ é bidiagonal inferior.

Golub e Kahan em seu artigo demonstraram duas maneiras de se chegar nessa decomposição. A primeira se baseia em zerar os termos da matriz A que não estão na diagonal e sub-diagonal inferior utilizando refletores de Householder (Apêndice A.4). Uma sequência de matrizes $A = A^{(1)}, A^{(\frac{3}{2})}, A^{(2)}, \ldots, A^{(n)}, A^{(\frac{n+1}{2})}$ é construída recursivamente, onde

$$A^{(k+\frac{1}{2})} = U^{(k)}A^{(k)}, \quad \text{para } k = 1, 2, \ldots, n$$
$$A^{(k+1)} = A^{(k+\frac{1}{2})}V^{(k)}, \quad \text{para } k = 1, 2, \ldots, n-1.$$

As matrizes $U^{(k)}$ e $V^{(k)}$ são transformações de Householder que zeram os últimos $m-(k+1)$ elementos da k-ésima coluna de $A^{(k)}$ e os últimos $n-k$ elementos da k-ésima linha de

[1] Caso contrário, podemos aplicar o processo em A^T

$A^{(k+\frac{1}{2})}$, respectivamente. Essas matrizes devem ser construídas de forma que os elementos zerados anteriormente não sejam afetados. Para mais detalhes, veja a Seção 5.4.8 de [12].

A outra alternativa, que será a usada no LSMR, é relacionada ao método de Lanczos (seção 3.4) para tridiagonalização de matrizes simétricas. Por esse motivo, esse procedimento recebeu o nome de bidiagonalização de Golub-Kahan-Lanczos [24]. Para um vetor inicial u_1, onde $\|u_1\|_2 = 1$, o método constrói duas sequências de vetores $u_1, u_2, \ldots, u_k$ e $v_1, v_2, \ldots, v_k$, $k = 1, \ldots, n$, da seguinte forma:
Tomando $\beta_1 v_0 \coloneqq 0$, para $i = 1, 2, \ldots, k$,

$$\alpha_i v_i = A^T u_i - \beta_i v_{i-1}, \tag{4.1}$$

$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i, \tag{4.2}$$

onde os escalares α_i e β_{i+1} são escolhidos para serem não negativos e $\|u_{i+1}\|_2 = \|v_i\|_2 = 1$. Caso todos α_i e β_{i+1} sejam não nulos para $i = 1, 2, \ldots, k$, o processo (4.1–4.2) é totalmente definido para k iterações, e, se definirmos

$$U \coloneqq \left[\begin{array}{c|c|c|c} & & & \\ u_1 & u_2 & \ldots & u_k \\ & & & \end{array}\right], \quad V \coloneqq \left[\begin{array}{c|c|c|c} & & & \\ v_1 & v_2 & \ldots & v_k \\ & & & \end{array}\right], L \coloneqq \begin{bmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & \beta_k & \alpha_k \end{bmatrix},$$

temos que as equações 4.1 e 4.2 podem ser reescritas como

$$A^T U = V L^T,$$
$$AV = UL + \beta_{k+1} u_{k+1} e_k^{k^T},$$

onde e_k^k está definido na Lista de Símbolos. Se definirmos

$$\tilde{U} \coloneqq \left[\begin{array}{c|c} & \\ U & u_{k+1} \\ & \end{array}\right], \tilde{L} \coloneqq \left[\begin{array}{c} L \\ \hline \beta_{k+1} e_k^{k^T} \end{array}\right]$$

e, caso $\alpha_{k+1} \neq 0$, podemos realizar o passo 4.1 mais uma vez para obter

$$A^T \tilde{U} = V \tilde{L}^T + \alpha_{k+1} v_{k+1} e_{k+1}^{k+1^T}, \tag{4.3}$$

$$AV = \tilde{U} \tilde{L}. \tag{4.4}$$

Além disso, Paige prova em [24] por meio de indução que

$$U^T U = V^T V = I_k.$$

Note que, se $A = ULV^T$, então $L^T L = V^T(A^T A)V$ e $LL^T = U^T(AA^T)U$ e, como L é bidiagonal, temos que $L^T L$ e LL^T são tridiagonais. Dessa forma, como a decomposição de Arnoldi é única dadas as condições do Teorema 3.3.2, a bidiagonalização de Golub-Kahan é equivalente ao método de Lanczos aplicado às matrizes simétricas $A^T A$ e AA^T, com os vetores iniciais $A^T u_1$ e u_1, respectivamente.

A partir disso, com esses vetores iniciais definidos, vamos ter que as colunas de V formam uma base ortonormal para o subespaço $\mathcal{K}_k(A^T A, A^T u_1)$ e as colunas de U formam uma base ortonormal para o subespaço $\mathcal{K}_k(AA^T, u_1)$.

4.2 Teoria

Enquanto alguns métodos de Krylov como o GMRES e o LSQR [25], também idealizado por Saunders, se baseiam na minimização de $\|r_k\|_2$, o LSMR busca encontrar a solução que minimiza $\left\|A^T r_k\right\|_2 = \left\|A^T b - A^T A x\right\|_2$, ou seja, que minimiza a norma do resíduo das equações normais. Provaremos que $\left\|A^T r_k\right\|_2$ decresce a cada iteração do algoritmo.

Seja $A \in \mathbb{R}^{m\times n}$. A cada iteração do processo de bidiagonalização de Golub-Kahan aplicado a matriz A, com vetor inicial $u_1 = \frac{b}{\|b\|_2}$, vamos definir $V_k \coloneqq V$, $U_{k+1} \coloneqq \tilde{U}$ e $B_k \coloneqq \tilde{L}$ como nas Equações (4.3, 4.4). Como vimos na última seção, as colunas de V_k formam uma base ortonormal para $\mathcal{K}_k(A^T A, A^T u_1)$. Dessa forma, podemos escrever cada estimativa x_k da solução nos subespaços de Krylov na forma $x_k = V_k y_k$, para um $y_k \in \mathbb{R}^k$. Seja $\bar{\beta}_k \coloneqq \alpha_k \beta_k$, para todo $k \in \mathbb{N}$. Como

$$\begin{aligned} A^T r_k &= A^T b - A^T A x_k \\ &= \beta_1 \alpha_1 v_1 - A^T A V_k y_k, \end{aligned}$$

temos então

$$\begin{aligned} A^T r_k &= \bar{\beta}_1 v_1 - A^T U_{k+1} B_k y_k \\ &= \bar{\beta}_1 v_1 - (V_k B_k^T B_k + \alpha_{k+1} v_{k+1} e_{k+1}^{k+1^T} B_k) y_k \\ &= \bar{\beta}_1 v_1 - (V_k B_k^T B_k + \bar{\beta}_{k+1} v_{k+1} e_k^{k^T}) y_k \\ &= \bar{\beta}_1 v_1 - V_{k+1} \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^{k^T} \end{bmatrix} y_k \end{aligned}$$

$$= V_{k+1}\left(\bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^{k^T} \end{bmatrix} y_k\right).$$

Dessa forma, lembrando que a norma de um vetor é invariante a uma matriz ortogonal, o problema de minimização em questão se torna

$$\begin{aligned} \arg\min_{r_k} \left\| A^T r_k \right\|_2 &= \arg\min_{y_k} \left\| V_{k+1}\left(\bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} y_k\right) \right\|_2 \\ &= \arg\min_{y_k} \left\| \bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^T \end{bmatrix} y_k \right\|_2 . \end{aligned} \tag{4.5}$$

O LSMR tem como base a busca de uma solução eficiente para esse problema de mínimos quadrados.

4.2.1 Decomposições QR

Para simplificar a Equação 4.5 novamente, vamos primeiro realizar uma decomposição QR na matriz B_k:

$$Q_k^T B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix} \tag{4.6}$$

onde $Q_k \in \mathbb{R}^{(k+1)\times(k+1)}$ é uma matriz ortogonal e $R_k \in \mathbb{R}^{k\times k}$ é uma matriz triangular superior.
Note que R_k será uma matriz bidiagonal superior. Temos então

$$R_k = \begin{bmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \rho_3 & \ddots & \\ & & & \ddots & \theta_k \\ & & & & \rho_k \end{bmatrix}.$$

Além disso, o vetor g_k que resolve a equação $R_k^T g_k = \bar{\beta}_{k+1} e_k^k$ é $g_k = \varphi_k e_k^k$, onde $\varphi_k := \frac{\bar{\beta}_{k+1}}{\rho_k}$, sendo $\rho_k := (R_k)_{kk}$. Defina também $t_k := R_k y_k$. Dessa forma, a Equação 4.5 se torna

$$\arg\min_{r_k} \left\| A^T r_k \right\|_2 = \arg\min_{y_k} \left\| \bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} B_k^T B_k \\ \bar{\beta}_{k+1} e_k^{k^T} \end{bmatrix} y_k \right\|_2$$

$$= \arg\min_{y_k} \left\| \bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} R_k^T R_k \\ g_k^T R_k \end{bmatrix} y_k \right\|_2$$
$$= \arg\min_{t_k} \left\| \bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} R_k^T \\ \varphi_k e_k^T \end{bmatrix} t_k \right\|_2 . \tag{4.7}$$

Vamos realizar mais uma decomposição QR, agora na matriz $\tilde{R}_k$ definida como

$$\tilde{R}_k \coloneqq \begin{bmatrix} R_k^T & \bar{\beta}_1 e_1^k \\ \varphi_k e_k^{kT} & 0 \end{bmatrix}.$$

Assim, vamos ter

$$\bar{Q}_k^T \tilde{R}_k = \begin{bmatrix} \bar{R}_k^T & z_k \\ 0 & \bar{\zeta}_{k+1} \end{bmatrix}, \tag{4.8}$$

onde $z_k \in \mathbb{R}^k$ é um vetor, $\bar{\zeta}_{k+1}$ é uma constante, $\bar{Q}_k \in \mathbb{R}^{(k+1)\times(k+1)}$ e $\bar{R}_k \in \mathbb{R}^{k\times k}$ é uma matriz triangular definida da seguinte forma:

$$\bar{R}_k = \begin{bmatrix} \bar{\rho}_1 & \bar{\theta}_2 & & & \\ & \bar{\rho}_2 & \bar{\theta}_3 & & \\ & & \bar{\rho}_3 & \ddots & \\ & & & \ddots & \bar{\theta}_k \\ & & & & \bar{\rho}_k \end{bmatrix}. \tag{4.9}$$

Combinando essas definições e a Equação 4.7, obtemos

$$\arg\min_{r_k} \left\| A^T r_k \right\|_2 = \arg\min_{t_k} \left\| \bar{\beta}_1 e_1^{k+1} - \begin{bmatrix} R_k^T \\ \varphi_k e_k^T \end{bmatrix} t_k \right\|_2$$
$$= \arg\min_{t_k} \left\| \begin{bmatrix} z_k \\ \bar{\zeta}_{k+1} \end{bmatrix} - \begin{bmatrix} \bar{R}_k \\ 0 \end{bmatrix} t_k \right\|_2 .$$

O subproblema de minimização acima é facilmente resolvido encontrando o vetor t_k que resolve a equação $\bar{R}_k t_k = z_k$.

4.3 Recorrências

Primeiro, vamos estabelecer a recorrência de x_k. Sejam W_k e $\bar{W}_k$ as matrizes que satisfazem

$$R_k^T W_k^T = V_k^T \quad \text{e} \quad \bar{R}_k^T \bar{W}_k^T = W_k^T. \tag{4.10}$$

Note que

$$V_{k+1} = \left[\begin{array}{c|c} V_k & v_{k+1} \end{array}\right], W_{k+1} = \left[\begin{array}{c|c} W_k & w_{k+1} \end{array}\right], \bar{W}_{k+1} = \left[\begin{array}{c|c} \bar{W}_k & \bar{w}_{k+1} \end{array}\right], z_{k+1} = \begin{bmatrix} z_k \\ \zeta_{k+1} \end{bmatrix}.$$

Dessa forma, como $x_k = V_k y_k$, $R_k y_k = t_k$ e $\bar{R}_k t_k = z_k$, então

$$x_k = W_k R_k y_k = W_k t_k = \bar{W}_k \bar{R}_k t_k = \bar{W}_k z_k = x_{k-1} + \zeta_k \bar{w}_k. \tag{4.11}$$

4.3.1 Primeira decomposição QR

As recorrências das decomposições QR serão baseadas no mesmo processo utilizado no desenvolvimento do GMRES (Seção 3.5).

Para a primeira decomposição QR, que será realizada na matriz B_k (Equação 4.6), na iteração $k = 1$ temos que

$$P_1 = Q_1^T \coloneqq \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} \alpha_1 \\ \beta_2 \end{bmatrix} \quad \text{e} \quad Q_1^T B_1 = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

onde $\rho_1 \coloneqq \sqrt{\alpha_1^2 + \beta_2^2}$, $c_1 \coloneqq \frac{\alpha_1}{\rho_1}$, $s_1 \coloneqq \frac{\beta_2}{\rho_1}$ e $R_1 = \rho_1$.

Agora, definimos uma matriz auxiliar A_1 tal que

$$A_1 \coloneqq \begin{bmatrix} Q_1^T & 0 \\ 0 & 1 \end{bmatrix}.$$

Para $k = 2$, podemos utilizar a decomposição QR obtida no passo anterior junto da matriz A_1 e de uma nova matriz de rotação P_2 para obter a decomposição QR de B_2. Ao multiplicar B_2 à esquerda por A_1 vamos ter

$$A_1 B_2 = \begin{bmatrix} Q_1^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} B_1 & \alpha_2 e_2^2 \\ 0 & \beta_3 \end{bmatrix} = \begin{bmatrix} R_1 & \alpha_2 s_1 \\ 0 & \alpha_2 c_1 \\ 0 & \beta_3 \end{bmatrix} = \begin{bmatrix} R_1 & \theta_2 \\ 0 & \tilde{\rho}_2 \\ 0 & \beta_3 \end{bmatrix}, \tag{4.12}$$

onde $\theta_2 = \alpha_2 s_1$ e $\tilde{\rho}_2 \coloneqq \alpha_2 c_1$. Podemos então definir a matriz de rotação P_2 e a matriz Q_2^T resultante de sua aplicação em A_1:

$$P_2 \coloneqq \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{bmatrix} \quad e \quad P_2 A_1 \coloneqq Q_2^T,$$

onde $\rho_2 := \sqrt{\tilde{\rho}_2^2 + \beta_3^2}$, $c_2 := \frac{\tilde{\rho}_2}{\rho_2}$ e $s_2 := \frac{\beta_3}{\rho_2}$. Assim, aplicando Q_2^T pela esquerda em B_2 temos

$$Q_2^T B_2 = P_2 A_1 B_2 = P_2 \begin{bmatrix} R_1 & \theta_2 \\ 0 & \tilde{\rho}_2 \\ 0 & \beta_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{bmatrix} \begin{bmatrix} R_1 & \theta_2 \\ 0 & \tilde{\rho}_2 \\ 0 & \beta_3 \end{bmatrix} = \begin{bmatrix} R_1 & \theta_2 \\ 0 & \rho_2 \\ 0 & 0 \end{bmatrix}.$$

Podemos então generalizar o processo para um passo qualquer k, onde vamos ter

$$Q_k^T B_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix}.$$

Logo, podemos iniciar a iteração $k+1$ da seguinte forma:

$$A_k B_{k+1} = \begin{bmatrix} Q_k^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} B_k & \alpha_{k+1} e_{k+1}^{k+1} \\ 0 & \beta_{k+2} \end{bmatrix} = \begin{bmatrix} R_k & \alpha_{k+1} s_k e_k^k \\ 0 & \alpha_{k+1} c_k \\ 0 & \beta_{k+2} \end{bmatrix} = \begin{bmatrix} R_k & \theta_{k+1} e_k^k \\ 0 & \tilde{\rho}_{k+1} \\ 0 & \beta_{k+2} \end{bmatrix},$$

onde $\theta_{k+1} = \alpha_{k+1} s_k$ e $\tilde{\rho}_{k+1} := \alpha_{k+1} c_k$. Podemos então definir a matriz de rotação P_{k+1} e a matriz Q_{k+1}^T resultante de sua aplicação em A_k:

$$P_{k+1} := \begin{bmatrix} I_{k-1} & 0 & 0 \\ 0 & c_{k+1} & s_{k+1} \\ 0 & -s_{k+1} & c_{k+1} \end{bmatrix} \quad e \quad P_{k+1} A_k := Q_{k+1}^T,$$

onde $\rho_{k+1} := \sqrt{\tilde{\rho}_{k+1}^2 + \beta_{k+2}^2}$, $c_{k+1} := \frac{\tilde{\rho}_{k+1}}{\rho_{k+1}}$ e $s_{k+1} := \frac{\beta_{k+2}}{\rho_{k+1}}$.

Assim, aplicando Q_{k+1}^T pela esquerda em B_{k+1} temos

$$Q_{k+1}^T B_{k+1} = P_{k+1} A_k B_{k+1}$$
$$= P_{k+1} \begin{bmatrix} R_k & \theta_{k+1} e_k^k \\ 0 & \tilde{\rho}_{k+1} \\ 0 & \beta_{k+2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{k+1} & s_{k+1} \\ 0 & -s_{k+1} & c_{k+1} \end{bmatrix} \begin{bmatrix} R_k & \theta_{k+1} e_k^k \\ 0 & \tilde{\rho}_{k+1} \\ 0 & \beta_{k+2} \end{bmatrix} = \begin{bmatrix} R_k & \theta_{k+1} e_k^k \\ 0 & \rho_{k+1} \\ 0 & 0 \end{bmatrix}.$$

Note que $\theta_{k+1} = \alpha_{k+1} s_k = \alpha_{k+1} \left(\frac{\beta_{k+1}}{\rho_k} \right) = \frac{\bar{\beta}_{k+1}}{\rho_k} = \varphi_k$. Assim, podemos substituir a notação θ_{k+1} por φ_k.

4.3.2 Segunda decomposição QR

Seja

$$\bar{B}_k = \begin{bmatrix} R_k^T \\ \theta_{k+1} e_k^{k^T} \end{bmatrix}.$$

Para a segunda decomposição QR, que será

$$\bar{Q}_k^T \bar{B}_k = \begin{bmatrix} \bar{R}_k \\ 0 \end{bmatrix}$$

na k-iteração, vamos seguir a mesma ideia vista na primeira decomposição QR.

Assim, começamos a iteração $k+1$ aplicando a matriz $\bar{A}_k$ em $\bar{B}_{k+1}$:

$$\bar{A}_k \bar{B}_{k+1} = \begin{bmatrix} \bar{Q}_k^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{B}_k & \rho_{k+1} e_{k+1}^{k+1} \\ 0 & \theta_{k+2} \end{bmatrix} = \begin{bmatrix} \bar{R}_k & \rho_{k+1} \bar{s}_k e_k^k \\ 0 & \rho_{k+1} \bar{c}_k \\ 0 & \theta_{k+2} \end{bmatrix} = \begin{bmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k^k \\ 0 & \dot{\rho}_{k+1} \\ 0 & \theta_{k+2} \end{bmatrix},$$

onde $\bar{\theta}_{k+1} = \rho_{k+1} \bar{s}_k$ e $\dot{\rho}_{k+1} \coloneqq \rho_{k+1} \bar{c}_k$. Vamos então definir a nova matriz de rotação $\bar{P}_{k+1}$ e a matriz $\bar{Q}_{k+1}^T$ resultante de sua aplicação em $\bar{A}_k$:

$$\bar{P}_{k+1} \coloneqq \begin{bmatrix} I_{k-1} & 0 & 0 \\ 0 & \bar{c}_{k+1} & \bar{s}_{k+1} \\ 0 & -\bar{s}_{k+1} & \bar{c}_{k+1} \end{bmatrix} \quad e \quad \bar{P}_{k+1} \bar{A}_k \coloneqq \bar{Q}_{k+1}^T,$$

onde $\bar{\rho}_{k+1} := \sqrt{\bar{\rho}_{k+1}^2 + \theta_{k+2}^2}$, $\bar{c}_{k+1} := \frac{\dot{\rho}_{k+1}}{\bar{\rho}_{k+1}}$ e $\bar{s}_{k+1} := \frac{\beta_{k+2}}{\bar{\rho}_{k+1}}$. Assim, aplicando $\bar{Q}_{k+1}^T$ pela esquerda em $\bar{B}_{k+1}$ temos

$$\begin{aligned}\bar{Q}_{k+1}^T \bar{B}_{k+1} &= \bar{P}_{k+1} \bar{A}_k \bar{B}_{k+1} \\ &= \bar{P}_{k+1} \begin{bmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k^k \\ 0 & \dot{\rho}_{k+1} \\ 0 & \theta_{k+2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \bar{c}_{k+1} & \bar{s}_{k+1} \\ 0 & -\bar{s}_{k+1} & \bar{c}_{k+1} \end{bmatrix} \begin{bmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k^k \\ 0 & \dot{\rho}_{k+1} \\ 0 & \theta_{k+2} \end{bmatrix} = \begin{bmatrix} \bar{R}_k & \bar{\theta}_{k+1} e_k^k \\ 0 & \bar{\rho}_{k+1} \\ 0 & 0 \end{bmatrix}.\end{aligned} \tag{4.13}$$

Observando a última linha das matrizes das Equações 4.10, obtemos duas equações que determinam w_{k+1} e $\bar{w}_{k+1}$:

$$\begin{aligned}\theta_{k+1} w_k^T + \rho_{k+1} w_{k+1}^T &= v_{k+1}^T, \\ \bar{\theta}_{k+1} \bar{w}_k^T + \bar{\rho}_{k+1} \bar{w}_{k+1}^T &= w_{k+1}^T.\end{aligned}$$

Além disso, podemos definir mais duas equações $h_k := \rho_k w_k$ e $\bar{h}_k := \rho_k \bar{\rho}_k \bar{w}_k$ que tornam o LSMR mais eficiente. O algoritmo poderá ser apresentado agora.

4.4 Algoritmo

O seguinte algoritmo implementa o que foi visto anteriormente a fim de encontrar a solução de $Ax = b$ que minimiza a norma $\left\|A^T r_k\right\|_2$, onde $r_k := b - Ax_k$. O valor dessa norma a cada iteração será calculado na sequência. Vale lembrar que x_k é a solução do sistema sobredeterminado se e somente se $\left\|A^T r_k\right\|_2 = 0$, ou seja, quando x_k soluciona as equações normais. Na prática, o algoritmo é interrompido antes disso acontecer, retornando uma aproximação da solução do sistema.

Algoritmo 7: LSMR

LSMR $(A,\, b)$

$\beta_1 := \|b\|_2, \quad u_1 := \frac{b}{\beta_1};$

$\alpha_1 := \left\|A^T u_1\right\|_2, \quad v_1 := \frac{A^T u_1}{\alpha_1};$

$\tilde{\rho}_1 := \alpha_1, \quad \bar{\zeta}_1 := \alpha_1 \beta_1;$

$\rho_0 := 1, \quad \bar{\rho}_0 := 1;$

$\bar{c}_0 := 1, \quad \bar{s}_0 := 0;$

$h_1 := v_1, \quad \bar{h}_0 := 0;$

$x_0 = 0;$

for $k = 1, 2, 3, \ldots$ **do**

(Continuar bidiagonalização)

$\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k;$

$\alpha_{k+1} v_{k+1} = A^T u_{k+1} - \beta_{k+1} v_k;$

(Construindo e aplicando P_k)

$\rho_k = \left(\tilde{\rho}_k^2 + \beta_{k+1}^2\right)^{\frac{1}{2}}, \quad c_k = \frac{\tilde{\rho}_k}{\rho_k}, \quad s_k = \frac{\beta_{k+1}}{\rho_k};$

$\theta_{k+1} = s_k \alpha_{k+1}, \quad \tilde{\rho}_{k+1} = c_k \alpha_{k+1};$

(Construindo e aplicando $\bar{P}_k$)

$\bar{\theta}_k = \bar{s}_{k-1} \rho_k, \quad \bar{\rho}_k = \left((\bar{c}_{k-1} \rho_k)^2 + \theta_{k+1}^2\right)^{\frac{1}{2}};$

$\bar{c}_k = \frac{\bar{c}_{k-1} \rho_k}{\bar{\rho}_k}, \quad \bar{s}_k = \frac{\theta_{k+1}}{\bar{\rho}_k};$

$\zeta_k = \bar{c}_k \bar{\zeta}_k, \quad \bar{\zeta}_{k+1} = -\bar{s}_k \bar{\zeta}_k;$

(Atualizar h, $\bar{h}$ e x)

$\bar{h}_k = h_k - \left(\frac{\bar{\theta}_k \rho_k}{\rho_{k-1} \bar{\rho}_{k-1}}\right) \bar{h}_{k-1};$

$x_k = x_{k-1} + \left(\frac{\zeta_k}{\rho_k \bar{\rho}_k}\right) \bar{h}_k;$

$h_{k+1} = v_{k+1} - \left(\frac{\theta_{k+1}}{\rho_k}\right) h_k;$

end

return x

4.5 Normas e condição de parada

Nesta seção estimaremos as normas $\|r_k\|_2$, $\left\|A^T r_k\right\|_2$, $\|x_k\|_2$ e $\|A\|_F$ em conjunto com $\kappa(A)$[2], que serão úteis para estabelecer condições de parada para o LSMR. É de se imaginar que esses valores podem ser facilmente calculados, já que temos todos os vetores e matrizes necessárias, mas Fong e Saunders [8] encontraram métodos computacionalmente

[2] Veja o Apêndice D

mais eficientes para se obter essas normas.

Note que $\left\|A^T r_k\right\|_2 = |\bar{\zeta}_{k+1}|$ e, como $\bar{\zeta}_{k+1} = -\bar{s}_k \bar{\zeta}_k$, temos que $\left\|A^T r_k\right\|_2 > \left\|A^T r_{k+1}\right\|_2$, para todo $k \in \mathbb{N}$, já que $|\bar{s}_k| < 1$. Dessa forma, $\left\|A^T r_k\right\|_2 \underset{k}{\to} 0$. Como visto no Apêndice B, essa convergência acontece em no máximo n iterações.

4.5.1 Cálculo de $\|r_k\|_2$

A ideia desta seção é definir algumas transformações de vetores/matrizes e gozar de propriedades das matrizes ortogonais a fim de encontrar uma expressão simplificada para a norma do resíduo r_k, já que o método tradicional, calculando $r_k = b - Ax_k$, é computacionalmente caro se realizado em cada iteração.

Primeiro, vamos transformar $\bar{R}_k^T$, definida em (4.9), em uma matriz bidiagonal superior realizando mais uma decomposição QR:

$$\tilde{Q}_k^T \bar{R}_k^T = \tilde{R}_k. \tag{4.14}$$

A matriz $\tilde{Q}_k$ será definida a partir de rotações $\tilde{P}_k$, semelhantemente às matrizes Q_k e $\bar{Q}_k$ vistas nas primeiras duas decomposições QR. Além disso, seja

$$\tilde{t}_k = \tilde{Q}_k^T t_k \quad e \quad \tilde{b}_k = \begin{bmatrix} \tilde{Q}_k^T & 0 \\ 0 & 1 \end{bmatrix} Q_k^T e_1 \beta_1.$$

Dessa forma

$$\begin{aligned}
r_k &= b - Ax_k \\
&= \beta_1 u_1 - AV_k y_k \\
&= \beta_1 U_{k+1} e_1 - U_{k+1} B_k y_k \\
&= U_{k+1} \left(e_1 \beta_1 - Q_k \begin{bmatrix} R_k \\ 0 \end{bmatrix} y_k \right) \\
&= U_{k+1} \left(e_1 \beta_1 - Q_k \begin{bmatrix} t_k \\ 0 \end{bmatrix} \right) \\
&= U_{k+1} \left(Q_k \begin{bmatrix} \tilde{Q}_k & 0 \\ 0 & 1 \end{bmatrix} \tilde{b}_k - Q_k \begin{bmatrix} \tilde{Q}_k \tilde{t}_k \\ 0 \end{bmatrix} \right) \\
&= U_{k+1} Q_k \left(\begin{bmatrix} \tilde{Q}_k & 0 \\ 0 & 1 \end{bmatrix} \tilde{b}_k - \begin{bmatrix} \tilde{Q}_k & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{t}_k \\ 0 \end{bmatrix} \right)
\end{aligned}$$

$$= U_{k+1} Q_k \begin{bmatrix} \tilde{Q}_k & 0 \\ 0 & 1 \end{bmatrix} \left(\tilde{b}_k - \begin{bmatrix} \tilde{t}_k \\ 0 \end{bmatrix} \right).$$

Assim, como U_{k+1}, Q_k e $\tilde{Q}_k$ são teoricamente ortogonais, temos

$$\|r_k\|_2 = \left\| \tilde{b}_k - \begin{bmatrix} \tilde{t}_k \\ 0 \end{bmatrix} \right\|_2.$$

Podemos escrever os vetores $\tilde{b}_k$ e $\tilde{t}_k$ da seguinte forma:

$$\tilde{b}_k := \begin{bmatrix} \tilde{\beta}_1 \\ \vdots \\ \tilde{\beta}_{k-1} \\ \dot{\beta}_k \\ \ddot{\beta}_{k+1} \end{bmatrix}, \quad \tilde{t}_k := \begin{bmatrix} \tilde{\tau}_1 \\ \vdots \\ \tilde{\tau}_{k-1} \\ \dot{\tau}_k \end{bmatrix}. \tag{4.15}$$

Note que $\tilde{t}_k$ pode ser calculado através da equação

$$\tilde{R}_k^T \tilde{t}_k = z_k. \tag{4.16}$$

Ademais, temos que, em (4.15), $\tilde{\beta}_i = \tilde{\tau}_i$ para $i = 1, \ldots, k-1$. A prova desse fato é longa e não somará a discussão, logo será omitida. Aos interessados, veja o Apêndice A de [8]. Com essa informação, podemos calcular $\|r_k\|_2$ usando apenas os últimos dois elementos de $\tilde{b}_k$ e o último elemento de $\tilde{t}_k$, como mostra o seguinte algoritmo, que deve ser implementado em conjunto com o Algoritmo 7:

Algoritmo 8: Cálculo de $\|r_k\|_2$

normaResíduo $(A,\ b)$
$\ddot{\beta}_1 := \beta_1, \quad \dot{\beta}_0 := 0;$
$\dot{\rho}_0 := 1, \quad \tilde{\tau}_{-1} := 0;$
$\tilde{\theta}_0 := 0, \quad \zeta_0 := 0;$
for $k = 1, 2, 3, \ldots$ **do**
  (Aplicando P_k)
  $\dddot{\beta}_k = c_k\ddot{\beta}_k, \quad \ddot{\beta}_{k+1} = -s_k\ddot{\beta}_k;$
  (Se $k \geq 2$, construir e aplicar $\tilde{P}_{k-1}$)
  $\bar{\bar{\rho}}_{k-1} = \left(\dot{\rho}_{k-1}^2 + \bar{\theta}_k^2\right)^{\frac{1}{2}};$
  $\tilde{c}_{k-1} = \frac{\dot{\rho}_{k-1}}{\bar{\bar{\rho}}_{k-1}}, \quad \tilde{s}_{k-1} = \frac{\bar{\theta}_k}{\bar{\bar{\rho}}_{k-1}};$
  $\tilde{\theta}_k = \tilde{s}_{k-1}\bar{\rho}_k, \quad \dot{\rho}_k = \tilde{c}_{k-1}\bar{\rho}_k;$
  $\tilde{\beta}_{k-1} = \tilde{c}_{k-1}\dot{\beta}_{k-1} + \tilde{s}_{k-1}\dddot{\beta}_k;$
  $\dot{\beta}_k = \tilde{c}_{k-1}\dddot{\beta}_k - \tilde{s}_{k-1}\dot{\beta}_{k-1};$
  (Atualizar $\tilde{t}_k$ a partir da Equação 4.16)
  $\tilde{\tau}_{k-1} = \frac{\zeta_{k-1} - \tilde{\theta}_{k-1}\tilde{\tau}_{k-2}}{\bar{\bar{\rho}}_{k-1}}, \quad \dot{\tau}_k = \frac{\zeta_k - \tilde{\theta}_k\tilde{\tau}_{k-1}}{\dot{\rho}_k};$
  (Formar $\|r_k\|_2$)
  $\gamma = \left(\dot{\beta}_k - \dot{\tau}_k\right)^2 + \ddot{\beta}_{k+1}^2, \quad \|r_k\|_2 = \sqrt{\gamma};$
end
return $\|r_k\|_2$

4.5.2 Cálculo de $\|x_k\|_2$

Da Equação 4.11 temos que $x_k = V_k R_k^{-1}\bar{R}_k^{-1}z_k$. Da terceira decomposição QR (4.14) e de uma nova decomposição QR

$$\dot{Q}_k^T(\tilde{Q}_k^T R_k)^T = \dot{R}_k,$$

podemos escrever

$$x_k = V_k R_k^{-1}\bar{R}_k^{-1}z_k = V_k R_k^{-1}\bar{R}_k^{-1}\bar{R}_k\tilde{Q}_k\tilde{z}_k = V_k R_k^{-1}\tilde{Q}_k\tilde{Q}_k^T R_k\dot{Q}_k\dot{z}_k = V_k\dot{Q}_k\dot{z}_k,$$

onde $\tilde{z}_k$ e $\dot{z}_k$ são os vetores que solucionam as equações $\tilde{R}_k^T\tilde{z}_k = z_k$ e $\dot{R}_k^T = \tilde{z}_k$. Como V_k é teoricamente ortogonal, temos que $\|x_k\|_2 = \|\dot{z}_k\|_2$. Além disso, como a última diagonal de $\tilde{R}_k$ e a submatriz $\dot{R}_k[k-1, k; k-1, k]$, composta pelos elementos das últimas duas linhas e colunas de $\dot{R}_k$, são alteradas a cada iteração, o cálculo de $\|x_k\|_2$ pode ser atualizado de forma computacionalmente econômica.

Iremos omitir o pseudocódigo, visto que o processo até se chegar nas recorrências é semelhante aos usados nos Algoritmos 7 e 8, agregando pouco ao conteúdo do texto.

4.5.3 Cálculo de $\|A\|_F$ e $\kappa(A)$

Note que de (4.1) temos que $v_k \in im(A^T) = ker(A)^\perp = ker(A^TA)^\perp$ e de (4.4) temos

$$B_k^T B_k = V_k^T A^T A V_k.$$

Pelo Teorema Min-Max de Courant-Fischer [37], sabemos que os valores singulares de B_k são limitados superior e inferiormente pelos valores singulares de A. Além disso, pelo Teorema de Interlaço de Cauchy para matrizes hermitianas [19], os valores singulares de B_k são interlaçados pelos valores singulares de A. Dessa forma, podemos usar $\|B_k\|_F$ como uma cota inferior para $\|A\|_F$, já que

$$\|B_k\|_F \leq \|B_{k+1}\|_F \leq \|A\|_F, \quad \forall k \in \mathbb{N}.$$

Como $\|B_{k+1}\|_F^2 = \|B_k\|_F^2 + \alpha_k^2 + \beta_{k+1}^2$, a norma de B_k pode ser atualizada de forma computacionalmente econômica a cada iteração.

De (4.6), (4.8) e (4.13) obtemos:

$$Q_{k+1} B_k \bar{Q}_k^T = \begin{bmatrix} \bar{R}_{k-1}^T & 0 \\ \bar{\theta}_k e_{k-1}^{k-1^T} & \bar{c}_{k-1}\rho_k. \end{bmatrix}$$

Note que o lado direito é igual a $\bar{R}_k^T$, exceto pelo último elemento da diagonal. Como os valores singulares de B_k são aproximados pelos elementos da diagonal da matriz acima [33] e esses elementos são todos positivos, podemos estimar $\kappa(A)$ pela razão entre os maiores e menores valores do conjunto $\{\bar{\rho}_1, \ldots, \bar{\rho}_{k-1}, \bar{c}_{k-1}\rho_k\}$. Novamente, esses valores podem ser atualizados de forma computacionalmente barata.

4.5.4 Condições de Parada

Vimos que o algoritmo irá convergir para a solução em até no máximo n iterações, mas uma solução satisfatória pode ser encontrada em muito menos passos. Cabe ao pesquisador decidir qual condição de parada será a mais adequada no experimento. Destacaremos três regras, sugeridas por Paige e Saunders como critérios de parada para o algoritmo LSQR [25], que podem ser aplicadas em métodos iterativos como um todo:

S1: Parar se $\|r_k\|_2 \leq \text{BTOL}\|b\|_2 + \text{ATOL}\|A\|_F\|x_k\|_2$,

S2: Parar se $\left\|A^T r_k\right\|_2 \leq \text{ATOL}\|A\|_F\|x_k\|_2$,

S3: Parar se $\kappa(A) \geq$ CONLIM,

onde ATOL, BTOL e CONLIM são constantes independentes de m ou n, escolhidas arbitrariamente pelo pesquisador. S1 se aplica a sistemas consistentes [3], permitindo incertezas em A e b [16, Teorema 7.1]. S2 é aplicável em sistemas inconsistentes[4] e S3 é aplicável em ambos. Uma discussão mais aprofundada, além de justificativas para as regras, é dada em [25] e [35]. Chang, Paige e Titley-Peloquin [5] analisam e tecem críticas aos critérios de parada S1, S2 e S3, afirmando que são regras conservadoras. Os autores propõem uma nova condição de parada, que contorna esse problema, mas que é inviável de ser aplicada na prática, pelo seu alto custo computacional.

[3] Sistemas consistentes são aqueles que possuem pelo menos uma solução.
[4] Sistemas inconsistentes não possuem solução.

Capítulo 5

Estudo de Simulação

Neste capítulo, tem-se por objetivo analisar a performance do algoritmo LSMR através de estudos de simulações. Vamos aplicá-lo a uma grande base de dados gerada no Python, no ambiente Jupyter, usando a biblioteca Numpy [14].

5.1 Comparação com Método Direto

A ideia do estudo desta seção é comparar o LSMR com o método direto da decomposição QR para solução do problema dos mínimos quadrados[1]. Como a solução do problema dos mínimos quadrados é equivalente a encontrar uma solução para as equações normais $A^TAx = A^Tb$, então, se $A = QR$ é a decomposição QR de A, vamos ter

$$\begin{aligned} A^TAx = A^Tb &\implies (QR)^TQRx = (QR)^Tb \\ &\implies R^TQ^TQRx = R^TQ^Tb \\ &\implies R^TRx = R^TQ^Tb \\ &\implies Rx = Q^Tb \end{aligned}$$

Assim, o problema dos quadrados mínimos se resume a solucionar um sistema facilmente resolvido através de substituição. Entretanto, realizar a decomposição QR usando refletores de Householder (Veja o Apêndice A) em uma matriz $A \in \mathbb{R}^{m\times n}$ tem gasto computacional de aproximadamente $2mn^2 - \frac{2}{3}n^3$ flops [38], quando n e m são grandes. Por outro lado, cada iteração do LSMR tem custo igual a $3m + 5n$ flops [8].

A condição de parada do LSMR será o EQM (Erro Quadrático Médio) do preditor obtido pela decomposição QR[2], ou seja, o algoritmo irá parar quando o EQM do preditor obtido pelo LSMR se torne menor que o EQM do preditor obtido da decomposição QR.

[1] Veja a Seção 1 para uma revisão do que são métodos diretos e suas diferenças em relação aos métodos iterativos.

[2] EQM do preditor é a média do quadrado dos erros $(y_i - \hat{y}_i)$, onde $\hat{y}$ é a label retornada pelo algoritmo. O EQM tem relação com a norma do resíduo. Veja a Seção 4.5.1

Essa condição de parada será útil para se comparar o tempo de execução dos algoritmos, algo de extrema importância quando aplicados a Big Data. O tempo será medido contando o número de iterações k do LSMR até se chegar na estimativa e fazendo $k(3m + 5n)$ para se comparar com os $2mn^2 - \frac{2}{3}n^3$ flops do método direto. Uma outra opção seria considerar $\left\|A^T r_k\right\|_2$, que representa o erro nas equações normais, mas, levando em consideração erros de arredondamento, a condição de parada $\left\|A^T r_k\right\|_2 = 0$ resultaria em mais iterações do que o necessário.

Como dito, a base de dados será gerada pelo Numpy. Em toda a seção, serão feitas cinco ou dez simulações por estudo, devido ao grande tamanho da base de dados e pelo fato de que todos os resultados serão anotados manualmente. Além disso, as observações e os erros ϵ serão gerados usando uma distribuição normal com média $\mu_X = 50$ e variância $\sigma_X^2 = 20$ e $\mu_\epsilon = 0$ e variância $\sigma_\epsilon^2 = 1$, respectivamente. Os parâmetros β serão gerados a partir de uma distribuição uniforme discreta, com valores $3 \leq \beta \leq 7$. As labels do problema serão geradas através da fórmula $Y = \tilde{X}\beta + \epsilon$, onde $\tilde{X}$ é a matriz obtida ao se concatenar um vetor de 1's na matriz de observações, antes da primeira coluna.

Após isso, solucionamos as equações normais $\tilde{X}^T\tilde{X}\hat{\beta} = \tilde{X}^T Y$ usando ambos os métodos. O primeiro teste será com $n = 10^6$ observações, com 2 parâmetros β, definidos como β_0 e β_1, indicados na tabela, junto com o EQM do preditor, arredondado em 5 casas decimais. Além disso, as estimativas $\hat{\beta}_0$ e $\hat{\beta}_1$ retornadas pelo LSMR, arredondadas em 5 casas decimais, junto com o número de iterações k realizadas, também são mostradas. No total, cinco testes foram realizados:

β_0	$\hat{\beta}_0^{LSMR}$	β_1	$\hat{\beta}_1^{LSMR}$	EQM Pred.	k
4	4.00100	5	4.99997	1.00168	2
6	6.00047	6	5.99998	1.00021	2
5	4.99722	7	7.00003	0.99588	2
3	2.99701	4	4.00009	0.99925	2
7	6.99717	3	3.00004	1.00081	2
			Média EQM:	0.99956	

Tabela 5.1: Cinco simulações com as mesmas $n = 10^6$ observações cada, 2 parâmetros β_0 e β_1 e uma covariável X, com algumas métricas obtidas pelo LSMR.

Podemos realizar mais dez simulações e observar o EQM das estimativas de cada um dos betas obtidos pelo LSMR e pela decomposição QR. As observações foram mantidas fixas, alterando apenas o valor de β_0 e β_1 entre as simulações, a fim de se manter a consistência e analisar se apenas o valor de β tem influência na performance dos algoritmos. As novas simulações estão na Tabela 5.2 e os EQMs dos $\hat{\beta}$ estão na Tabela 5.3. O EQM de $\hat{\beta}_0$ e $\hat{\beta}_1$ foi calculado a partir da equação

$$EQM(\hat{\beta}_i) = \frac{1}{10}\sum_{k=1}^{10}(\beta_i - \hat{\beta}_i^k)^2, \quad i = 0, 1 \tag{5.1}$$

onde k indica o resultado na k-ésima iteração.

β_0	$\hat{\beta}_0^{LSMR}$	$\hat{\beta}_0^{QR}$	β_1	$\hat{\beta}_1^{LSMR}$	$\hat{\beta}_1^{QR}$	EQM_{LSMR}	EQM_{QR}
6	6.00356	5.99772	6	5.99995	6.00004	1.00050	1.00029
3	3.00328	3.00276	3	2.99992	2.99993	1.00028	1.00204
4	3.99886	3.99717	5	5.00001	5.00005	0.99899	1.0043
6	5.99955	5.99902	5	5.00001	5.00004	1.00001	0.99834
7	7.00086	6.99835	5	4.99994	5.00003	1.00021	1.00148
6	5.99980	6.00132	7	7.00001	6.99998	0.99823	1.00063
6	5.99621	5.99257	4	4.00004	4.00012	1.00016	1.00171
5	5.00554	5.00103	4	3.99987	3.99998	1.00033	1.00219
5	4.99979	4.99536	6	5.99998	6.00005	1.00026	1.00176
5	4.99986	4.99931	7	6.99998	7.00001	1.00133	1.00122

Tabela 5.2: Dez simulações com as mesmas $n = 10^6$ observações cada, 2 parâmetros β_0 e β_1, comparando as estimativas obtidas pelo LSMR com as estimativas obtidas usando o método direto da decomposição QR.

EQM_{LSMR} de $\hat{\beta}_0$:	7.08331×10^{-6}
EQM_{QR} de $\hat{\beta}_0$:	10.45217×10^{-6}
EQM_{LSMR} de $\hat{\beta}_1$:	3.21000×10^{-6}
EQM_{QR} de $\hat{\beta}_1$:	3.72515×10^{-6}
Média EQM_{LSMR} da Reg.:	1.00003
Média EQM_{QR} da Reg.:	1.00101

Tabela 5.3: Erros Quadráticos Médios das estimativas $\hat{\beta}_0$ e $\hat{\beta}_1$ retornadas pelo LSMR e pela método direto, junto a média dos EQMs das regressões.

Note na Tabela 5.3 como o LSMR desempenhou melhor que o método direto, apesar de a diferença ser pequena na prática. De forma geral, nota-se que ambos os métodos tiveram boa performance em relação às métricas analisadas, alcançando estimativas satisfatórias em todas as simulações.

Agora, faremos simulações comparando os EQMs dos $\hat{\beta}$s do LSMR com os EQMs dos $\hat{\beta}$s do método da decomposição QR, enquanto fixamos o número de observações e alteramos o número de covariáveis, e enquanto fixamos o número de covariáveis e variamos o número de observações. Os números de observações e covariáveis serão definidos arbitrariamente. Não vamos comparar o EQM dos preditores, já que a diferença, na prática, como visto na Tabela 5.2, é irrelevante. Vamos definir uma constante

$$\eta \coloneqq \frac{\text{flops QR}}{\text{flops LSMR}} = \frac{2nm^2 - \frac{2}{3}m^3}{k(3n+6m)},$$

que relaciona o número de operações dos algoritmos, onde n é o número de observações e m o número de covariáveis. Além disso, vamos medir o tempo de execução de cada algoritmo em segundos, definidos como T_{LSMR} e T_{QR}, para comparar diretamente os métodos e analisar a relação entre o número de operações e o tempo de execução. Essa medição vai ser realizada através da biblioteca Timeit do Python. As métricas resultantes das simulações estão na Tabela 5.4. Valores maiores de n e m não foram possíveis por falta de armazenamento na máquina onde as simulações foram realizadas.

n	m	$\text{EQM}_{LSMR}\hat{\beta}$	$\text{EQM}_{QR}\hat{\beta}$	T_{LSMR}	T_{QR}	k	η
10^6	5	9.32×10^{-6}	2.43×10^{-6}	0.103	0.107	6	2.8
10^6	10	11.95×10^{-6}	3.12×10^{-6}	0.178	0.355	7	9.5
10^6	50	5.95×10^{-5}	5.95×10^{-5}	0.716	3.515	10	166.6
10^6	100	11.30×10^{-5}	11.30×10^{-5}	1.238	11.629	11	605.9
10^6	300	32.30×10^{-5}	32.32×10^{-5}	3.583	48.86	12	4997
3×10^6	10	5.10×10^{-7}	5.26×10^{-7}	0.62	1.26	7	9.52
5×10^6	10	2.508×10^{-6}	2.505×10^{-6}	1.05	2.11	7	9.52
10×10^6	10	5.681×10^{-8}	5.683×10^{-8}	2.28	4.72	7	9.52
20×10^6	10	1.80×10^{-7}	2.15×10^{-7}	4.48	9.49	7	9.52
30×10^6	10	4.73	26.72	5.82	30.89	6	11.11

Tabela 5.4: Resultados de dez estudos de simulação com diferentes valores de observações n e covariáveis m, comparando o LSMR com o método direto da decomposição QR.

Note na Tabela 5.4 que, quando n está fixo e m variando, não há uma relação direta de proporcionalidade entre a razão de operações η e o tempo T de execução dos algoritmos.

Observa-se, contudo, uma correlação, visto que na medida em que η cresce, T também cresce. Além disso, T_{LSMR} tende a aumentar mais com o aumento de m do que com o aumento de n, fato este explicado pelo custo computacional de $3n+6m$ flops por iteração do LSMR. Observe que, quando $n >> m$, o fator $6m$ se torna irrelevante. O número de iterações do LSMR tende a se manter constante quando m está fixo e n varia, já que k está diretamente ligado ao EQM do preditor do método direto, que é estável, visto que o número de parâmetros está fixo. Um fato curioso ocorre quando $n = 30 \times 10^6$ e $m = 10$. Note como os EQMs dos $\hat{\beta}$ do LSMR e do método da decomposição QR disparam, mostrando o provável surgimento de uma instabilidade nesses algoritmos a partir da quantidade de observações. Ademais, como $n >> m$, k é quase constante e

$$\eta = \frac{2nm^2 - \frac{2}{3}m^3}{k(3n+6m)} = \frac{m^2(2n - \frac{2}{3}m)}{k(3n+6m)} \approx \frac{2m^2n}{3kn} = \frac{2}{3} \cdot \frac{m^2}{k}$$

então η também se mantém quase constante, como vemos na Tabela 5.4. Por fim, percebe-se que $T_{LSMR} < T_{QR}$ em todos os casos, como era de se esperar.

5.2 Comparação com outros Métodos de Krylov

O objetivo desta seção é comparar o LSMR com outros dois métodos de Krylov já citados no texto, o LSQR [25] e o método dos Gradientes Conjugados [15], que chamaremos aqui de CG. Os três algoritmos serão executados em bases de dados de diferentes dimensionalidades, como na seção anterior, e várias métricas de performance serão anotadas e comparadas.

A condição de parada de todos os algoritmos será a Regra S2 da Seção 4.5.4, levando em conta que as constantes $\left\|A^T r_k\right\|_2, \|r_k\|_2$ e $\|A\|_F$ são economicamente retornadas pelo LSMR (Seção 4.5.4) e pelo LSQR [25]. O que não é o caso para o CG, onde essas constantes tiveram que ser calculadas pelos métodos convencionais, ocasionando um maior custo computacional do algoritmo. De qualquer forma, para uma comparação mais justa, o número de iterações necessárias para o CG retornar uma solução foi anotado e uma nova execução do CG com a mesma base de dados e quantidade de iterações foi realizada, sem ser necessário o cálculo de $\left\|A^T r_k\right\|_2, \|r_k\|_2$ e $\|A\|_F$. Fixamos $ATOL = 10^{-8}$, o mesmo valor utilizado para os testes comparativos de [8].

Algumas métricas resultantes estão na Tabela 5.5. Os valores T_{LSMR}, T_{LSQR} e T_{CG} indicam os tempos de execução do LSMR, LSQR e CG, respectivamente, em cada uma das simulações. O número de iterações em todas as simulações foi igual nos três algoritmos, por esse motivo k foi representado apenas uma vez. O EQM de $\hat{\beta}$ final dos algoritmos foi omitido, já que foram iguais na precisão de 7 casas decimais em todas as

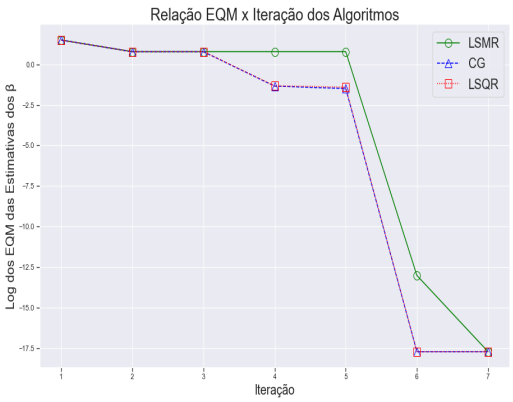
simulações. Entretanto, as relações entre os EQM dos $\hat{\beta}$ dos algoritmos e as iterações estão na Figura 5.1. Observe como os erros nos LSQR e CG caem mais rapidamente que os erros no LSMR, mas são alcançados rapidamente depois. Apesar desse decaimento mais rápido dos outros algoritmos, o tempo para o LSMR retornar uma solução foi relativamente menor em comparação nas seis simulações, como mostra a 5.5, se provando mais vantajoso. Note nas Figuras 5.1 como os erros no LSQR são iguais aos erros do CG em todas as simulações. Isso se deve ao fato de que, matematicamente, o LSQR gera a mesma sequência de aproximações x_k que o CG [3], apesar de ser mais rápido.

n	m	T_{LSMR}	T_{LSQR}	T_{CG}	k
50×10^6	10	7.064	7.076	9.488	7
25×10^6	30	6.509	6.640	9.681	8
10×10^6	50	6.926	7.670	8.305	10
5×10^6	100	6.438	6.475	8.403	11
3×10^6	250	9.120	9.158	10.820	13
10^6	500	7.551	7.950	8.776	15

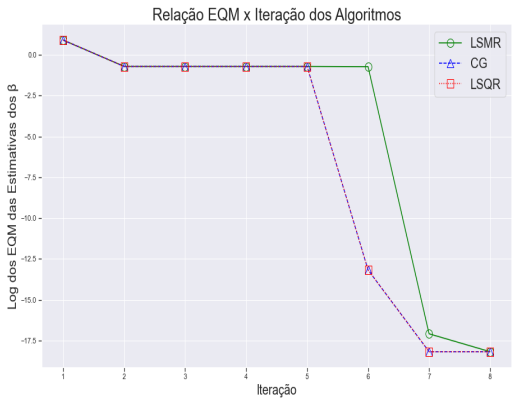
Tabela 5.5: Resultados de seis simulações com diferentes valores de observações n e covariáveis m, comparando três métodos de Krylov: LSMR, LSQR e CG.

Figura 5.1: Gráficos das seis simulações, comparando o EQM dos $\hat{\beta}$ de cada iteração dos algoritmos LSMR, LSQR e CG.

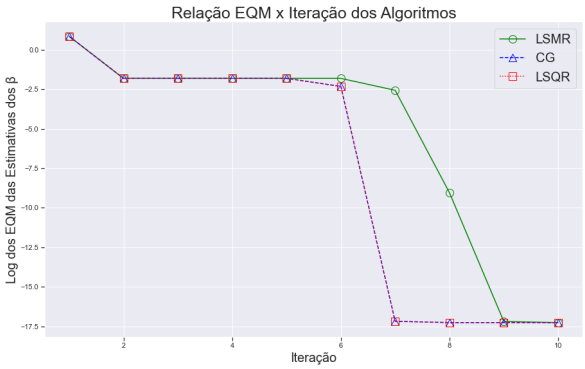
(a) $n = 50 \times 10^6$, $m = 10$



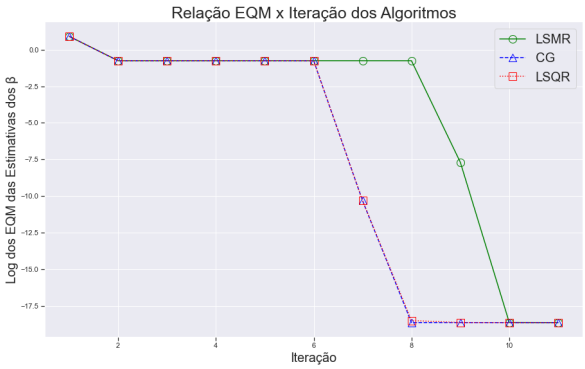
(b) $n = 25 \times 10^6$, $m = 30$



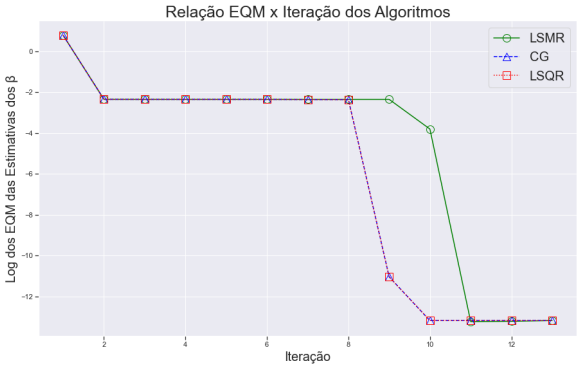
(c) $n = 10 \times 10^6$, $m = 50$



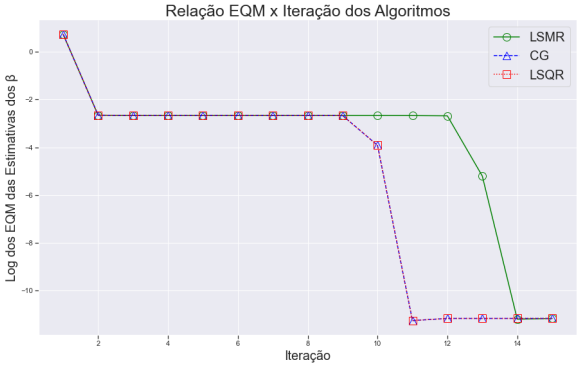
(d) $n = 5 \times 10^6$, $m = 100$



(e) $n = 3 \times 10^6$, $m = 250$



(f) $n = 10^6$, $m = 500$



Fonte: Elaborado pelo autor.

Capítulo 6

Aplicação

Neste capítulo, todos os algoritmos analisados nos estudos de simulações, tanto métodos iterativos quanto diretos, serão novamente analisados, agora em duas bases de dados reais. O objetivo das análises é compará-los na prática, sem ambiente controlado inato das simulações. Como o foco é a comparação dos métodos, não foi realizado um preparo dos dados para as análises. Os estudos foram realizados no Python, no ambiente Jupyter, usando as bibliotecas Numpy e Pandas [26, 42], a última sendo essencial para a importação das bases de dados.

6.1 Base de dados: Timbres Musicais

Nesta seção, os algoritmos são aplicados a um conjunto de dados reais retirados do repositório *UC Irvine Machine Learning Repository*[1] chamado *Year Prediction MSD* [2]. O conjunto de dados possui 515345 exemplos de músicas, 90 covariáveis, que caracterizam o timbre da música, e uma variável dependente representando o ano que a música foi lançada, variando de 1922 até 2011.

De acordo com a documentação [21] do software de onde as covariáveis foram retiradas, timbre é a qualidade de uma nota ou som musical que distingue diferentes tipos de instrumentos musicais ou vozes. É uma noção complexa que independe do tom e do volume. Cada uma das covariáveis representa uma abstração de característica do timbre. Por exemplo, a primeira covariável representa o volume médio do segmento, a segunda enfatiza o brilho, a terceira está mais intimamente correlacionada com a planicidade de um som, a quarta para sons com um ataque mais forte, etc. No fim, o timbre é melhor descrito como uma combinação linear das covariáveis.

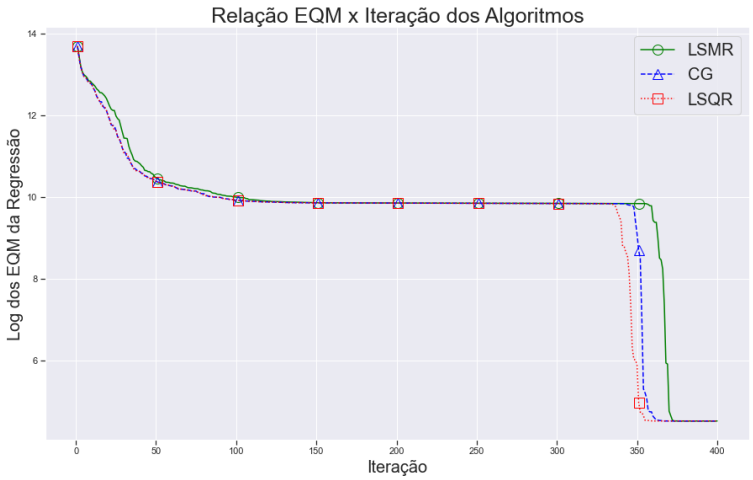
[1] URL: https://archive.ics.uci.edu

Método	EQM	k	T
LSMR	91.1689	400	11.9772
LSQR	91.1689	400	11.8040
CG	91.1689	410	23.5197
QR	91.1689	1	4.1921

Tabela 6.1: Comparação entre os algoritmos iterativos LSMR, LSQR e CG, junto com o método direto da decomposição QR, aplicados na análise de regressão linear da primeira base de dados real.

Os resultados das análises de regressão linear estão na Tabela 6.1, onde o EQM representa o erro quadrático médio dos preditores, k simboliza o número de iterações e T representa o tempo de execução. Usamos $ATOL = 10^{-8}$ e a mesma condição de parada S2 da Seção 4.5.4 para todos os algoritmos iterativos. Note, curiosamente, como o método direto retornou a solução mais rapidamente que os outros algoritmos, que precisaram de muitas iterações para convergirem. A provável causa é o fato de o número de condicionamento da matriz A[2] da regressão linear ser $\kappa(A) = 70614$, considerado alto. O ajuste do modelo R^2 foi similar em todos os métodos, aproximadamente igual a $R^2 = 0.237$. Além disso, observe como, provavelmente pelo mesmo motivo, o EQM de todos os algoritmos é alto, ou seja, o modelo não conseguiu representar bem os dados. Em relação aos métodos diretos, vê-se então que os métodos iterativos não são necessariamente a melhor escolha para a análise de qualquer tipo de dados.

Figura 6.1: EQM dos preditores obtidos dos método iterativos por iteração.



Fonte: Elaborado pelo autor.

A Figura 6.1 mostra como os EQM do preditor obtido dos algoritmos se comporta

[2]Veja o Apêndice D

com o passar das iterações, quando aplicados a essa base de dados real. Note que, apesar de um número superior de iterações, o comportamento do gráfico é semelhante ao gráfico das Figuras 5.1, onde os mesmos algoritmos foram testados em bases de dados artificiais.

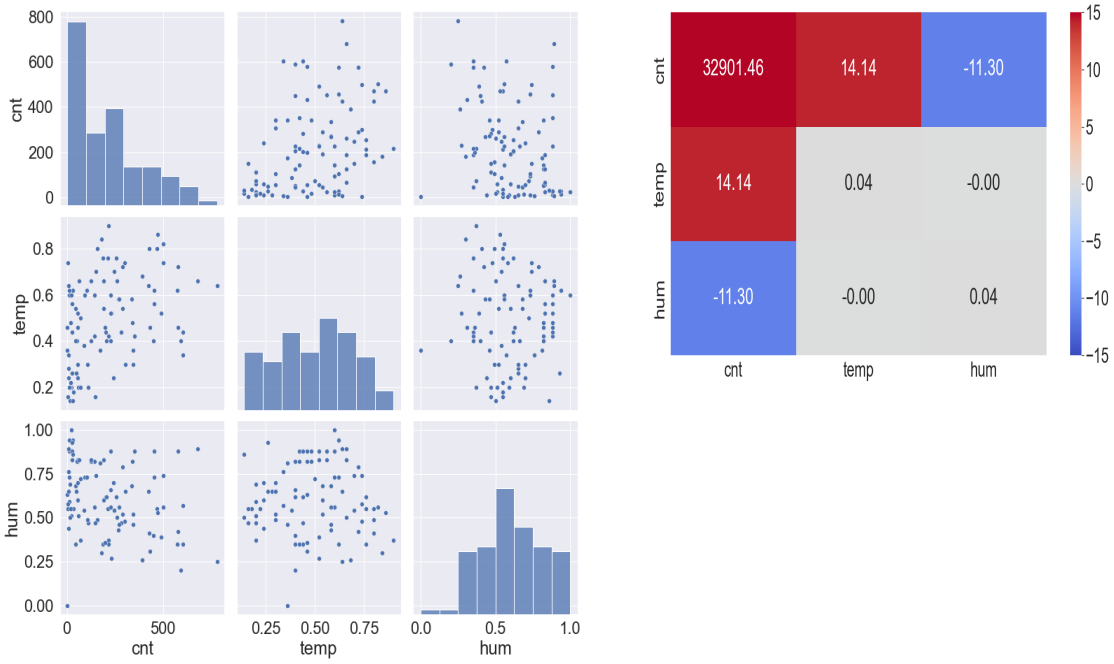
6.2 Base de dados: Bicicletas Compartilhadas

Esta seção foi inspirada no artigo de Jun, Lee e Ryu [22], em que os autores aplicaram uma abordagem dividir e conquistar na análise de regressão linear da base de dados que será tratada nesta seção. A base de dados também é proveniente do repositório *UC Irvine Machine Learning Repository* e se baseia em um sistema de aluguel de bicicletas [7]. Os dados são constituídos por 17389 observações, duas covariáveis (temperatura e umidade) e uma variável dependente, que representa o número total de bicicletas alugadas. A Figura 6.2 contém uma breve análise exploratória dessa base de dados.

Figura 6.2: Uma simples análise exploratória da base de dados das bicicletas compartilhadas.

(a) Gráficos de dispersão de 100 observações, relacionando as variáveis.

(b) Mapa de calor da matriz de covariância.



Fonte: Elaborado pelo autor.

Como na aplicação do banco de dados dos timbres musicais, a comparação do LSMR com os outros algoritmos é feita comparando-se as mesmas métricas tratadas na Tabela 6.1. O objetivo aqui é avaliar o comportamento desses algoritmos em uma base

de dados menor e com número de condicionamento $\kappa(A) = 11$, considerado pequeno. O coeficiente de determinação é $R^2 = 0.251$, próximo ao da base de dados da seção anterior.

Método	EQM	k	T
LSMR	24639	3	0.00096
LSQR	24639	3	0.00090
CG	24639	3	0.00122
QR	24639	1	0.00596

Tabela 6.2: Comparação entre os algoritmos iterativos LSMR, LSQR e CG, junto com o método direto da decomposição QR, aplicados na análise de regressão linear da segunda base de dados real.

Os resultados estão na Tabela 6.2. As estimativas $\hat{\beta}_0$, $\hat{\beta}_1$ e $\hat{\beta}_2$ foram bastante similares para todos os métodos, em torno de $\hat{\beta}_0 = 184.2446$, $\hat{\beta}_1 = 361.8051$ e $\hat{\beta}_2 = -278.3578$. Veja como esses resultados foram iguais aos obtidos por Jun, Lee e Ryu em [22]. Observe as diferenças nos tempos de execução dos algoritmos aplicados na análise dessa base de dados em comparação com os resultados obtidos na Tabela 6.1. Diferentemente da primeira base de dados, o método direto da decomposição QR teve tempo de execução superior quando comparado aos métodos iterativos, mostrando indícios da influência do número de condicionamento na performance desses algoritmos. Com uma tolerância menor nas condições de parada, os métodos iterativos talvez precisassem de até menos iterações que o método direto, aumentando ainda mais a diferença nos tempos de execução. Note como, mesmo em uma base de dados relativamente pequena, onde o custo computacional da decomposição QR não é alto, os métodos iterativos se mostraram superiores ao método direto. Como poucas iterações foram realizadas, todas seguindo um padrão já observado, o gráfico como o da Figura 6.1 não será apresentado.

Capítulo 7

Conclusões

7.1 Observações Finais

Este trabalho define e dá exemplos de métodos de Krylov, uma categoria de algoritmos iterativos usados principalmente para se encontrar a solução do problema dos mínimos quadrados, aplicando-se esses métodos na análise de regressão linear de dados considerados de grande volume, definidos como Big Data. Com foco no algoritmo LSMR, essa proposta leva em consideração a performance desses algoritmos em relação a métricas relevantes na prática, como o tempo de execução do algoritmo e o erro quadrático médio da regressão obtida através das soluções dos algoritmos.

Com o objetivo de avaliar as performances dos algoritmos, o presente trabalho apresenta estudos de simulação, gerados no Python através da biblioteca Numpy. Primeiro, estuda-se como o LSMR se compara com o método direto da decomposição QR, usado no problema dos mínimos quadrados, quando se varia a dimensionalidade da base de dados e os parâmetros da regressão. Nota-se que a performance do LSMR é, no geral, melhor quando se leva em conta o tempo de execução do algoritmo, apesar da precisão dos dois algoritmos ser semelhante quanto ao EQM dos parâmetros obtidos. Da mesma forma, compara-se o LSMR com outros dois métodos de Krylov, o LSQR e o método dos Gradientes Conjugados. Novamente, varia-se a dimensionalidade da base de dados e os parâmetros. Percebe-se uma leve vantagem do LSMR em relação aos outros algoritmos quanto ao número de iterações para a convergência e, consequentemente, uma vantagem em relação ao tempo de execução.

Analisa-se também dois conjuntos de dados reais, considerados Big Data, onde compara-se todos os métodos citados anteriormente. O primeiro banco de dados contém informações sobre o timbre de diversos exemplos de músicas de anos variados. Curiosamente, o método direto da decomposição QR, computacionalmente mais exigente que os métodos de Krylov, apresentou melhores métricas na análise final. A provável causa disso é o fato de o número de condicionamento da matriz de regressão desse banco de dados ser considerado alto, tornando o problema mal condicionado. Para comparar os algoritmos

em uma base de dados melhor condicionada, foi-se usada uma nova base de dados, que contém dados sobre compartilhamento de bicicletas. Nesse caso, os testes indicaram uma superioridade dos métodos de Krylov em relação ao método direto, apesar de ser uma base de dados menos volumosa em comparação à última.

Conclui-se então a partir dos estudos simulados e aplicações que os métodos de Krylov tiveram um desempenho melhor que os métodos diretos quando aplicados em problemas de mínimos quadrados bem condicionados e de grande porte, permitindo uma espera menor para a obtenção de uma solução satisfatória.

7.2 Tópicos para Trabalhos Futuros

Apesar dos resultados serem promissores, os métodos de Krylov não podem ser aplicados em qualquer problema de mínimos quadrados. Sugere-se como trabalho futuro a pesquisa em formas de contornar o mal condicionamento de matrizes e um estudo de como os pré-condicionadores influenciam nas métricas estudadas neste trabalho. Além disso, sugere-se um estudo aprofundado em relação à complexidade de tempo dos algoritmos vistos neste texto, tanto os métodos diretos quanto iterativos, a fim de se encontrar maneiras de acelerar a convergência dos algoritmos.

Referências

[1] Robert Gardner Bartle and Donald R. Sherbert. *Introduction To Real Analysis*. Wiley, 4th ed edition. OCLC: ocn671573454.

[2] T. Bertin-Mahieux. Year Prediction MSD. UCI Machine Learning Repository, 2011. DOI: https://doi.org/10.24432/C50K61.

[3] Åke Björck. *Numerical Methods For Least Squares Problems*. SIAM, 1996.

[4] Peter N. Brown and Youcef Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, 1990.

[5] X.-W. Chang, C. C. Paige, and D. Titley-Peloquin. Stopping criteria for the iterative solution of linear least squares problems. 31(2):831–852.

[6] David A. Cox, John B. Little, and Donal O'Shea. *Using Algebraic Geometry*. Number 185 in Graduate texts in mathematics. Springer, 1998.

[7] Hadi Fanaee-T. Bike Sharing. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[8] David Chin-Lung Fong and Michael Saunders. LSMR: An iterative algorithm for sparse least-squares problems. 33(5):2950–2971.

[9] John B. Fraleigh and Victor J. Katz. *A First Course in Abstract Algebra*. Addison-Wesley, 7th ed edition.

[10] Ricardo AS Frantz, J-Ch Loiseau, and J-Ch Robinet. Krylov methods for large-scale dynamical systems: Application in fluid dynamics. *Applied Mechanics Reviews*, 75(3):030802, 2023.

[11] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. 2(2):205–224.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, fourth edition edition, 2013. OCLC: 824733531.

[13] Martin H Gutknecht. A brief introduction to krylov space methods for solving linear systems. In *Frontiers of Computational Science: Proceedings of the International Symposium on Frontiers of Computational Science 2005*, pages 53–62. Springer, 2007.

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[15] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[16] Nicholas J. Higham. *Accuracy and stability of numerical algorithms*. Number 80 in Other titles in applied mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2nd ed edition.

[17] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. 5(4):339–342.

[18] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[19] Suk-Geun Hwang. Cauchy's interlace theorem for eigenvalues of hermitian matrices. 111(2):157–159.

[20] Ilse C.F. Ipsen and Carl D. Meyer. The idea behind krylov methods. *The American Mathematical Monthly*, 1998.

[21] T. Jehan and D. DesRoches. Analyzer documentation - model AI assignments. http://modelai.gettysburg.edu/2012/music/docs/EchoNestAnalyzeDocumentation.pdf, 2011. [Accessed 18-02-2025].

[22] Sunghae Jun, Seung-Joo Lee, and Jea-Bok Ryu. A divided regression analysis for big data. 9(5):21–32.

[23] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, 2011.

[24] C. C. Paige. Bidiagonalization of matrices and solution of linear equations. 11(1):197–209.

[25] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. 8(1):43–71.

[26] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[27] Steven Roman. *Advanced Linear Algebra*. Springer, third edition edition, 2010. OCLC: 1413816834.

[28] David S. Watkins. Fundamentals of matrix computations. 35(3):520–521.

[29] Y. Saad. *Iterative Methods For Sparse Linear Systems*. SIAM, 2nd ed edition, 2003.

[30] Youcef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[31] Michael Sipser. *Introduction To The Theory Of Computation*. Cengage Learning, third edition edition. OCLC: 1194477366.

[32] Steven S. Skiena. *Sorting and Searching*, pages 103–144. Springer London, London, 2008.

[33] G. W. Stewart. The QLP approximation to the singular value decomposition. 20(4):1336–1348.

[34] G. W. Stewart. *Matrix Algorithms*. Society for Industrial and Applied Mathematics, 1998.

[35] G. W. Stewart. Research, development and LINPACK. In J. R. Rice, editor. *Mathematical Software III*, pages 1-14. Academic Press, New York, 1977.

[36] I Putu Alit Sudrastawa, Agus Parwata, Made Wisnu Adhi Saputra I Gusti Agung Made Wirautama, and I Wayan Septa Malan Vergantana. Conceptual and practical review of gaussian elimination and gauss-jordan reduction. *Jurnal Ilmu Komputer Indonesia (JIK)*, 7, 2022.

[37] Terence Tao. *Topics in Random Matrix Theory*. Number vol. 132 in Graduate studies in mathematics. American mathematical society.

[38] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[39] Felipe Vieira and Rafael Aleixo. *Métodos Iterativos para Problemas de Quadrados Mínimos Lineares*. SBMAC, 2022.

[40] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J

Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[41] Henk A. van der Vorst. *Iterative Krylov Methods For Large Linear Systems*. Number 13 in Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2003.

[42] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[43] Virginia Vassilevska Williams. Breaking the coppersmith-winograd barrier. *Stanford University*, 07 2014.

[44] Xin Yan and Xiaogang Su. *Linear Regression Analysis: Theory And Computing*. World Scientific Pub. Co., 2009. OCLC: 613658550.

[45] Igor Rostislavovič Šafarevič and Alexey O. Remizov. *Linear Algebra And Geometry*. Springer, 2013. p. 56.

Apêndice A

Decomposição QR

Muitas vezes estamos interessados nos espaços coluna de uma matriz $A \in \mathbb{R}^{n\times m}$, ou seja, nos sucessivos subespaços gerados pelas colunas a_1, a_2, ..., a_m de A:

$$\langle a_1\rangle \subseteq \langle a_1, a_2\rangle \subseteq \cdots \subseteq \langle a_1, a_2, \ldots, a_m\rangle.$$

A decomposição (ou fatoração QR) se baseia na construção de uma sequência de vetores ortonormais que geram esses subespaços sucessivamente.

A.1 Processo de Gram-Schmidt

O Processo de Ortogonalização de Gram-Schmidt (PGS) é um processo clássico usado na ortogonalização de um conjunto qualquer de vetores. Seja $S_i := \{a_1, a_2, \ldots, a_i\}$, $i = 1, 2, \ldots, m$, a sequência de conjuntos formados pelos i primeiros vetores do conjunto $S := \{a_1, a_2, \ldots, a_m\}$, onde $a_i \in \mathbb{R}^n$.

O processo é realizado da seguinte forma: no i-ésimo passo, buscamos encontrar o vetor unitário $q_i \in \langle S_i\rangle$ tal que $q_i \perp q_j$ para $j = 1, \ldots, i-1$. Dessa forma, podemos começar definindo:

$$u_1 := a_1 \quad \text{e} \quad q_1 := \frac{u_1}{r_{11}}.$$

Obviamente, $q_1 \in \langle a_1\rangle$. Para que q_1 seja unitário, basta tomar $r_{11} = \|a_1\|$. Fazendo

$$u_2 := a_2 - r_{12}q_1,$$

temos que $u_2 \in \langle a_1, a_2\rangle$, já que $u_2 := a_2 - r_{12}r_{11}a_1$. Além disso,

$$\begin{aligned}\langle q_1, u_2\rangle &= \langle q_1, a_2 - r_{12}q_1\rangle \\ &= \langle q_1, a_2\rangle - r_{12}\langle q_1, q_1\rangle \\ &= \langle q_1, a_2\rangle - r_{12}.\end{aligned}$$

Para que u_2 seja ortogonal a q_1, devemos ter $\langle q_1, u_2\rangle = 0$. Dessa forma, a partir das equações acima, basta fazer $r_{12} = \langle q_1, a_2\rangle$. A partir dessa definição, temos então que u_2 é

ortogonal a q_1, mas não é unitário. A fim de satisfazer essa condição, podemos dividir u_2 por sua norma e obter um novo vetor q_2, ou seja

$$q_2 := \frac{u_2}{\|u_2\|}.$$

É óbvio que q_2 é ortogonal a q_1.

Vamos definir

$$u_i := a_i - r_{1i}q_1 - r_{2i}q_2 - \cdots - r_{i-1,i}q_{i-1}. \tag{A.1}$$

Juntando as informações acima, podemos assumir que q_i tem a forma

$$q_i := \frac{u_i}{r_{ii}}, \tag{A.2}$$

onde temos

$$r_{ji} = \langle q_j, a_i \rangle \text{ , se } j < i, \tag{A.3}$$

e os coeficientes r_{ii} são escolhidos para normalização:

$$|r_{ii}| = \|u_i\|_2 \text{ , se } j = i. \tag{A.4}$$

Note que o sinal da constante de normalização r_{ii} é arbitrário. Por conveniência, escolhemos $r_{ii} > 0$.

Vamos provar usando o Princípio Da Indução Forte (ou Segundo Princípio Da Indução) [1] que a definição recursiva (A.2), junto com (A.3) e (A.4), dos vetores q_i satisfaz as condições:

1. $q_i \perp q_j$, para $i \neq j$.
2. $\|q_i\| = 1$, para todo $i \leq m$.
3. $q_i \in \langle S_i \rangle$, para todo $i \leq m$.

É fácil ver que, a partir de (A.2) e (A.4), a regra (2) é satisfeita. Para as regras (1) e (3), o caso base da indução, em que $i = 1$, já foi provado. Dessa forma, vamos supor que essas regras valem para todo q_l tal que $1 \leq l \leq k$. Definindo q_{k+1} como em (A.2), temos que

$$\begin{aligned}
\langle q_{k+1}, q_l \rangle &= \frac{1}{r_{k+1,k+1}} \left\langle a_{k+1} - \sum_{j=1}^{k} r_{j,k+1} q_j, q_l \right\rangle \\
&= \frac{1}{r_{k+1,k+1}} \left(\langle a_{k+1}, q_l \rangle - \sum_{j=1}^{k} r_{j,k+1} \langle q_j, q_l \rangle \right).
\end{aligned}$$

Pela hipótese de indução, $\langle q_j, q_l \rangle = 0$, para todo $l \neq j$ e $1 \leq l \leq k$. Dessa forma

$$\begin{aligned}
\langle q_{k+1}, q_l \rangle &= \frac{1}{r_{k+1,k+1}} \left(\langle a_{k+1}, q_l \rangle - \sum_{j=1}^{k} r_{j,k+1} \langle q_j, q_l \rangle \right) \\
&= \frac{1}{r_{k+1,k+1}} \left(\langle a_{k+1}, q_l \rangle - r_{l,k+1} \langle q_l, q_l \rangle \right) \\
&= \frac{1}{r_{k+1,k+1}} \left(\langle a_{k+1}, q_l \rangle - r_{l,k+1} \right) \\
&= \frac{1}{r_{k+1,k+1}} \left(\langle a_{k+1}, q_l \rangle - \langle q_l, a_{k+1} \rangle \right) \\
&= 0.
\end{aligned}$$

Assim, temos que $q_{k+1} \perp q_l$, para $1 \leq l \leq k$. A indução matemática confirma então a validade dessa condição para todo $i \leq m$.

Quanto à condição (3), basta ver que, pela hipótese de indução, q_l é combinação linear de $a_1, a_2, \ldots, a_l$, quando $1 \leq l \leq k$. Dessa forma, como q_{k+1} é combinação linear de $q_1, q_2, \ldots, q_k$, então q_{k+1} também é combinação linear de $a_1, a_2, \ldots, a_k, a_{k+1}$. Assim, $q_{k+1} \in \langle S_{k+1} \rangle$. Novamente, pela indução matemática, a condição vale para todo $i \leq m$.

Provamos então que, a partir das definições (A.2), (A.3) e (A.4), o Processo de Gram-Schmidt, em m iterações, produz uma sequência de vetores ortonormais q_i que geram o mesmo subespaço que o conjunto S.[1]

A.1.1 PGS em Conjunto Linearmente Dependente

Apesar de funcionar para qualquer tipo de conjunto enumerável de vetores, precisamos fazer algumas considerações especiais quando o conjunto é linearmente dependente.

Primeiro, vamos definir $S_i := \{a_1, a_2, \ldots, a_i\}$. Agora, suponha que $j \leq m$ é o menor inteiro i tal que a_i é combinação linear dos outros vetores de S_i, ou seja, $a_i \in \langle S_i \rangle$. Afirmamos que $u_j = 0$ em A.1.

Com efeito, sabemos que o vetor

$$v_j := \sum_{l=1}^{j-1} r_{l,j} q_l = \sum_{l=1}^{j-1} \langle q_l, a_j \rangle q_l$$

é a projeção ortogonal de a_j no subespaço $\mathcal{O}$ gerado pelos vetores $q_1, q_2, \ldots, q_{j-1}$.
A condição (3) implica que $\mathcal{O} = \langle S_{j-1} \rangle$ e, como $a_j \in \langle S_{j-1} \rangle$, então $a_j \in \mathcal{O}$. Dessa forma, a projeção de a_j em $\mathcal{O}$ é o próprio a_j. Assim, $v_j = a_j$, fazendo com que $u_j = 0$.

[1] No geral, esse procedimento funciona para qualquer conjunto enumerável de vetores.

O vetor 0 não pode ser normalizado. Devemos então descartá-lo para continuar o PGS. Observe que, se $a_j \in \langle S_{j-1} \rangle$, então $\mathcal{O} = \langle S_j \rangle$.

A.2 Decomposição QR Reduzida

Vimos na seção anterior que, se $S := \{a_1, a_2, \ldots, a_m\}$ é um conjunto de vetores de tamanho n linearmente independentes, então o PGS gera uma sequência de vetores ortonormais q_i, $i = 1, \ldots, m$, através das equações

$$\begin{aligned} a_1 &= r_{11} q_1 \\ a_2 &= r_{12} q_1 + r_{22} q_2 \\ a_3 &= r_{13} q_1 + r_{23} q_2 + r_{33} q_3 \\ &\vdots \\ a_m &= r_{1m} q_1 + r_{2m} q_2 + \cdots + r_{mm} q_m. \end{aligned}$$

Em formato matricial, temos

$$A = \hat{Q}\hat{R} \tag{A.5}$$

onde os vetores de S formam as colunas de $A \in \mathbb{R}^{n \times m}$, $\hat{Q} \in \mathbb{R}^{n \times m}$ é uma matriz com as colunas ortonormais e $\hat{R} \in \mathbb{R}^{m \times m}$ é uma matriz triangular superior. Ou seja,

$$\left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \ldots & a_m \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \ldots & q_m \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1m} \\ & r_{22} & \ldots & \vdots \\ & & \ddots & \vdots \\ & & & r_{mm} \end{bmatrix}.$$

Essa decomposição é chamada de *decomposição QR reduzida* de A.

A.3 Decomposição QR Completa

Podemos transformar a matriz $\hat{Q}$ (Equação A.5) em uma matriz ortogonal adicionando $n - m$ vetores colunas ortonormais à matriz $\hat{Q}$. Caso o conjunto S seja linearmente

dependente, podemos, durante o PGS, substituir os vetores $u_j = 0$ em A.1 por um vetor ortonormal em relação ao conjunto S_j, definido naquela seção. Um método para se gerar esses vetores ortonormais é calculando a solução do sistema $\tilde{Q}^T q = 0$, onde as colunas de $\tilde{Q}$ são formadas pelos vetores ortonormais q_i já calculados.

Após esse processo, vamos ter então uma nova matriz ortogonal $Q \in \mathbb{R}^{n\times n}$. Para balancear a Equação A.5, adicionamos $n - m$ linhas compostas por zeros à matriz $\hat{R}$ e chamamos a nova matriz de $R \in \mathbb{R}^{n\times m}$. Assim, a Equação A.5 se torna

$$A = QR \tag{A.6}$$

ou seja

$$\left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_m \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \dots & q_n \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ & r_{22} & \dots & \vdots \\ \vdots & & \ddots & \vdots \\ & & & r_{mm} \\ 0 & 0 & \dots & 0 \\ & & \vdots & \end{bmatrix}.$$

É fácil provar usando o PGS que toda matriz real A tem uma decomposição QR reduzida, logo possui uma completa também. Além disso, caso a matriz A tenha posto completo, a decomposição QR reduzida A.5 é única se os elementos da diagonal de $\hat{R}$ são positivos, ou seja, $r_{ii} > 0$ para $i = 1, \dots, m$. Para mais detalhes, veja [38, Seção 7].

A.4 Refletores de Householder

Apesar de teoricamente útil, o PGS não é numericamente estável para a construção da decomposição QR de uma matriz A [12]. Em vista desses problemas, Householder publicou em 1958 um artigo que introduziu o método dos *refletores de Householder* [17], mais numericamente estável que o PGS.

O método de Householder se baseia na multiplicação pela esquerda de uma matriz $A \in \mathbb{R}^{n\times m}$ por matrizes ortogonais $Q_i \in \mathbb{R}^{n\times n}$, $i = 1, \dots, m$, a fim de se obter uma matriz triangular $R \in \mathbb{R}^{n\times m}$. Matematicamente,

$$Q_n Q_{n-1} \dots Q_1 A = R. \tag{A.7}$$

O produto $Q^T := Q_m Q_{m-1} \dots Q_1$ é uma matriz ortogonal e, a partir dela, temos a decomposição QR

$$A = QR$$

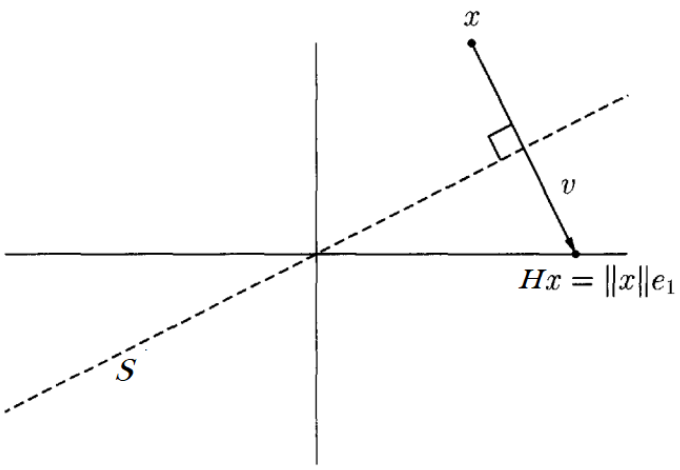
Cada matriz Q_i será construída de forma que os últimos $n-i$ elementos da i-ésima coluna de A se tornem zeros. Por exemplo, para uma matriz 4×3, buscamos o seguinte processo

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1 A}{\begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2 Q_1 A}{\begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} \\ 0 & 0 & \mathbf{x} \end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3 Q_2 Q_1 A}{\begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} \\ 0 & 0 & 0 \end{bmatrix}}. \tag{A.8}$$

A ideia dos refletores de Householder é obter a reflexão de um vetor em relação a um hiperplano[2](Veja a Figura A.1). Definindo como em [27], sendo V um espaço vetorial, para um $v \in V, \|v\| = 1$, um refletor de Householder[3] é uma matriz H_v tal que

$$H_v v = -v \quad \text{e} \quad H_v w = w \quad \text{para } w \in \langle v \rangle^{\perp}$$

Figura A.1: Transformação de Householder do vetor x em respeito ao subespaço $S := \langle v \rangle^{\perp}$.



Fonte: Figura retirada de [38], levemente alterada.

É uma prova direta que $H_v = I - 2vv^t$. Para ver isso, note que para todo $x \in V$, $x = \alpha v + w$, aonde $\alpha \in \mathbb{R}$ e $w \in \langle v \rangle^{\perp}$. Além disso

$$\begin{aligned} 0 &= \langle w, v \rangle \\ &= \langle x - \alpha v, v \rangle \end{aligned}$$

[2]Lembre que um hiperplano em um espaço vetorial m-dimensional é um subespaço $(m-1)$-dimensional. Note também que todo vetor $v \in V$ define um hiperplano $S := \langle v \rangle^{\perp}$

[3]Também chamada de transformação de Householder ou matriz de Householder.

$$\begin{aligned} &= \langle x, v\rangle - \alpha\langle v, v\rangle \\ &= \langle x, v\rangle - \alpha \\ \implies \alpha &= \langle x, v\rangle. \end{aligned}$$

Dessa forma

$$\begin{aligned} H_v x &= H_v(\alpha v + w) \\ &= \alpha H_v v + H_v w \\ &= -\alpha v + w \\ &= (x - \langle x, v\rangle v) - \langle x, v\rangle v \\ &= x - 2\langle x, v\rangle v \\ &= x - 2vv^t x \\ &= (I - 2vv^t)x. \end{aligned}$$

Logo, concluímos que

$$H_v = I - 2vv^t.$$

A fim de se obter um processo generalizado do Exemplo A.8, podemos aplicar uma transformação de Householder em cada vetor, que vamos definir como $x_{i:n}$, formado pelos últimos $n - i + 1$ elementos da i-ésima coluna de $Q_{i-1} \dots Q_1 A$ (aonde $Q_0 \coloneqq I$), ou seja

$$x_{i:n} \coloneqq \begin{bmatrix} x_{ii} \\ x_{(i+1)i} \\ \vdots \\ x_{ni} \end{bmatrix} \xrightarrow{H_v} H_v x_{i:n} = \begin{bmatrix} \mathbf{x} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A.9}$$

Podemos ver pela Figura A.1 que uma das possíveis reflexões para o processo A.9 é a que transforma o vetor $x_{i:n}$ no vetor αe_i, para algum $\alpha \in \mathbb{R}$. Roman prova em [27] que, se $\|v\| = \|w\|$, $H_{\frac{v-w}{\|v-w\|}}$ é o único refletor de Householder que envia v para w e vice-versa. Dessa forma, temos que H_v com $v = \frac{x_{i:n} - \|x\| e_1}{\left\|x_{i:n} - \|x\| e_1\right\|}$ é uma das possíveis escolhas para resolver A.9.

Podemos agora construir as matrizes Q_i. Elas terão a forma

$$Q_1 \coloneqq H_v \text{ , com } v = \frac{x_{1:n} - \|x\| e_1}{\left\|x_{1:n} - \|x\| e_1\right\|} \quad \text{e} \quad Q_i \coloneqq \begin{bmatrix} I_{i-1} & \mathbf{0} \\ \mathbf{0} & H_v \end{bmatrix} \text{ , com } v = \frac{x_{i:n} - \|x\| e_1}{\left\|x_{i:n} - \|x\| e_1\right\|}.$$

Temos que cada Q_i é uma matriz ortogonal. Assim, após n multiplicações o processo terá a forma A.7. Como a diagonal de R é positiva, pela unicidade da decomposição QR, vamos ter no fim a mesma decomposição que a obtida no PGS.

Apêndice B

Polinômios Mínimo e Característico de uma matriz

Este apêndice foi baseado em [27].

B.1 Polinômio Mínimo

Um *polinômio de grau n de uma variável*, $n \in \mathbb{N} \cup \{0\}$, sempre pode ser escrito na forma

$$P(x) = \sum_{k=0}^{n} a_k x^k = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x_1 + a_0,$$

onde x é a variável e a_k são os coeficientes. Podemos definir um tipo específico de polinômio de uma variável, o polinômio **mônico**.

Definição B.1.1 ([9]). *O Polinômio Mônico é um polinômio $P(x)$ de grau n uma variável onde $P(x) \neq 0$ e $a_n = 1$.*

Como exemplo, os polinômios $P_1(x) = x$, $P_2(x) = 1$ e $P_3(x) = x^3 - 2x + 3$ são polinômios mônicos.

Por definição, o **Polinômio Mínimo** P_A de uma matriz $A \in \mathbb{R}^{m \times m}$ é o polinômio mônico de menor grau P_n tal que $P_n(A) = 0$. Ou seja, usando a matriz A como variável, temos que $P_n(A) = a_n A^n + a_{n-1} A^{n-1} + \cdots + a_1 A_1 + a_0 I_m$. Tomando $A = 2I_m$ como exemplo, vamos ter que $P_1(x) = x - 2$ é o polinômio mínimo de A, já que $P_1(A) = 2I_m - 2I_m = 0$ e não existe um polinômio de grau 0 que satisfaça essa equação.

B.2 Polinômio Característico

Antes de apresentar um teorema essencial da Álgebra Linear, vamos fazer a seguinte definição. Seja $A \in \mathbb{R}^{n\times n}$, então

Definição B.2.1. *O Polinômio Característico Q_A de A é o polinômio definido pela equação $Q_A(x) = det(xI - A)$, onde $det(xI - A)$ é o determinante da matriz $xI - A$ e x é um escalar.*

Agora podemos enunciar o teorema.

Teorema B.2.1 (Teorema de Cayley-Hamilton, [27])**.** *O polinômio mínimo de A, P_A, divide o polinômio característico de A, Q_A. Ou seja, a divisão $P_A \mid Q_A$ tem resto igual a 0. Dessa forma, $Q_A(A) = 0$.*

Note que o Teorema de Cayley-Hamilton garante a existência do polinômio mínimo.

Os dois polinômios P_A e Q_A possuem as mesmas raízes [27], que são os chamados autovalores λ_i de A. A multiplicidade de cada autovalor λ_i em P_A é o menor inteiro m tal que

$$ker((A - \lambda_i I)^l) = ker((A - \lambda_i I)^{l+1}),\ \forall l \geq m_i$$

ou seja, o núcleo da matriz $(A - \lambda_i I)^l$ é o mesmo para expoentes maiores que m.

Sabemos que

$$\begin{aligned} Q_A(x) &= \prod_{i=1}^{m}(x - \lambda_i)^{m_i} \\ &= \sum_{i=0}^{m} a_i x^i \\ \implies a_0 &= \prod_{i=1}^{m}(-\lambda_i)^{m_i}, \end{aligned}$$

onde a_i são os coeficientes do polinômio Q_A, m_i é a multiplicidade do autovalor λ_i e $m = \sum_i m_i$. Logo, como A é não singular, $a_0 \neq 0$. Dessa forma

$$\begin{aligned} Q_A(A) &= \sum_{i=0}^{m} a_i A^i = 0 \\ \implies a_0 I &= -\sum_{i=1}^{m} a_i A^i \\ \implies A^{-1} &= -\frac{1}{a_0}\sum_{i=1}^{m} a_i A^{i-1}. \end{aligned}$$

Dessa forma, a matriz inversa A^{-1} pode ser escrita como combinação linear das potências de A.

Apêndice C

Prova do Teorema 3.3.1

O seguinte lema será necessário na demonstração do Teorema 3.3.1:

Lema C.0.1 ([39, Página 98]). *Sejam $R \in \mathbb{R}^{n\times n}$ e $S \in \mathbb{R}^{n\times n}$ matrizes triangulares superiores com diagonal não nula e $W = \begin{bmatrix} 0 \\ S \end{bmatrix} \in \mathbb{R}^{(n+1)\times n}$. Então a matriz $H = RW$ é uma matriz de Hessenberg superior não reduzida.*

Demonstração. Sejam $R = \begin{bmatrix} r_1 & r_2 & \dots & r_n \end{bmatrix}$, $w = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$ e $S = \begin{bmatrix} s_1 & s_2 & \dots & s_n \end{bmatrix}$. Temos que

$$\begin{aligned} RW &= \begin{bmatrix} Rw_1 & Rw_2 & \dots & Rw_n \end{bmatrix} \\ &= \begin{bmatrix} R(0 \quad s_1)^T & R(0 \quad s_2)^T & \dots & R(0 \quad s_n)^T \end{bmatrix} \\ &= \begin{bmatrix} 0r_1 + s_{11}r_2 + \cdots + s_{(n-1)1}r_n & \dots & 0r_1 + s_{1(n-1)}r_2 + \cdots + s_{(n-1)(n-1)}r_n \end{bmatrix} \\ &= \begin{bmatrix} s_{11}r_2 + \cdots + s_{(n-1)1}r_n & \dots & s_{1(n-1)}r_2 + \cdots + s_{(n-1)(n-1)}r_n \end{bmatrix} \\ &= \begin{bmatrix} s_{11}r_2 & s_{12}r_2 + s_{22}r_3 & \dots & s_{1(n-1)}r_2 + \cdots + s_{(n-1)(n-1)}r_n \end{bmatrix} \\ &= H. \end{aligned}$$

Obviamente H é uma matriz de Hessenberg superior e $h_{i+1,i} = s_{ii}r_{i+1,i+1}$, provando que também é não reduzida. □

Agora, podemos provar o teorema:

Demonstração do Teorema 3.3.1. Se particionarmos 3.4 como

$$\begin{bmatrix} K_k & A^k b \end{bmatrix} = \begin{bmatrix} Q_k & q_{k+1} \end{bmatrix} \begin{bmatrix} R_k & r_{k+1} \\ 0 & \rho_{k+1,k+1} \end{bmatrix},$$

vemos que $K_k = Q_k R_k$ é a decomposição QR de K_k. Seja $S_k = R_k^{-1}$. Então

$$\begin{aligned} AQ_k &= AK_kS_k \\ &= \begin{bmatrix} Ab & A^2b & \dots & A^kb \end{bmatrix} S_k \end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} b & Ab & \dots & A^k b \end{bmatrix} \begin{bmatrix} 0 \\ S_k \end{bmatrix} \\
&= K_{k+1} \begin{bmatrix} 0 \\ S_k \end{bmatrix} \\
&= Q_{k+1} R_{k+1} \begin{bmatrix} 0 \\ S_k \end{bmatrix} \\
&= Q_{k+1} \hat{H}_k.
\end{aligned}
$$

onde

$$
\hat{H}_k = R_{k+1} \begin{bmatrix} 0 \\ S_k \end{bmatrix}.
$$

Pelo Lema C.0.1, $\hat{H}_k$ é uma matriz de Hessenberg superior não reduzida.

Reciprocamente, suponha que a matriz ortonormal Q_{k+1} satisfaz (3.5), onde $\hat{H}_k$ é uma matriz $(k+1) \times k$ não reduzida de Hessenberg. Se $k = 1$, então $Aq_1 = h_{11}q_1 + h_{21}q_2$. Como $h_{21} \neq 0$, q_2 é combinação linear de Aq_1 e q_1 e é ortogonal a q_1, então $\begin{bmatrix} q_1 & q_2 \end{bmatrix}$ é o fator-Q de K_2.

Assuma indutivamente que Q_k é o fator-Q de K_k. Se particionarmos

$$
\hat{H}_k = \begin{bmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{bmatrix},
$$

de (3.5) obtemos

$$
Aq_k = Q_k h_k + h_{k+1,k} q_{k+1}. \tag{C.1}
$$

Logo, q_{k+1} é uma combinação linear de Aq_k e as colunas de Q_K, pertencendo ao subespaço $\mathcal{K}_{k+1}(A, b)$. Dessa forma, Q_{k+1} é o fator-Q de K_{k+1}, finalizando a prova. □

Apêndice D

Normas

Neste apêndice revisaremos o conceito de norma, tanto de vetores quanto de matrizes. Após isso, vamos introduzir o importante número de condicionamento de uma matriz.

D.1 Norma de Vetores

Vamos começar definindo o que é norma:

Definição D.1.1 ([39]). *Uma norma em $\mathbb{R}^n$ é uma função real $\|\cdot\|$ que satisfaz as seguintes propriedades:*

N1. $\|x\| \geq 0$ *para todo* $x \in \mathbb{R}^n$, *e* $\|x\| = 0$ *se e somente se* $x = 0$.
N2. $\|\lambda x\| = |\lambda| \cdot \|x\|$, *para todos* $x \in \mathbb{R}^n$ *e* $\lambda \in \mathbb{R}$.
N3. $\|x + y\| \leq \|x\| + \|y\|$, *para todos* $x, y \in \mathbb{R}^n$.

A norma mais conhecida é a função módulo sobre $\mathbb{R}$, mas existem uma infinidade de funções que satisfazem as condições acima. Intuitivamente, as normas transmitem uma noção de tamanho e distância nos espaços vetoriais. É através delas que surgem o conceito de aproximações e convergência.

A classe mais importante de norma de vetores são as chamadas p-normas, que são definidas da seguinte forma:

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}}$$

Dentre as p-normas, as mais usadas são as normas 1,2 e ∞:

$$\|x\|_1 = |x_1| + \cdots + |x_n|,$$
$$\|x\|_2 = (|x_1|^2 + \cdots + |x_n|^2)^{\frac{1}{2}},$$
$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

Por exemplo, se $x = (1, 0, -2), \|x\|_1 = |1|+|0|+|-2| = 3, \|x\|_2 = (|1|^2+|0|^2+|-2|^2)^{\frac{1}{2}} = \sqrt{5}$ e $\|x\|_\infty = \max_{1\leq i\leq n}\{|1|, |0|, |-2|\} = 2$.

A 2-norma é a norma euclidiana, que é a única norma vetorial utilizada neste texto. O teorema a seguir exemplifica o tamanho de sua utilidade:

Teorema D.1.1 ([39]). *Seja $Q \in \mathbb{R}^{n\times n}$ uma matriz ortogonal e $x \in \mathbb{R}^n$. Então*

$$\|Qx\|_2^2 = \|x\|_2^2$$

Demonstração. $\|Qx\|_2^2 = (Qx)^T(Qx) = x^TQ^TQx = x^Tx = \|x\|_2^2$. □

Ou seja, uma transformação ortogonal não expande e nem comprime um vetor, apenas realiza rotações e/ou reflexões.

D.2 Norma de Matrizes

Como uma matriz $m\times n$ pode ser vista como um vetor mn-dimensional, toda norma nessa dimensão pode ser usada para medir o 'tamanho' de tal matriz. De qualquer forma, como vetores e matrizes são objetos matemáticos diferentes, certas normas especiais são mais úteis que as normas vetoriais aplicadas em matrizes. Essas são as chamadas *normas matriciais induzidas*.

Dadas normas vetoriais $\|\cdot\|_{(n)}$ e $\|\cdot\|_{(m)}$ em $\mathbb{R}^n$ e $\mathbb{R}^m$, respectivamente, a norma matricial induzida $\|A\|_{(m,n)}$, onde $A \in \mathbb{R}^{n\times n}$, é o menor número C tal que a seguinte desigualdade é verdadeira para todo $x \in \mathbb{R}^n$:

$$\|Ax\|_{(m)} \leq C\|x\|_{(n)}\,.$$

Em outras palavras, $\|A\|_{(m,n)}$ é o supremo [1] das razões $\frac{\|Ax\|_{(m)}}{\|x\|_{(n)}}$ sobre todos os vetores $x \in \mathbb{R}^n$, ou seja, é o fator máximo que A consegue expandir um vetor x. Chamamos $\|\cdot\|_{(m,n)}$ de norma induzida por $\|\cdot\|_{(m)}$ e $\|\cdot\|_{(n)}$.

Por N3 da Definição D.1.1, podemos escrever a norma de matriz induzida em termos de vetores unitários:

$$\|A\|_{(m,n)} = \sup_{\substack{x\in\mathbb{R}^n\\ x\neq 0}} \frac{\|Ax\|_{(m)}}{\|x\|_{(n)}} = \sup_{\substack{x\in\mathbb{R}^n\\ \|x\|_{(n)}=1}} \|Ax\|_{(m)}\,.$$

Da mesma forma que as normas vetoriais, podemos definir algumas normas matriciais induzidas:

$$\|A\|_1 = \max_{1\leq j\leq n} \|a_j\|_1\,,\ \text{onde } a_j \text{ é a } j\text{-ésima coluna de A,}$$

$$\|A\|_2 = \sigma_{max}(A), \text{ onde } \sigma_{max}(A) \text{ é o maior valor singular de A,}$$
$$\|A\|_\infty = \max_{1\leq i\leq m} \|a_i^*\|_1, \text{ onde } a_i^* \text{ é a } i\text{-ésima linha de A.}$$

Apesar da importância das normas matriciais induzidas, as normas matriciais não precisam ser definidas dessa forma. No geral, uma norma matricial deve satisfazer as três condições para normas vetoriais da Definição D.1.1 aplicadas no espaço vetorial mn-dimensional para matrizes:

M1. $\|A\| \geq 0$ para todo x $\in \mathbb{R}^n$, e $\|A\| = 0$ se e somente se $A = 0$

M2. $\|\lambda A\| = |\lambda| \cdot \|A\|$, para todos $A \in \mathbb{R}^{m\times n}$ e $\lambda \in \mathbb{R}$

M3. $\|A + B\| \leq \|A\| + \|B\|$, para todos $A, B \in \mathbb{R}^{m\times n}$.

A norma matricial não-induzida mais importante é a *norma de Frobenius*, definida como

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}$$

Note que a norma de Frobenius é a norma euclidiana da matriz vista como um vetor mn-dimensional.

A norma de Frobenius é a única norma matricial utilizada no texto.

D.3 Número de Condicionamento

Nesta seção vamos analisar o que acontece quando perturbamos a matriz A no sistema linear $Ax = b$ e vamos definir o número de condicionamento $\kappa(A)$ da matriz A.

Seja b fixado e vamos observar o comportamento do sistema $Ax = b$ quando A é perturbado por um valor pequeno δA. Então, x também deve ser perturbado por um valor pequeno δx, onde

$$(A + \delta A)(x + \delta x) = b.$$

Vamos definir $\hat{x} := x + \delta x$ como a solução do sistema perturbado e o número de condicionamento de A como $\kappa(A) = \|A\|\left\|A^+\right\|$, onde A^+ é a matriz inversa de Moore-Penrose [27]. Então vale o seguinte teorema:

Teorema D.3.1 ([28, Página 134]). *Seja A não singular, $b \neq 0$, e sejam x e $\hat{x} = x + \delta x$ as soluções de $Ax = b$ e $(A + \delta A)\hat{x} = b$, respectivamente. Então*

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \kappa(A)\frac{\|\delta A\|}{\|A\|}. \tag{D.1}$$

O teorema mostra a importância do número de condicionamento de A. Se $\kappa(A)$ não é tão grande, então uma pequena perturbação em A resulta em uma pequena perturbação em x, no sentido de que $\frac{\|\delta x\|}{\|\hat{x}\|}$ é pequeno. Por outro lado, observe que é mais natural buscarmos uma estimativa para $\frac{\|\delta x\|}{\hat{x}}$, já que x representa de fato a solução do problema original que estamos interessados em resolver. Este resultado é apresentado a seguir.

Teorema D.3.2 ([28, Página 134]). *Se A é não singular, $\frac{\|\delta A\|}{\|A\|} < \frac{1}{\kappa(A)}$, $b \neq 0$, $Ax = b$ e $(A + \delta A)(x + \delta x) = b$, então*

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\kappa(A)\frac{\|\delta A\|}{\|A\|}}{1 - \kappa(A)\frac{\|\delta A\|}{\|A\|}}. \tag{D.2}$$

Se A é bem condicionada, ou seja, $\kappa(A)$ não é grande[1] e $\frac{\|\delta A\|}{\|A\|}$ é suficientemente pequeno, então $\frac{\|\delta A\|}{\|A\|} << \frac{1}{\kappa(A)}$. Nesse caso, o denominador do lado direito de D.2 é aproximadamente igual a 1. Logo, a Desigualdade D.3.2 diz praticamente que

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\delta A\|}{\|A\|},$$

o que é praticamente a mesma desigualdade do Teorema D.3.3. Isso mostra que se A é bem condicionada e $\frac{\|\delta A\|}{\|A\|}$ é pequeno, então $\frac{\|\delta x\|}{\|x\|}$ é pequeno. Por outro lado, se A é mal condicionada, então a desigualdade D.2 permite que $\frac{\|\delta x\|}{\|x\|}$ seja grande, mesmo que $\frac{\|\delta A\|}{\|A\|}$ seja pequeno.

Chegamos a uma conclusão semelhante à do Teorema D.3.3 quando perturbamos b por um valor δb e fixamos A:

Teorema D.3.3 ([28, Página 121]). *Seja A não singular, e seja x e $\hat{x} = x + \delta x$ as soluções de $Ax = b$ e $A\hat{x} = b + \delta b$, respectivamente. Então*

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \kappa(A)\frac{\|\delta b\|}{\|b\|}.$$

Para finalizar esta seção, vamos enunciar mais duas propriedades importantes do número de condicionamento $\kappa(A)$ de uma matriz A. Para um tratamento mais detalhado do assunto, veja a Seção 2 de [28].

Lema D.3.4. *Se $\kappa(A)$ é o número de condicionamento de A, então $\kappa(A) = \kappa(A^{-1})$, e, para c constante, $\kappa(cA) = \kappa(A)$ e $\kappa(A) \geq 1$.*

[1] Watkins [28] discute em seu livro sobre o que é considerado um número de condicionamento grande ou pequeno.