Determinação Numérica de Autovalores e Autovetores
Resolução Numérica de Sistemas de Equações Não Lineares

Projeto Computacional 2
MS512 - Análise Numérica
Professor Eduardo Cardoso de Abreu

Lucas Carrilho Pessoa - RA 094052

Conteúdo

1 SVD - Singular Value Decomposition

A **decomposição em valores singulares** ou, em inglês, *singular value decomposition (SVD)* é uma forma muito importante de decomposição de matrizes, sendo classificadas em [Watkins, 2010] como a forma mais importante de todas, tanto com propósitos teóricos, quanto computacionais.

Comparando a decomposição QR com a SVD, podemos observar que a decomposição SVD é uma ferramenta mais poderosa, já que com ela, por exemplo, podemos resolver problemas que envolvem matrizes que não possuem posto completo, ou, tampouco conhecemos o seu posto.

São inúmeras as aplicações de SVD, mas podemos elencar algumas: determinar o posto de matrizes; resolver problemas de quadrados mínimos (mesmo se a matriz de coeficientes não tiver posto completo); determinar a matriz pseudo-inversa e compressão de sinais.

A seguir será apresentado o teorema SVD, fortemente baseado em [Golub and Van Loan, 2013].

Teorema 1.1 (Singular Value Decomposition). Se A é uma matriz real de dimensão $m \times n$ ($A \in \mathbb{R}^{m\times n}$), existem matrizes ortogonais

$$U = [u_1|\cdots|u_m] \in \mathbb{R}^{m\times m} \text{ e } V = [v_1|\cdots|v_n] \in \mathbb{R}^{n\times n}$$

tais que

$$U^T AV = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n},\ p = \min\{m, n\},$$

onde $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$.

Antes de iniciarmos a demonstração do **Teorema SVD**, iremos apresentar um resultado importante da Álgebra Linear, que será utilizado na demonstração principal.

Teorema 1.2. Se $V_1 \in \mathbb{R}^{n\times r}$ tem colunas ortonormais, então existe $V_2 \in \mathbb{R}^{n\times(n-r)}$, de forma que $V = [V_1|V_2]$ é ortogonal.

Prova. Se $V \in \mathbb{R}^{n\times n}$ é uma matriz ortogonal, $V^T V = I$. Se dizemos que $V = [v_1|\cdots|v_n]$, então v_i forma uma base ortonormal em $\mathbb{R}^n$. Portanto, se $V_1 = [v_1|\cdots|v_r]$, e sabemos que $v_i,\ i = 0, 1, 2, \ldots r$, são colunas ortonormais, podemos construir $V_2 = [v_{r+1}|\cdots|v_n]$ de forma que V seja ortogonal.

□

Agora, vamos à demonstração do **Teorema 1.1 (Singular Value Decomposition)**.

Prova. Sejam $x \in \mathbb{R}^n$ e $y \in \mathbb{R}^m$ dois vetores tais que $|x|_2 = |y|_2 = 1$, e que satisfaçam $Ax = \sigma y$ com $\sigma = ||A||_2$. Do Teorema 1.2, sabemos que existe $V_2 \in \mathbb{R}^{n\times(n-1)}$ e $U_2 \in \mathbb{R}^{m\times(m-1)}$ de forma que $V = [\ x \mid V_2\ ] \in \mathbb{R}^{n\times n}$ e $U = [\ y \mid U_2\ ] \in \mathbb{R}^{m\times m}$ sejam ortogonais. Podemos, então, escrever

$$U^T AV = \left[\begin{array}{c} y^T \\ \hline U_2^T \end{array}\right] A \left[\ x \mid V_2\ \right] = \begin{bmatrix} \sigma & w^T \\ 0 & B \end{bmatrix} \equiv A_1$$

onde $w \in \mathbb{R}^{n-1}$ e $B \in \mathbb{R}^{(m-1)\times(n-1)}$. Como

$$\left\| A_1 \left(\begin{bmatrix} \sigma \\ w \end{bmatrix} \right) \right\|_2^2 \geq (\sigma^2 + w^T w)^2$$

teremos que $||A_1||_2^2 \geq (\sigma^2 + w^T w)$. Mas, pela definição, sabemos que $\sigma = ||A||_2$, então $\sigma^2 = ||A||_2^2 = ||A_1||_2^2$, e para isso ser verdade, temos que $w = 0$. Aplicando esses passos recursivamente à matriz B, teremos

$$U^T AV = \left[\begin{array}{c|cc} \sigma_1 & 0 & 0 \\ \hline 0 & \sigma_2 & w_2^T \\ 0 & 0 & B_2 \end{array}\right] \equiv A_2$$

onde $w_2 \in \mathbb{R}^{n-2}$ e $B_2 \in \mathbb{R}^{(m-2)\times(n-2)}$. Com isso podemos ver que, recursivamente, a prova do teorema está completa.

□

A decomposição SVD pode ser aplicada para matrizes que não são definida-positivas e simétricas. Já que a decomposição SVD utiliza bases diferentes V e U para os espaços linha e coluna e usa bases ortonormais, ela pode ser utilizada para qualquer matriz. Contudo, ao analisamos a decomposição em autovalores $A = X \wedge X^{-1}$, ela utiliza a mesma base X para os espaços linha e coluna, geralmente não usa uma base ortonormal e é definida somente para matrizes quadradas e podemos, então, concluir que para matrizes definida-positivas e simétricas, a decomposição SVD e a decomposição em autovalores são idênticas.

1.1 Interpretação Geométrica

Como pode ser visto em [Trefethen and III, 1997] na sua Lecture 4, a decomposição SVD é motivada pelo seguinte fato geométrico:

A imagem de uma esfera unitária sobre qualquer matriz $m \times n$ é uma hiperelipse.

Para uma interpretação geométrica, consideraremos apenas matrizes reais na decomposição SVD, contudo, sabemos que essa transformação pode ser aplicada para matrizes $\mathbb{R}^{n \times m}$ e $\mathbb{C}^{n \times m}$.

Como descrito em [Trefethen and III, 1997], o termo hiperelipse pode não ser familiar, mas é somente uma generalização m-dimensional de uma elipse. Consideremos uma hiperelipse em $\mathbb{R}^m$ como uma superfície obtida alongando uma esfera unitária em $\mathbb{R}^m$ por fatores $\sigma_1, \ldots, \sigma_m$ ao longo das direções ortogonais $u_1, \ldots, u_m \in \mathbb{R}^m$. Note que os fatores $\sigma_1, \ldots, \sigma_m$ podem ser, inclusive, zero. Por conveniência, adotemos u_i como vetores unitários. Assim, os vetores $\sigma_i u_i$ formam os semieixos principais da hiperelipse com normas $\sigma_i, \ldots, \sigma_m$.

Se uma matriz A possui posto de dimensão r, exatamente r normas de σ_i serão não nulas, e, em particular, se $m \geq n$, no máximo n normas de σ_i serão não nulas.

Voltando ao fato geométrico apresentado a priori, por *esfera unitária*, estamos nos referindo a uma esfera Euclidiana usual n-dimensional, denotada por S. Então AS, a imagem da esfera S sobre uma transformação A é uma hiperelipse.

Na Figura 1 podemos ver o que acontece na decomposição SVD de uma matriz quadrada de dimensão 2.

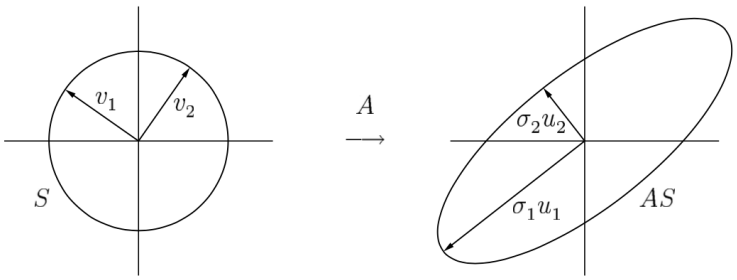

Figura 1: Decomposição SVD de uma matriz 2×2. Extraída de [Trefethen and III, 1997].

Fundamentalmente, a decomposição SVD $A = U\Sigma V^T$ decompõe a matriz A em três transformações: a rotação definida por V^T, a escala definida por Σ e uma segunda rotação definida por U. Na Figura 2 podemos ver os três passos da transformação: rotação V^T, escala Σ e a segunda rotação U.

1.2 Compressão de Imagens

Em teoria de sinais e telecomunicações, existe uma grande necessidade de se comprimir sinais garantindo o mínimo de perdas. SVD é uma forma eficiente de se minimizar a quantidade de espaço necessário para se armazenar imagens, preservando informações importantes. Imagens são, essencialmente matrizes $m \times n$ onde $f(i,j),\ \forall i = 0, 1, \ldots, m-1$ *e* $\forall j = 0, 1, \ldots, n-1$ representa um pixel dessa imagem. O valor máximo que um pixel $f(i,j)$ pode atingir é chamado de profundidade do pixel. Existem imagens binárias, onde $f(i,j) = [0,1]$, ou imagens em 256 tons de cinza, onde $f(i,j) = [0,255]$.

Uma imagem M de tamanho $m \times n$ que não está comprimida necessita de um espaço de armazenamento $z_M = mn\rho$ bits, onde ρ representa o número de bits necessário para armazenar um pixel, ou seja, $\rho = log_2 P$, onde P representa a profundidade máxima de um pixel na imagem. Quando uma decomposição SVD for aplicada à imagem M, ela necessitará de um espaço de armazenamento $z_M = k(m+n+1)\rho$ onde k é o posto da matriz M. Isso mostra que SVD é utilizado para reduzir o posto da matriz e, consequentemente, o tamanho de armazenamento.

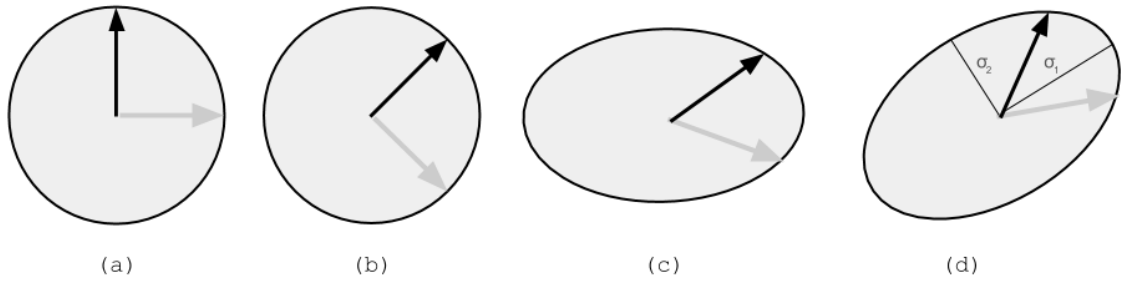

Figura 2: Decomposição SVD de uma matriz 2×2. (a) A esfera original. (b) A esfera após a rotação V^T. (c) A esfera rotacionada após a transformação de escala Σ. (d) A esfera final, depois de passar pela transformação de rotação U, ou seja, a esfera após a decomposição SVD: $A = U\Sigma V^T$.

Para imagens em tons de cinza, podemos observar nas Figuras 3 e 4 o resultado da compressão obtida para diferentes ranks.

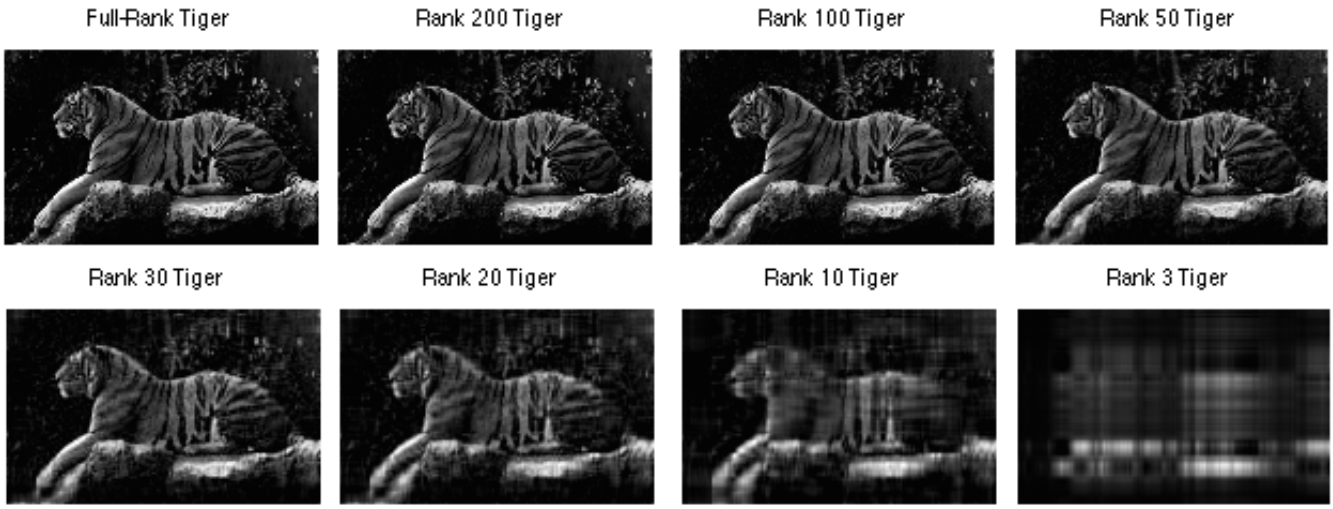

Figura 3: Resultados da decomposição SVD para uma imagem de um tigre sujeita a diversas compressões. Observe os postos variando entre 3 até o completo.

Agora que já aplicamos a decomposição SVD para compactação de imagens em escala de cinza, veremos como aplicar a decomposição em imagens coloridas, como mostrado em [Wheeler, 2013]. Essas imagens são armazenadas na forma de três matrizes, cada uma delas contendo um canal de cor aditiva: vermelho, verde e azul. A forma que cada um desses canais é estruturado é o mesmo que foi apresentado para as imagens em escala de cinza: uma matriz $m \times n$ onde $f(i, j)$, $\forall i = 0, 1, \ldots, m-1$ *e* $\forall j = 0, 1, \ldots, n-1$ representa um pixel desse canal. A junção dos três canais forma a imagem colorida.

Cada um dos canais pode ser tratado como uma diferente imagem em tom de cinza. Observe na Figura 5 uma imagem colorida e as representações em escala de cinza de cada um dos seus canais.

Depois de separar a imagem em cada um de seus três canais, aplicamos então a decomposição SVD em cada um de seus componentes de cor. Na Figura 6 pode-se observar o resultado da decomposição SVD para uma imagem colorida, variando o nível de compressão em cada uma delas. O processo também foi aplicado a uma outra imagem que explora a variação de tons entre o vermelho e o violeta. Com essa imagem é bem visível a diferença entre os seus canais vermelho, verde e azul, e isso pode ser visto na Figura 7. O resultado da compressão, para diferentes níveis, pode ser visto na Figura 8. Como a imagem explora a variação das cores, é possível visualizar bem o impacto da compressão em cada um dos canais.

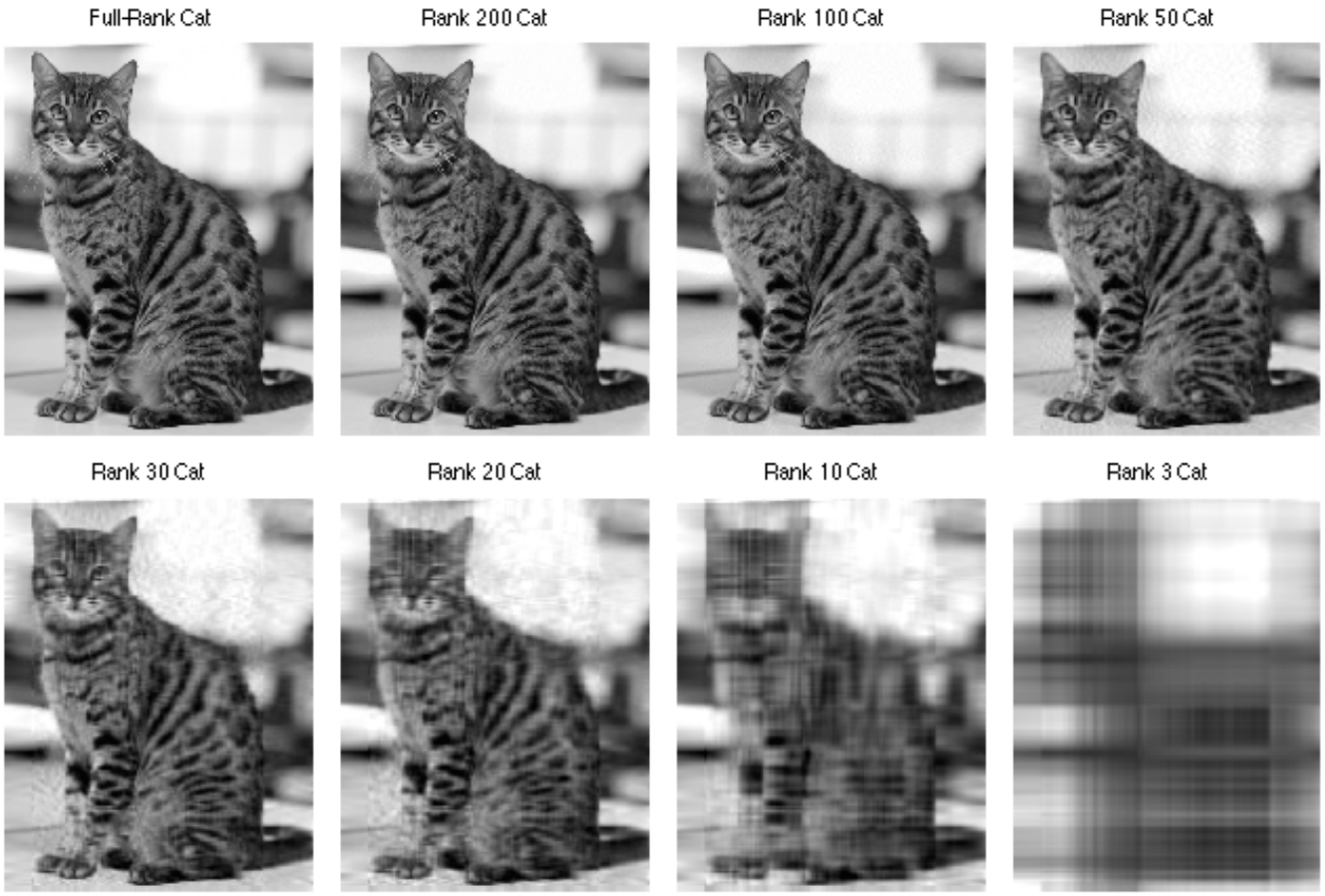

Figura 4: Resultados da decomposição SVD para uma imagem de um gato sujeita a diversas compressões. Observe os postos variando entre 3 até o completo.

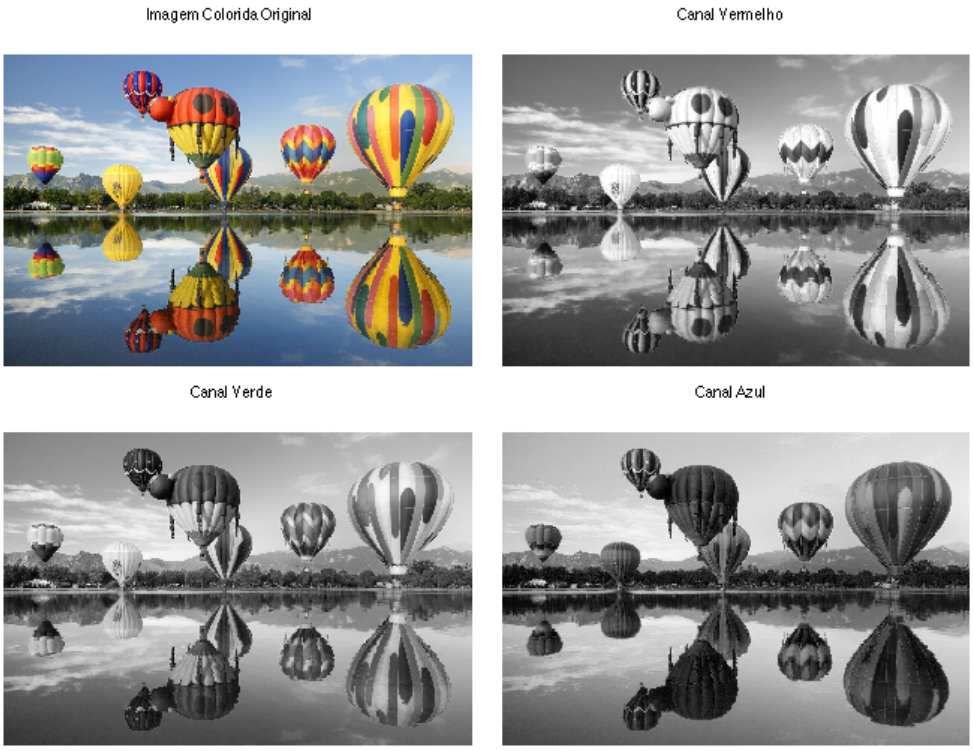

Figura 5: Imagem colorida original e uma visualização de cada um de seus três canais: vermelho, verde e azul.

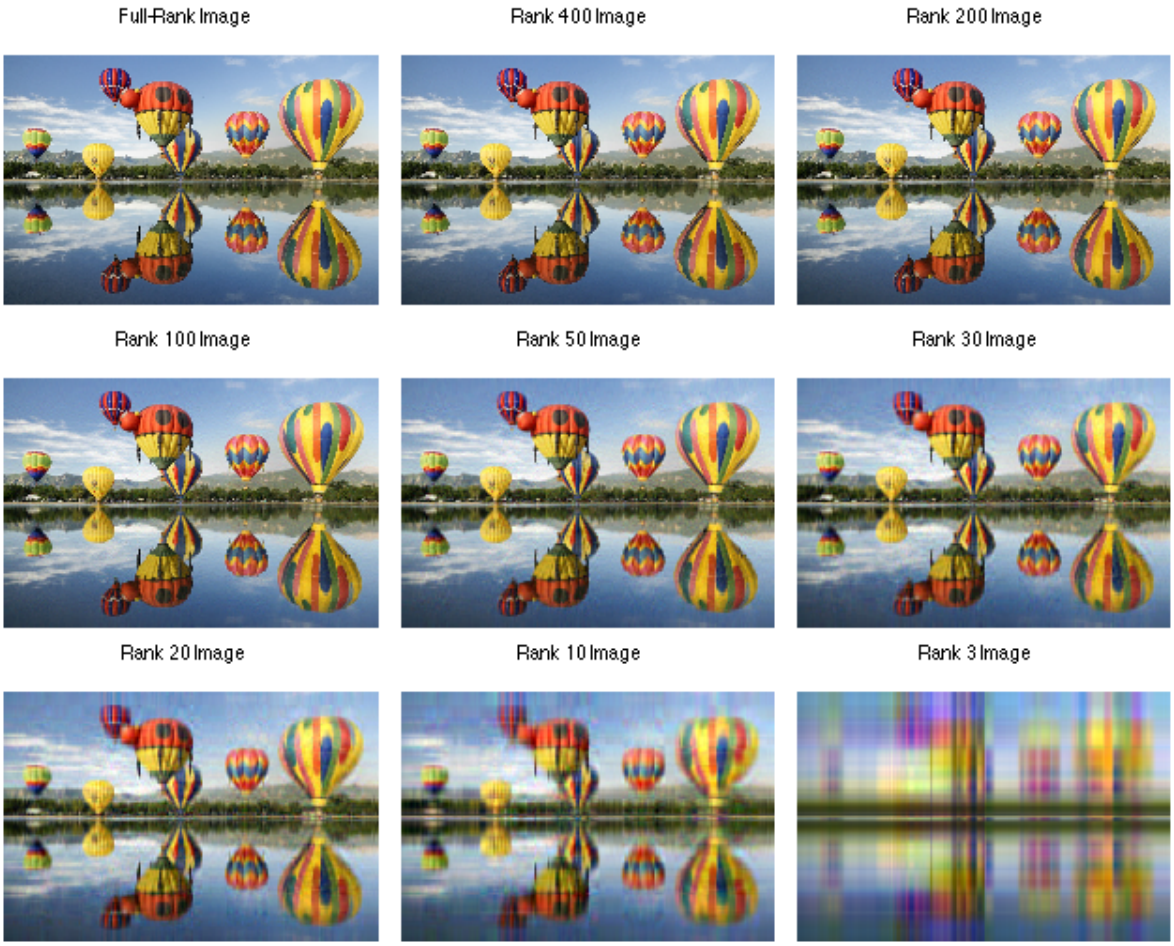

Figura 6: Resultado da decomposição SVD para uma imagem colorida de balões na natureza. Observe as diferentes compressões e a sua interferência no resultado obtido.

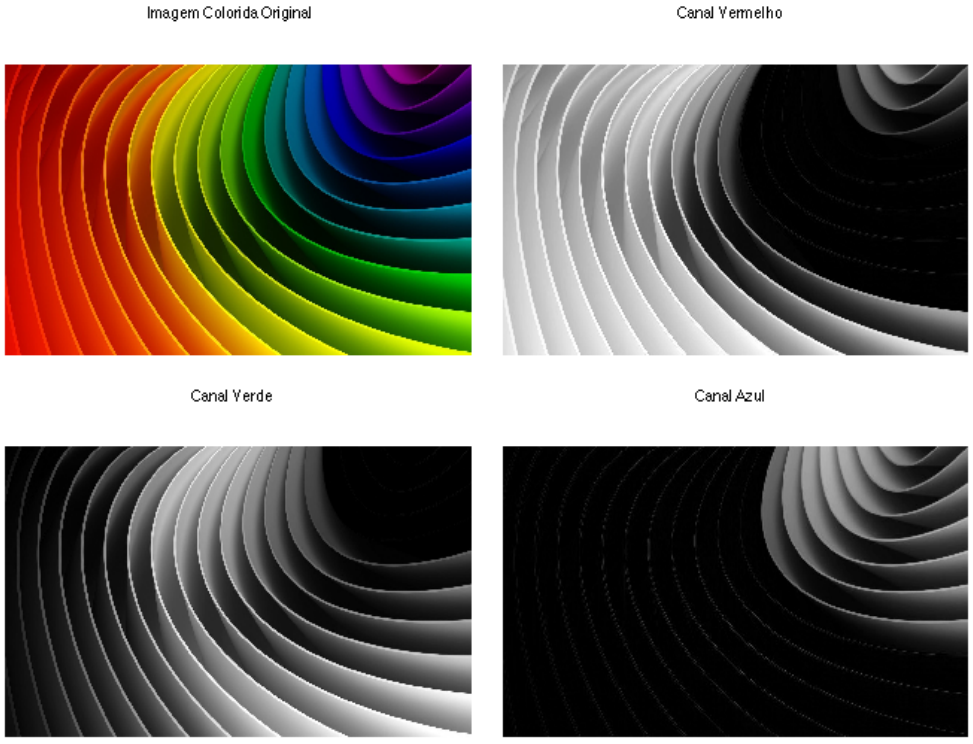

Figura 7: Imagem colorida original e uma visualização de cada um de seus três canais: vermelho, verde e azul.

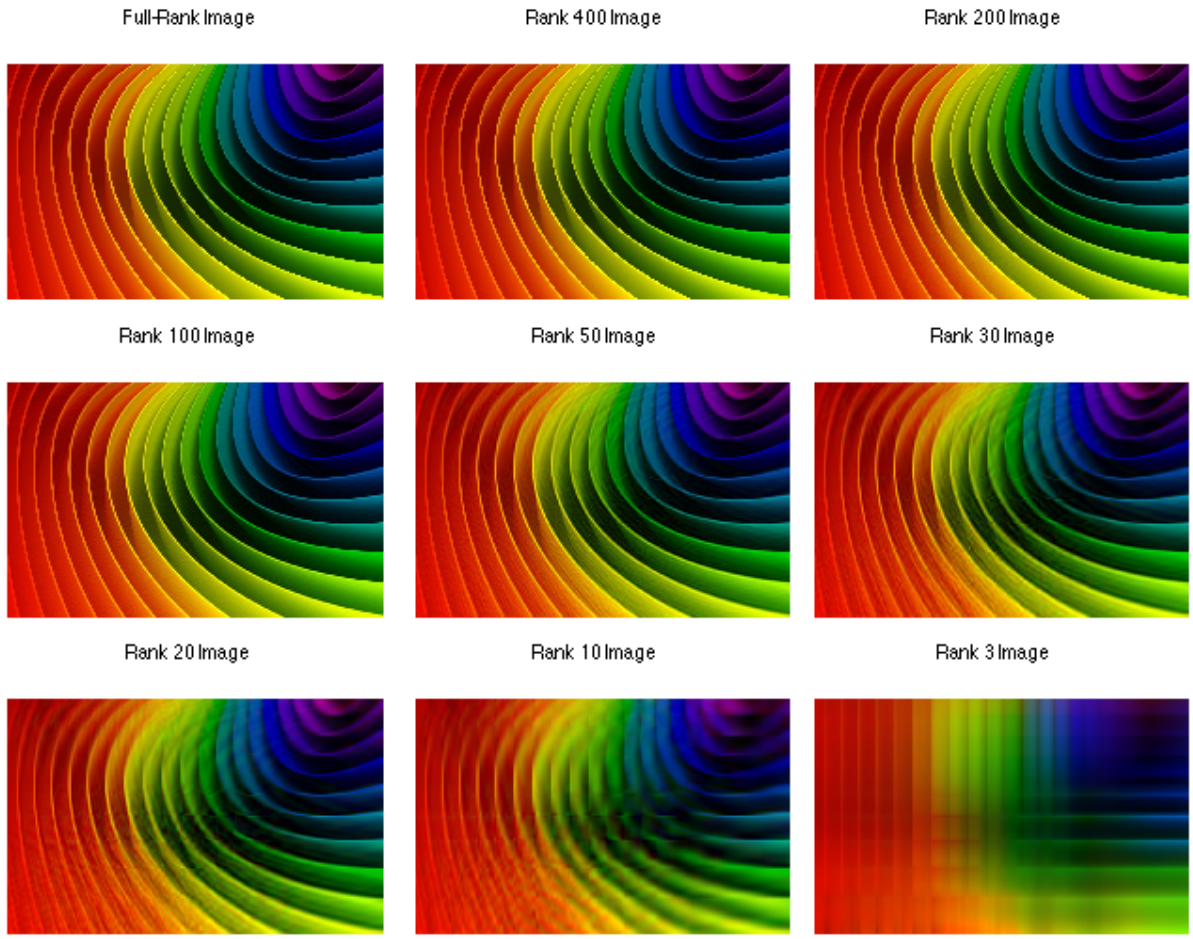

Figura 8: Resultado da decomposição SVD para uma imagem que explora a variação do espectro de cores. Observe as diferentes compressões e a sua interferência no resultado obtido em cada um dos canais, que pode ser observado em cada uma das regiões da imagem.

A seguir, é apresentado o código MATLAB que foi modificado para a determinação da decomposição SVD para imagens coloridas.

```
%Lucas Carrilho Pessoa - RA 094052
%
%Programa modificado do exemplo exibido em aula
%MS512 - Analise Numerica

close all
clear all

% Read the picture and show it's three channel

inputImage = imread('ballons.jpg'); %matriz n x m x 3: R,G,B
imageR = inputImage(:,:,1);
imageG = inputImage(:,:,2);
imageB = inputImage(:,:,3);

%Plot all channels from the original full-coloured picture
figure(1);
subplot(2,2,1); image(inputImage),title('Imagem Colorida Original'); axis off; axis('equal');
subplot(2,2,2), colormap(gray(256)); image(imageR),title('Canal Vermelho'); axis off; axis('equal');
subplot(2,2,3), colormap(gray(256)); image(imageG),title('Canal Verde'); axis off; axis('equal');
subplot(2,2,4), colormap(gray(256)); image(imageB),title('Canal Azul'); axis off; axis('equal');


% Show full-rank image
figure(2); subplot(3,3, 1); image(inputImage), title('Full-Rank Image'); axis off;
axis('equal')


inputImage = double(inputImage);

% For each channel, compute SVD
imageR = inputImage(:,:,1);
```

```
imageG = inputImage(:,:,2);
imageB = inputImage(:,:,3);
[UR, SR, VR] = svd(imageR);
[UG, SG, VG] = svd(imageG);
[UB, SB, VB] = svd(imageB);


sigmasR = diag(SR); sigmasG = diag(SG); sigmasB = diag(SB);

figure(2)
% Compute low-rank approximations of the tiger, and show them
ranks = [400, 200, 100, 50, 30, 20, 10, 3];
for i = 1:length(ranks)
    % Keep largest singular values, and nullify others.
    approx_sigmasR = sigmasR; approx_sigmasR(ranks(i):end) = 0;
    approx_sigmasG = sigmasG; approx_sigmasG(ranks(i):end) = 0;
    approx_sigmasB = sigmasB; approx_sigmasB(ranks(i):end) = 0;

    % Form the singular value matrix, padded as necessary
    nsR = length(sigmasR);
    approx_SR = SR; approx_SR(1:nsR, 1:nsR) = diag(approx_sigmasR);
    nsG = length(sigmasG);
    approx_SG = SG; approx_SG(1:nsG, 1:nsG) = diag(approx_sigmasG);
    nsB = length(sigmasB);
    approx_SB = SB; approx_SB(1:nsB, 1:nsB) = diag(approx_sigmasB);

    % Compute low-rank approximation by multiplying out component matrices.
    approx_imageR = UR * approx_SR * VR';
    approx_imageG = UG * approx_SG * VG';
    approx_imageB = UB * approx_SB * VB';

    % Plot approximation
    img = cat(3,approx_imageR/255., approx_imageG/255., approx_imageB/255.);
    subplot(3,3, i + 1); image(im2uint8(img)), title(sprintf('Rank %d Image', ranks(i))); axis off;
    axis('equal')
end
```

2 Teorema de Schur e Teorema Espectral para Matrizes Simétricas Reais

2.1 Teorema de Schur

O seguinte enunciado e demonstração do teorema, se baseia fortemente no posto em [Watkins, 2010], porém, a demonstração também pode ser vista em [Pulino, 2015].

Teorema 2.1 (Schur's Theorem). Se $A \in \mathbb{C}^{n\times n}$, então existe uma matriz unitária $U \in \mathbb{C}^{n\times n}$ e uma matriz triangular superior $T \in \mathbb{C}^{n\times n}$ que $T = U^*AU$. Podemos também escrever $A = UTU^*$. Essa é a decomposição de Schur da matriz A.

Prova. A prova é feita sob indução em n. O resultado é trivial para $n = 1$. Agora, devemos provar que o resultado é mantido para $n = k$, dada a hipótese que ele vale para $n = k - 1$.

Seja $A \in \mathbb{C}^{k\times k}$. Seja λ um autovalor de A e v o seu autovetor associado, escolhido de forma que $||v||_2 = 1$. Seja U_1 uma matriz unitária qualquer que contém v como a sua primeira coluna. Existem várias matrizes nessa condição, peguemos qualquer base ortonormal de $\mathbb{C}^k$ em que o primeiro elemento é v, e fazendo U_1 uma matriz cujas colunas são elementos dessa base. Seja $W \in \mathbb{C}^{k\times(k-1)}$ uma submatriz de U_1 contendo as colunas de 2 a k, de forma que $U_1 = \begin{bmatrix} v & W \end{bmatrix}$. Como as colunas de W são ortogonais a v, $W^*v = 0$. Seja $A_1 = U_1^*AU_1$. Então

$$A_1 = \begin{bmatrix} v^* \\ W^* \end{bmatrix} A \begin{bmatrix} v & W \end{bmatrix} = \begin{bmatrix} v^*Av & v^*AW \\ W^*Av & W^*AW \end{bmatrix}.$$

Como $Av = \lambda v$, segue que $v^*Av = \lambda$ e $W^*Av = \lambda W^*v = 0$. Seja $\hat{A} = W^*AW$. Então A_1 tem a forma

$$A_1 = \left[\begin{array}{c|ccc} \lambda & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & \hat{A} & \\ 0 & & & \end{array}\right]$$

de forma que $\hat{A} \in \mathbb{C}^{(k-1)\times(k-1)}$, então pela hipótese de indução existe uma matriz unitária $\hat{U}_2$ e uma matriz triangular superior $\hat{T}$ de forma que $\hat{T} = \hat{U}_2^*\hat{A}\hat{U}_2$. Definimos, então, a matriz $U_2 \in \mathbb{C}^{k\times k}$ como

$$U_2 = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \hat{U}_2 & \\ 0 & & & \end{array}\right].$$

Então, U_2 é unitária, e

$$U_2^*A_1U_2 = \left[\begin{array}{c|ccc} \lambda & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & \hat{U}_2^*\hat{A}\hat{U}_2 & \\ 0 & & & \end{array}\right] = \left[\begin{array}{c|ccc} \lambda & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & \hat{T} & \\ 0 & & & \end{array}\right],$$

que é triangular superior. Chamemos essa matriz de T, e seja $U = U_1U_2$. Então,

$$T = U_2^*A_1U_2 = U_2^*U_1^*AU_1U_2 = U^*AU.$$

□

2.2 Teorema Espectral para Matrizes Simétricas Reais

O seguinte enunciado e demonstração do teorema, se baseia fortemente no posto em [Watkins, 2010], porém, a demonstração, de uma forma mais elaborada, também pode ser vista em [Pulino, 2015]. A prova é muito semelhante à exibida na seção anterior, para o Teorema de Schur, exceto pelo fato de que, agora, as matrizes são reais e de que podemos explorar o fato de suas simetrias.

Teorema 2.2 (Teorema Espectral para Matrizes Simétricas Reais). Seja A uma matriz simétrica e que $A \in \mathbb{R}^{n\times n}$. Então, existe uma matriz ortogonal $U \in \mathbb{R}^{n\times n}$ e uma matriz diagonal $D \in \mathbb{R}^{n\times n}$ em que $D = U^TAU$.

Prova. A prova é feita sob indução em n. O resultado é trivial para $n = 1$. Agora, devemos provar que o resultado é mantido para $n = k$, dada a hipótese que ele vale para $n = k - 1$.

Seja $A \in \mathbb{R}^{k\times k}$. Seja λ um autovalor de A e v o seu autovetor associado. Como λ é real, então, o seu autovetor associado v também é real, e deve ser escolhido de forma que $||v||_2 = 1$. Seja U_1 uma matriz real ortogonal onde v é a sua primeira coluna. Existem várias matrizes nessa condição, peguemos qualquer base ortonormal de $\mathbb{R}^k$ em que o primeiro elemento é v, e fazendo U_1 uma matriz cujas colunas são elementos dessa base, isto é, as outras colunas são o complemento ortogonal de uma base qualquer em $\mathbb{R}^n$. Seja $W \in \mathbb{R}^{k\times(k-1)}$ uma submatriz de U_1 contendo as colunas de 2 a k, de forma que $U_1 = [\ v \ \ W\ ]$. Como as colunas de W são ortogonais a v, $W^T v = 0$. Seja $A_1 = U_1^T A U_1$. Então

$$A_1 = \begin{bmatrix} v^T \\ W^T \end{bmatrix} A \begin{bmatrix} v & W \end{bmatrix} = \begin{bmatrix} v^T Av & v^T AW \\ W^T Av & W^T AW \end{bmatrix}.$$

Como $Av = \lambda v$, segue que $v^T Av = \lambda$ e $W^T Av = \lambda W^T v = 0$. Seja $\hat{A} = W^T AW$. Então A_1 tem a forma

$$A_1 = \left[\begin{array}{c|ccc} \lambda & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & \hat{A} & \\ 0 & & & \end{array}\right]$$

de forma que $\hat{A} \in \mathbb{R}^{(k-1)\times(k-1)}$ é simétrica. Então, pela hipótese de indução, podemos assumir que existe uma matriz ortogonal $\hat{U}_2$ e uma matriz diagonal $\hat{D}$ de forma que $\hat{D} = \hat{U}_2^T \hat{A} \hat{U}_2$. Definimos, então, as matrizes $U_2 \in \mathbb{R}^{k\times k}$ e $D \in \mathbb{R}^{k\times k}$ como

$$U_2 = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \hat{U}_2 & \\ 0 & & & \end{array}\right], \quad D = \left[\begin{array}{c|ccc} \lambda & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \hat{D} & \\ 0 & & & \end{array}\right],$$

fazendo $U = U_1 U_2$ e utilizando a expressão da matriz A_1, assim obtemos

$$D = U_2^T A_1 U_2 = U_2^T U_1^T A U_1 U_2 = U^T AU.$$

□

3 Sylvester

3.1 Equação de Sylvester

Como visto em [Higham, 2014], a equação de Sylvester é escrita como a seguinte equação matricial:

$$AX + XB = C \tag{1}$$

onde A é $m \times m$, B é $n \times n$, e X é a matriz desconhecida, com dimensão $m \times n$. Em 1889, Sylvester, em [Sylvester, 1884], considerou a forma homogênea da equação e mostrou que a equação (1) tem uma única solução se A e $-B$ não possuírem nenhum autovalor em comum.

Como a equação é linear em X, é possível escrevê-la na forma mais usual $Ax = b$. Além disso, utilizando o operador de *vetorização vec*, que converte uma matriz em um vetor-coluna, empilhando cada uma de suas colunas sobre as outras; e também o produto de Kronecker, podemos escrever:

$$(I_n \otimes A + B^T \otimes I_m)vec(X) = vec(C) \tag{2}$$

onde, par $F \in \mathbb{R}^{p \times q}$ e $G \in \mathbb{R}^{r \times s}$, o produto de Kronecker é definido como $F \otimes G := (f_{ij}G) \in \mathbb{R}^{pr \times qs}$.

A matriz de coeficientes apresentada na equação (2) tem uma estrutura grande, mas tira-se proveito direto desse fato na análise da equação (1) diretamente. Dentre diversas maneiras que já se obtiveram para a solução, podemos mencionar a seguinte: se a integral $\int_0^\infty e^{At}Ce^{Bt}dt$ existe, então o oposto dessa integral é a solução da equação de Sylvester.

Existem diversas aplicações da equação de Sylvester. Em teoria de controle, se faz uso de uma variação da equação de Sylvester:

$$AX + XA^* = C \tag{3}$$

onde $*$ denota a matriz conjugada transposta. Essa variação é denominada *Equação de Lyapunov*. Em [Gene F. Franklin and Emami-Naeini, 2013] podemos observar que se uma função de Lyapunov pode ser encontrada para um sistema, então o movimento é estável. Sendo assim, essa equação é amplamente utilizada em métodos que buscam projetar sistemas estáveis.

Encontrar a matriz X que satisfaça a equação de Sylvester não é uma tarefa fácil. Um algoritmo clássico para uma solução numérica dessa equação é o *algoritmo de Bartels-Stewart*. Em [G. H. Golub and Loan, 1979] podemos ver uma apresentação do algoritmo, e também trabalhos que foram desenvolvidos sobre ele, o tornando mais rápido e eficiente. A ideia por trás do *algoritmo de Bartels-Stewart* consiste em transformar as matrizes A e B na forma de Schur, através do algoritmo QR, e resolvendo o sistema triangular resultante através de substituição. O MATLAB apresenta uma função chamada `sylvester` que é capaz de determinar a matriz X, quando ela existir.

3.2 Lei da Inércia de Sylvester

A lei da inercia de Sylvester, também mostrada em [Higham, 2014], baseia-se inércia de uma matriz Hermitiana é o conjunto dos três inteiros (ν, ζ, π), onde ν representa o número de autovalores negativos, ζ o número de autovalores nulos e π o número de autovalores positivos. A lei da inércia de Sylvester diz que para qualquer matriz Hermitiana A e, dado uma matriz X não singular, a inércia de A é a mesma que de X^*AX. A transformação de forma X^*AX é chamada de congruência, então a lei da inércia de Sylvester diz que o número de autovalores negativos, zeros e positivos não muda nesse tipo de transformação.

Uma das diversas aplicações da lei da inércia de Sylvester é o cálculo dos autovalores de matrizes Hermitianas com estrutura de banda T. Suponha que queremos calcular o i-ésimo menor autovalor de T. Seja $N(x)$ o número de autovalores de T que são menores do que x. Queremos encontrar o ponto onde $N(x)$ muda de $i-1$ para i. Suponha que fazemos a fatoração $T - xI = LDL^*$, onde D é uma matriz diagonal e L é uma matriz unitária, também com estrutura de banda. Essa fatoração pode ser computada em $O(n)$ operações, e a lei da inércia de Sylvester nos diz que $T - xI$ tem a mesma inércia que D, então o número de elementos negativos na diagonal de D é igual ao número de autovalores negativos de $T - xI$, que será o número de autovalores de T menores que do que 0, que será, então, o número de autovalores de T menores do que x, ou seja, $N(x)$. Esse tipo de fatoração é numericamente instável se desejamos resolver um sistema linear na forma $Ax = b$, porém, para determinar a diagonal de D, o método é perfeitamente estável, como mostrado em [Demmel, 1997].

Contudo, a lei da inércia de Sylvester não fala nada sobre a magnitude dos autovalores depois da transformação de congruência. Em [Horn and Johnson, 2013] Ostrowski mostra que

$$\lambda_i(X^*AX) = \theta_i\lambda_i(A) \tag{4}$$

onde $\lambda_n(X^*X) \leq \theta_i \leq \lambda_1(X^*X)$, contando que os autovalores estão ordenados na forma $\lambda_n \leq \lambda_{n-1} \leq \ldots \lambda_1$. Esse resultado é muito útil para o desenvolvimento mínimo de perturbações em grupos de matrizes que mudam a sua inércia de uma maneira bem definida.

3.3 James Joseph Sylvester

Sylvester ($\star$1814 - †1897) foi um matemático Inglês, que fez substanciais contribuições nas áreas de álgebra matricial, teoria de invariantes, teoria dos números, teoria das partições e combinatória. Ele foi por muito tempo professor da Universidade de Johns Hopkins nos Estados Unidos da América, ao longo do século XIX, e foi o fundador do *American Journal of Mathematics*.

Em 1850 ele criou o termo **matriz** com a seguinte definição (extraído de [Higham, 2014]):

> *We must commence, not with a square, but with an oblong arrangement of terms consisting, suppose, of m lines and n columns. This will not in itself represent a determinant, but is, as it were, a Matrix out of which we may form various systems of determinants by fixing upon a number p, and selecting at will p lines and p columns, the squares corresponding to which may be termed determinants of the p-th order.*

Apesar da primeira definição de matriz vir de Sylvester, o determinante já estava em uso comum quando o termo matriz foi definido. Além desta definição, Sylvester também publicou um glossário, em 1853, contendo seis páginas de "termos novos, ou não usuais, ou termos utilizados em um sentido novo, ou não usual", definindo termos utilizados até hoje na matemática. O dicionário Inglês de Oxford credita os seguintes termos a Sylvester: *matrix, minor, syzygy, canonical form, discriminant, invariant, Jacobian, covariant, latent root, nullity.* Além disso, em todos os seus estudos sobre matrizes e desenvolvimentos nesse sentido, Sylvester marcou importância para vários termos e conceitos que são amplamente utilizados, até hoje, nos conhecimentos de análise numérica.

Um exemplo forte, foi o termo *latent root*, que apesar de ter caído em desuso, ainda é muito utilizado. Esse termo foi utilizado até os anos 1970, mas, agora, foi substituído por autovalor, em uma tradução não muito correta do alemão *eigenwert*. Contudo, Sylvester contém o crédito de ter utilizado o símbolo λ, para autovalores de uma matriz, em 1852.

4 Fatoração QR

4.1 Transformação de Givens

A partir do código fornecido para a Fatoração QR, utilizando a transformação de Givens, foi feita uma modificação que exibe, para cada passo iterativo, a matriz de rotação ortogonal U e também o resultado da matriz Q. Abaixo é possível visualizar, para uma matriz A de dimensão 4×3 o passo a passo do processo iterativo de fatoração. Observe que, logo abaixo, consta a matriz R calculada diretamente pelo MATLAB, assim como estava no código original. Logo abaixo das matrizes, consta o código modificado.

$$U_{2,1} = \begin{bmatrix} 0.991 & 0.13 & 0 & 0 \\ -0.13 & 0.991 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_{2,1} = \begin{bmatrix} 0.991 & -0.13 & 0 & 0 \\ 0.13 & 0.991 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_{2,1} = \begin{bmatrix} 0.807 & 0.407 & 0.835 \\ 0 & 0.38 & 0.655 \\ 0.821 & 0.572 & 0.389 \\ 0.841 & 0.701 & 0.429 \end{bmatrix}$$

$$U_{3,1} = \begin{bmatrix} -0.701 & 0 & -0.713 & 0 \\ 0 & 1 & 0 & 0 \\ 0.713 & 0 & -0.701 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_{3,1} = \begin{bmatrix} -0.695 & -0.13 & 0.707 & 0 \\ -0.0912 & 0.991 & 0.0928 & 0 \\ -0.713 & 0 & -0.701 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_{3,1} = \begin{bmatrix} -1.15 & -0.694 & -0.863 \\ 0 & 0.38 & 0.655 \\ 0 & -0.111 & 0.323 \\ 0.841 & 0.701 & 0.429 \end{bmatrix}$$

$$U_{4,1} = \begin{bmatrix} 0.808 & 0 & 0 & -0.59 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.59 & 0 & 0 & 0.808 \end{bmatrix} \quad Q_{4,1} = \begin{bmatrix} -0.561 & -0.13 & 0.707 & -0.41 \\ -0.0737 & 0.991 & 0.0928 & -0.0538 \\ -0.576 & 0 & -0.701 & -0.421 \\ -0.59 & 0 & 0 & 0.808 \end{bmatrix} \quad A_{4,1} = \begin{bmatrix} -1.43 & -0.974 & -0.95 \\ 0 & 0.38 & 0.655 \\ 0 & -0.111 & 0.323 \\ 0 & 0.157 & -0.162 \end{bmatrix}$$

$$U_{3,2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.96 & -0.279 & 0 \\ 0 & 0.279 & 0.96 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_{3,2} = \begin{bmatrix} -0.561 & -0.322 & 0.643 & -0.41 \\ -0.0737 & 0.926 & 0.366 & -0.0538 \\ -0.576 & 0.196 & -0.673 & -0.421 \\ -0.59 & 0 & 0 & 0.808 \end{bmatrix} \quad A_{3,2} = \begin{bmatrix} -1.43 & -0.974 & -0.95 \\ 0 & 0.396 & 0.539 \\ 0 & 0 & 0.493 \\ 0 & 0.157 & -0.162 \end{bmatrix}$$

$$U_{4,2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.93 & 0 & 0.368 \\ 0 & 0 & 1 & 0 \\ 0 & -0.368 & 0 & 0.93 \end{bmatrix} \quad Q_{4,2} = \begin{bmatrix} -0.561 & -0.451 & 0.643 & -0.262 \\ -0.0737 & 0.841 & 0.366 & -0.391 \\ -0.576 & 0.027 & -0.673 & -0.463 \\ -0.59 & 0.297 & 0 & 0.751 \end{bmatrix} \quad A_{4,2} = \begin{bmatrix} -1.43 & -0.974 & -0.95 \\ 0 & 0.426 & 0.441 \\ 0 & 0 & 0.493 \\ 0 & 0 & -0.349 \end{bmatrix}$$

$$U_{4,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.816 & -0.578 \\ 0 & 0 & 0.578 & 0.816 \end{bmatrix} \quad Q_{4,3} = \begin{bmatrix} -0.561 & -0.451 & 0.676 & 0.158 \\ -0.0737 & 0.841 & 0.525 & -0.108 \\ -0.576 & 0.027 & -0.281 & -0.767 \\ -0.59 & 0.297 & -0.434 & 0.613 \end{bmatrix} \quad A_{4,3} = \begin{bmatrix} -1.43 & -0.974 & -0.95 \\ 0 & 0.426 & 0.441 \\ 0 & 0 & 0.604 \\ 0 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} -1.43 & -0.974 & -0.95 \\ 0 & -0.426 & -0.441 \\ 0 & 0 & 0.604 \end{bmatrix}$$

```
%Lucas Carrilho Pessoa - RA094052
%MS512 - Analise Numerica
%Modificacoes a partir do codigo fornecido

% Ilustracao da factoracao QR via "Givens rotations"
% # linhas m > n # colunas
m = 4; n=3; %m=12; n=8; %m=5; n=3;
A=rand(m,n);
% calculo da fatorazacao QR reduzida
[Q,R]=qr(A,0);

% matriz Q computada a cada iteracao
Qi = eye(m,m);
for j=1:n,
    for k=j+1:m,
        % calculo da rotacao Givens para zerar o elemento (k,j)
        [c,s]=givensrot(A(j,j),A(k,j));
        G=[ c s; -s c ];

        % define matriz de transformacao ortogonal U para iteracao i
        Ui = eye(m,m);
        Ui(k,k) = c; Ui(j,j) = c;
        Ui(k,j) = s; Ui(j,k) = -s
        Qi = Qi * Ui'
        % rotacao das linha j e k
        % observe como linhas sao selecionadas usando um vetor coluna
        A([ j k ],j:n)=G'*A([ j k ],j:n)
        % aguardando digitacao <enter> no teclado pelo usuario
```

```
        pause
    end
end
% comparando a solucao matlab R ("[Q,R]=qr(A,0);") com a solucao
% aproximada obtida acima com a fatoracao QR via "Householder reflections"
R
```

4.2 Transformação de Householder

A partir do código fornecido para a Fatoração QR, utilizando a transformação de Householder, foi feita uma modificação que exibe, para cada passo iterativo, a matriz de reflexão de Householder P e também o resultado da matriz Q. Abaixo é possível visualizar, para uma matriz A de dimensão 4×3 o passo a passo do processo iterativo de fatoração. Observe que, logo abaixo, constam as matrizes Q e R calculada diretamente pelo MATLAB, assim como estava no código original. Logo abaixo das matrizes, consta o código modificado.

$$P_1 = \begin{bmatrix} -0.574 & -0.514 & -0.526 & -0.361 \\ -0.514 & 0.832 & -0.172 & -0.118 \\ -0.526 & -0.172 & 0.824 & -0.121 \\ -0.361 & -0.118 & -0.121 & 0.917 \end{bmatrix} \quad Q_1 = \begin{bmatrix} -0.574 & -0.514 & -0.526 & -0.361 \\ -0.514 & 0.832 & -0.172 & -0.118 \\ -0.526 & -0.172 & 0.824 & -0.121 \\ -0.361 & -0.118 & -0.121 & 0.917 \end{bmatrix} \quad A_1 = \begin{bmatrix} -0.849 & -1.27 & -1.18 \\ 0 & -0.07 & -0.218 \\ 0 & 0.222 & 0.2 \\ 0 & 0.387 & 0.114 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.155 & 0.492 & 0.856 \\ 0 & 0.492 & 0.79 & -0.365 \\ 0 & 0.856 & -0.365 & 0.365 \end{bmatrix} \quad Q_2 = \begin{bmatrix} -0.574 & -0.489 & -0.537 & -0.379 \\ -0.514 & -0.315 & 0.317 & 0.733 \\ -0.526 & 0.329 & 0.61 & -0.492 \\ -0.361 & 0.744 & -0.488 & 0.278 \end{bmatrix} \quad A_2 = \begin{bmatrix} -0.849 & -1.27 & -1.18 \\ 0 & 0.451 & 0.23 \\ 0 & 0 & 0.00938 \\ 0 & 0 & -0.218 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.043 & 0.999 \\ 0 & 0 & 0.999 & 0.043 \end{bmatrix} \quad Q_3 = \begin{bmatrix} -0.574 & -0.489 & -0.356 & -0.553 \\ -0.514 & -0.315 & 0.718 & 0.348 \\ -0.526 & 0.329 & -0.518 & 0.589 \\ -0.361 & 0.744 & 0.299 & -0.476 \end{bmatrix} \quad A_3 = \begin{bmatrix} -0.849 & -1.27 & -1.18 \\ 0 & 0.451 & 0.23 \\ 0 & 0 & -0.218 \\ 0 & 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} -0.574 & -0.489 & -0.356 \\ -0.514 & -0.315 & 0.718 \\ -0.526 & 0.329 & -0.518 \\ -0.361 & 0.744 & 0.299 \end{bmatrix} \quad R = \begin{bmatrix} -0.849 & -1.27 & -1.18 \\ 0 & 0.451 & 0.23 \\ 0 & 0 & -0.218 \end{bmatrix}$$

```
%Lucas Carrilho Pessoa - RA094052
%MS512 - Analise Numerica
%Modificacoes a partir do codigo fornecido

% Ilustrando a fatoracao QR via "Householder reflections"
m=12; n=8; %m=4; n=3;
A=rand(m,n);

% calculo da fatorizacao reduzida QR
[Q,R]=qr(A,0);

% matriz Q computada a cada iteracao
Qiterativo = eye(m);
for j=1:n,
    % construcao da "Householder reflection"
    % zerando os elementos da subdiagonal na coluna j
    u=houserefle(A(j:m,j))
    P=eye(m-j+1)-2*u*u'
    % determina matriz de transformacao ortogonal dessa iteracao
    Qj=eye(m);
    Qj(j:m,j:m)=P
    % vai formando, iterativamente, a matriz Q
    Qiterativo = Qiterativo*Qj'
    % aplicando a "Householder transformation"
    A(j:m,j:n)=P*A(j:m,j:n)
    % aguardando digitacao <enter> no teclado pelo usuario
    pause
end
% comparando a solucao matlab R ("[Q,R]=qr(A,0);") com a solucao
% aproximada obtida acima com a fatoracao QR via "Householder reflections"
R
Q
```

5 Algoritmo QR

O algoritmo QR é um dos métodos mais utilizados para o cálculo de autovalores, e seus autovetores associados, de matrizes. A maioria das discussões sobre o algoritmo QR começa com uma versão bem básica do algoritmo e itera sobre essa versão até chegar no método que é amplamente utilizado.

A maioria das discussões começam com as equações fundamentais:

$$A_{k-1} = Q_k Rk, \qquad R_k Q_k = A_k. \tag{5}$$

Em palavras, a partir da iteração $k-1$, faça uma fatoração QR da matriz A_{k-1}, multiplique de forma reversa as matrizes Q e R e faça que $A_k = R_k Q_k$. Essa é a versão básica do algoritmo, e é, aparentemente, simples. Contudo, ela não é muito utilizada nas implementações atuais do algoritmo QR, como consta em [Watkins, 2008]. As versões mais atuais são baseadas em algoritmos de 'eliminação dos bojos' (ou melhor, em inglês, *bulge-chasing algorithms*): a cada iteração começa com uma matriz superior de Hessenberg, faz uma transformação inicial que produz um bojo nas linhas superiores da matriz de Hessenberg e então faz com que esse bojo vá para a parte mais inferior da matriz, retornando a matriz à forma de Hessenberg. Veja na Figura 5 uma simples representação desse fato.

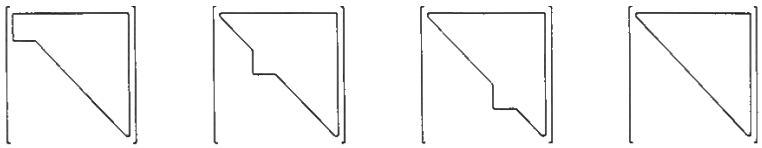

Figura 9: Passo-a-passo das transformações, fazendo com que o bojo, inicialmente posto no topo da matriz de Hessenberg, vá para a parte de baixo, voltando a matriz a sua forma de Hessenberg. Extraído de [Watkins, 2008].

Para entender o funcionamento do algoritmo, e ter uma ideia de sua convergência, temos que apresentar alguns conceitos: transformação similar, método das potências, matrizes superiores de Hessenberg e Subespaços de Krylov.

5.1 Transformação Similar

Matrizes $A \in \mathbb{C}^{n\times n}$ e $B \in \mathbb{C}^{n\times n}$ são similares se existe uma matriz não singular $S \in \mathbb{C}^{n\times n}$ onde $B = S^{-1}AS$. Matrizes similares possuem os mesmos autovalores, e seus autovetores são associados de uma forma simples: se v é um autovetor de A associado com o autovalor λ, então $S^{-1}v$ é um autovetor de B associado a λ. Contudo, o fato importante é que matrizes similares representam a mesma transformação linear em diferentes sistemas de coordenadas.

Um subespaço S de $\mathbb{C}^n$ é dito invariante sobre A se para todo $x \in S$, $Ax \subseteq S$. Se é possível encontrar um subespaço invariante, então é possível dividir o problema de encontrar os autovalores em pedaços menores, como mostra a seguinte proposição:

Proposição 5.1. Seja S um subespaço j-dimensional que é invariante em A, e faça S uma matriz não singular onde as primeiras j colunas são a base de S, e que $B = S^{-1}AS$. Então B é dita bloco-triangular superior:

$$B = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix}$$

onde B_{11} tem dimensão $j \times j$.

Com isso, temos que os autovalores de A são os mesmo que de B, que serão, por sua vez, os mesmos de B_{11} e B_{12} juntos. Com isso, podemos partir para táticas que envolvem divisão e conquista, para determinar os autovalores de A. Então, se é possível encontrar subespaços invariantes, é possível resolver o problema dos autovalores.

5.2 O método das potências

Suponha uma matriz $A \in \mathbb{C}^{n\times n}$ que possui n autovetores v_1, v_2, ..., v_n, associado com os autovalores λ_1, λ_2, ..., λ_n, respectivamente, e assuma que $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. Se $|\lambda_1| > |\lambda_2|$, para uma escolha qualquer de um vetor q, a sequência q, Aq, A^2q, A^3q, ..., se escalada da forma correta, convergirá para um múltiplo de v_1, o autovetor associado ao autovalor dominante λ_1. Em poucas palavras, esse é o método das potências.

Cada vetor em q, Aq, A^2q, A^3q, ... é somente uma representação de um subespaço unidimensional que ele estende. Quando fazemos a escala, só estamos trocando um representante do subespaço por outro. Então,

podemos escrever a sequência de subespaços unidimensionais S, AS, A^2S, A^3S, ... que converge para o autoespaço $\text{span}\{v_1\}$.

Vamos, então, denominar m como um número pequeno (entre 1, 2, 4 ou 6) e escolher m trocas $\rho_1, \ldots, \rho_m \in \mathbb{C}$, de forma que $p(A) = (A-\rho_1 I)(A-\rho_2 I)\cdots(A-\rho_m I)$. $p(A)$ tem os mesmo autovetores que A, e seus correspondentes autovalores $p(\lambda_1), \ldots, p(\lambda_n)$, de forma que $|p(\lambda_1)| \geq \ldots \geq |p(\lambda_n)|$. Então, se $|p(\lambda_k)| > |p(\lambda_{k+1})|$, a iteração de subespaços S, $p(A)S$, $p(A)^2S$, $p(A)^3S$, ... originados por $p(A)$ converge ao subespaço invariante $\text{span}\{v_1, \ldots, v_k\}$. Dado o conjunto de shifts $\rho_1, \ldots, \rho_m$, é possível escolhe-los de forma que a convergência se dê de forma mais rápida $|p(\lambda_{k+1})/p(\lambda_k)| \ll 1$.

Se desejarmos não realizar as iterações dos subespaços S, $p(A)S$, $p(A)^2S$, $p(A)^3S$, ... na prática, teremos que operar em uma base S. Seja $q_1^{(0)}, q_2^{(0)}, \ldots, q_k^{(0)}$ uma base ortonormal de S. Então $p(A)q_1^{(0)}, p(A)q_2^{(0)}, p(A)\ldots, q_k^{(0)}$ é uma base para $p(A)S$, que pode ser transformada na base ortonormal $q_1^{(1)}, q_2^{(1)}, \ldots, q_k^{(1)}$ para $p(A)S$, utilizando alguma versão do processo de Gram-Schmidt. Repetindo o procedimento, podemos obter bases ortonormais $q_1^{(j)}, q_2^{(j)}, \ldots, q_k^{(j)}$ para $p(A)^jS$, com $j = 0, 1, 2, 3, \ldots$. Esse método é chamado de *iterações simultâneas*.

Suponha que somos capazes de encontrar shifts que são boas aproximações para os autovalores de A, ou seja

$$\rho_1 \approx \lambda_n,\ \rho_2 \approx \lambda_{n-1},\ \ldots, \rho_m \approx \lambda_{n-m+1}.$$

Pensemos no caso genérico onde todos os autovalores são distintos, ou seja, cada ρ_i aproxima um autovalor muito bem e nenhum rho_i é perto o suficiente de outro autovalor. Retomando que

$$p(z) = (z - \rho_1 I)(z - \rho_2 I)\cdots(z - \rho_m I)$$

veremos que $p(\rho_i) = 0$, $i = 1, \ldots, m$, então, dessa forma, $|p(\lambda_{n-i+1})|$ é muito pequeno se ρ_i e uma aproximação boa o suficiente para λ_{n-i+1}, $i = 1, \ldots, m$. Os outros $n - m$ valores não serão pequenos, por que não existe ρ_i próximo o suficiente dos outros autovalores de A.

A pergunta central é, como vamos encontrar bons *shifts*? Inicialmente podemos não ter nenhuma ideia de onde os autovalores estão, contudo após algumas iterações alguns autovalores vão começar a aparecer. Com isso, podemos modificar os fatores de *shift* para valores melhores a cada passo iterativo. O custo computacional de performar essas iterações simultâneas em n vetores é $O(n^3)$.

5.3 Matrizes Superiores de Hessenberg

A chave da eficiência é trabalharmos com matrizes de Hessenberg. Uma matriz H é dita superior de Hessenberg se $h_{ij} = 0$ sempre que $i > j+1$. Como elas são próximas a matrizes triangulares, é muito mais barato trabalharmos com elas do que com matrizes completas. Toda matriz $A \in \mathbb{C}^{n\times n}$ é similar a uma matriz na forma superior de Hessenberg. A transformação em uma matriz similar pode ser realizada em $n-1$ reflexões de Householder. Contudo, a complexidade computacional dessa transformação é $O(n^3)$.

A primeira matriz de reflexão tem a forma

$$Q_1 = Q_1^* = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \widetilde{Q}_1 & \\ 0 & & & \end{array}\right]$$

e cria zeros na primeira coluna. Pela forma da matriz Q_1, a primeira transformação deia a primeira linha da matriz transformada A inalterada. A segunda matriz de reflexão tem a forma

$$Q_2 = Q_2^* = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \widetilde{Q}_2 & \\ 0 & 0 & & & \end{array}\right]$$

e cria zeros na segunda coluna, Q_3 na terceira coluna e assim por diante até $n-2$ matrizes de similaridade, onde teremos a matriz superior de Hessenberg. Se fizermos $Q = Q_1 Q_2 \cdots C_{n-2}$ a matriz $B = Q^* A Q$ será superior de Hessenberg.

5.4 Subespaços de Krylov

Seja $x \neq 0$, o j-ésimo subespaço de Krylov associado com A e x, denotado $\kappa_j(A, x)$, é definido por

$$\kappa_j(A, x) = span\{x, Ax, A^2x, \ldots, A^{j-1}x\}.$$

Sejam, também, $e_1, \ldots, e_n$ as colunas de uma matriz identidade $n \times n$. E uma matriz superior de Hessenberg é dita própria superior de Hessenberg se $h_{ij} \neq 0$ onde $i = j + 1$.

Se H é dita própria superior de Hessenberg, então $\kappa_j(H, e_1) = span\{e_1, \ldots, e_j\},\ j = 1, \ldots, n$. Se $x = p(A)e_1$, então $p(A)\kappa_j(A, e_1) = \kappa_j(A, x),\ j = 1, \ldots, n$. Por fim, estabelecemos um link entre subespaços de Krylov e matrizes próprias de Hessenberg: em uma transformação de similaridade para a forma própria superior de Hessenberg, as colunas principais da matriz de transformação geram subespaços de Krylov. Suponha $B = Q^{-1}AQ$ e B é própria superior de Hessenberg. Seja $q_1, \ldots, q_n$ as colunas de Q. Então

$$span\{q_1, \ldots, q_j\} = \kappa_j(A, q_1),\ j = 1, \ldots, n.$$

Agora que os conceitos básicos foram apresentados, iremos melhor comentar o algoritmo QR. Assumimos, sem perda de generalização que a matriz $A \in \mathbb{C}^{n \times x}$ está na forma superior de Hessenberg. A iteração QR de ordem m transforma a matriz A da forma própria superior de Hessenberg, para uma matriz superior de Hessenberg $B = Q^*AQ$. O passo iterativo começa com a escolha de m *shifts* $\rho_1, \ldots, \rho_m$. Uma forma de se escolher esses ρ é determinar os autovalores da sub-matriz inferior-direita de tamanho $m \times m$. Como m é pequeno, essa operação pode ser considerada de complexidade $O(1)$.

Em seguida, um vetor $x = \alpha(A - \rho_1 I)(A - \rho_2 I) \cdots (A - \rho_m I)e_1$ é computado. Essa operação também é barata pois a matriz está na forma superior de Hessenberg. Defina $\widetilde{x} \in \mathbb{C}^{m+1}$ por $x = \begin{bmatrix} \widetilde{x} \\ 0 \end{bmatrix}$. Seja $\widetilde{Q}_0 \in \mathbb{C}^{(m+1)\times(m+1)}$ uma matriz de reflexão tal que $\widetilde{Q}_0 e_1 = \beta\widetilde{x}$ para algum $\beta \neq 0$, e seja

$$Q_0 = \begin{bmatrix} \widetilde{Q}_0 & 0 \\ 0 & I_{n-m-1} \end{bmatrix}$$

então $Q_o e_1 = \beta x$.

Começando uma transformação de similaridade, formaremos uma nova matriz $B_0 = Q_0^* A Q_0$. Pela forma de Q_0, a transformação $A \to Q_0^* A$ recombina somente as primeiras $m + 1$ linhas de A. Isso perturba a forma superior de Hessenberg. Agora, existe um bojo na forma de Hessenberg, como pode ser visto na Figura 5.4.

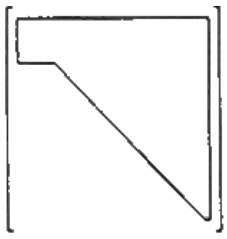

Figura 10: Observe a forma geral da matriz após a primeira transformação. Agora, existe um bojo na parte superior da matriz superior de Hessenberg. Todos os valores fora da região contornada são zeros. Extraído de [Watkins, 2008].

Agora, o método consiste em retornar a matriz para a forma superior de Hessenberg. Cada transformação subsequente remove uma coluna do bojo e adiciona uma linha, movendo o bojo para baixo e para a direita. Eventualmente o bojo é trazido completamente para a parte de baixo da matriz, e então a forma de Hessenberg é restaurada, completando o passo implícito do algoritmo QR. O novo iterante será $B = B_{n-2} = Q^*AQ$, onde $Q = Q_0 Q_1 \cdots Q_{n-2}$. Como $Q_1 e_1 = e_1, \ldots, Q_{n-2} e_1 = e_1$, teremos que $Qe_1 = Q_0 e_1 = \beta x$. Isso faz com que a primeira coluna de Q seja proporcional a x, que será a primeira coluna de $p(A)$. O passo implícito do algoritmo QR é muito menos custoso do que a redução para a forma de Hessenberg a cada iteração, porque cada Q_i usado na eliminação do bojo age apenas em $m + 1$ linhas ou colunas (lembre-se que m é pequeno).

A matriz B é superior de Hessenberg, mas ela pode falhar na propriedade de ser própria superior de Hessenberg. Isso é uma boa notícia, já que quando isso acontece, podemos dividir o problem em dois ou mais sub-problemas

(divisão e conquista). Suponha que B seja própria superior de Hessenberg, nesse caso, as colunas Q geradoras do subespaço de Krylov serão:

$$span\{q_1, \ldots, q_j\} = \kappa_j(A, q_1),\ j = 1, \ldots, n,$$

onde $q_1, \ldots, q_n$ denotam as colunas de Q. Como q_1 é proporcional a $x = p(A)e_1$, deduzimos que $\kappa_j(A, q_1) = \kappa_j(A, x) = p(A)\kappa_j(A, e_1)$. Sabemos ainda que $\kappa_j(A, e_1) = span\{e_1, \ldots, e_j\}$, então

$$p(A)span\{e_1, \ldots, e_j\} = span\{q_1, \ldots, q_j\},\ j = 1, \ldots, n.$$

Essa equação nos mostra que as colunas de Q são o resultado de um passo do processo de *iteração simultânea* orientado por $p(A)$, começando pelos vetores $e_1, \ldots, e_n$. A transformação de similaridade $B = Q^*AQ$ é uma mudança no sistema de coordenadas que transforma cada vetor x em Q^*x. Em resumo, o passo implícito do algoritmo QR afeta uma iteração sob subespaços aninhados, dirigidos por $p(A)$ nos espaços $span\{e_1, \ldots, e_j\},\ j = 1, \ldots, n$. Isso também faz uma mudança no sistema de coordenadas que mapeia os espaços resultantes de volta para $span\{e_1, \ldots, e_j\},\ j = 1, \ldots, n$.

Anteriormente falamos que os autovalores da matriz inferior direita $m \times m$ faz bons *shifts*. Imagine que escolhemos *shifts* $\rho_1, \ldots, \rho_m$ ao acaso e fazemos sucessivas iterações do método QR usando esse mesmo *shift* para produzir a sequência de matrizes A_k. Os passo do algoritmo são efetivas iterações no subespaço dirigido pela matriz $p(A) = \alpha(A - \rho_1 I) \cdots (A - \rho_m I)$.

Suponha, então, que $|p(\lambda_1)| \geq |p(\lambda_2)| \geq \cdots \geq |p(\lambda_n)|$. Para todo j em que $|p(\lambda_j)| > |p(\lambda_{j+1})|$, o subespaço $span\{e_1, \ldots, e_j\}$ fica cada vez mais próximo de ser um subespaço invariante de A_k conforme k cresce. Como $span\{e_1, \ldots, e_j\}$ é invariante sobre A_k. Como $span\{e_i, \ldots, e_j\}$ é invariante sobre A_k se, e somente se

$$A_k = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad A_{11} \in \mathbb{C}^{j \times j} \tag{6}$$

inferimos que a convergência das iterações sobre o subespaço implica na convergência de (A_k) para a forma triangular superior de bloco, mostrada na equação (6).

Agora, consideremos o caso em que $j = n - m$. Se $|p(\lambda_{n-m})| \geq |p(\lambda_{n-m+1})|$, então $span\{e_1, \ldots, e_{n-m}\}$ se aproxima cada vez mais do subespaço invariante correspondente aos autovalores $\lambda_1, \ldots, \lambda_{n-m}$. No limite teremos a configuração proposta pela equação (6), onde os autovalores de A_{11} serão $\lambda_1, \ldots, \lambda_{n-m}$, associados com $span\{e_1, \ldots, e_{n-m}\}$. Assim, os autovalores de A_{22} são $\lambda_{n-m+1}, \ldots, \lambda_n$.

Suponha, agora, que ainda estamos caminhando para a convergência. Nesse caso, teremos

$$A_k = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{7}$$

onde A_{21} tem uma única entrada não nula, que é pequena. Nesse caso, os autovalores de A_{22} não serão $\lambda_{n-m+1}, \ldots, \lambda_n$, mas serão muito próximos desse valor. Nesse momento, tomemos m autovalores de A_{22} como novos *shifts* $\rho_1, \ldots, \rho_m$ para as seguintes iterações. O novo $p(z) = (z - \rho_1)(z - \rho_2) \cdots (z - \rho_m)$ terá $p(\lambda_{n-m+1}), \ldots, p(\lambda_n)$ muito próximos de zero, porque cada um desses λ_j são muito bem aproximados por ρ_i. Contudo, nenhum $p(\lambda_1), \ldots, p(\lambda_{n-m})$ serão pequenos, porque nenhum desses λ_j são bem aproximados pelo *shift* em questão. Após alguma iterações, teremos uma aproximação muito melhor para $\lambda_{n-m+1}, \ldots, \lambda_n$, que poderão ser utilizadas como novos *shifts*, acelerando ainda mais a taxa de convergência.

6 Aproximação para modelos de sistemas de equações não lineares

A principal fonte para elaboração dessa seção é [Burden and Faires, 2010]. Um sistema de equações não lineares tem a forma:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ldots, x_n) &= 0, \\ \vdots & \quad \vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0, \end{aligned}$$

onde cada função f_i pode ser pensada como o mapeamento de um vetor $x = (x_1, x_2, \ldots, x_n)^t$ no espaço $\mathbb{R}^n$ sobre a reta real $\mathbb{R}$. Uma figura que representa um conjunto de duas funções de várias variáveis é apresentada na Figura .

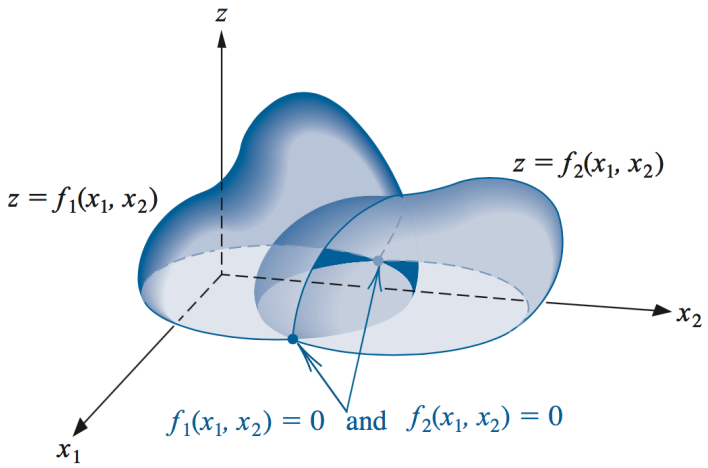

Figura 11: Representação de duas funções não lineares de duas variáveis. Figura extraída de [Burden and Faires, 2010].

6.1 Método de Ponto Fixo

Um sistema de n variáveis pode ser definido por uma função F de mapeamento de $\mathbb{R}^n$ em $\mathbb{R}^n$, como

$$F(x_1, x_2, \ldots, x_n) = (f_1(x_1, x_2, \ldots, x_n), f_1(x_1, x_2, \ldots, x_n), \ldots, f_1 n(x_1, x_2, \ldots, x_n))^t \tag{8}$$

Dessa forma, utilizando notação vetorial, conseguimos escrever o problema como $F(x) = 0$, das funções coordenadas $f_1, f_2, \ldots, f_n$ de F.

Definição 6.1 Seja f uma função definida em um conjunto $D \subset \mathbb{R}^n$ e mapeada em $\mathbb{R}$. A função f é dita ter um limite L em x_0, escrito $\lim_{x \to x_0} f(x) = L$, se, dado qualquer número $\epsilon > 0$, um número $\delta > 0$ existe com $|f(x) - L| < \epsilon$, para qualquer $x \in D$ e $0 < ||x - x_0|| < \delta$.

□

É importante notar que a existência do limite é independente da norma utilizada. O valor específico de δ dependerá da norma escolhida, mas a existência de δ é independente.

Definição 6.2 Seja f uma função definida em um conjunto $D \subset \mathbb{R}^n$ e mapeada em $\mathbb{R}$. A função f é dita contínua em $x_0 \in D$ se $\lim_{x \to x_0} f(x)$ existir e se $\lim_{x \to x_0} f(x) = f(x_0)$. Ainda, f é dita contínua no conjunto D se f é contínua em todos os pontos de D, ou $f \in C(D)$.

□

Agora, definiremos os conceitos de limite e continuidade para funções de $\mathbb{R}^n$ em $\mathbb{R}^n$ levando em conta a sua função de mapeamento de $\mathbb{R}^n$ em $\mathbb{R}$.

Definição 6.3 Seja F a função que leva $D \subset \mathbb{R}^n$ em $\mathbb{R}^n$ da seguinte forma:

$$F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^t$$

onde f_i é um mapeamento de $\mathbb{R}^n$ em $\mathbb{R}$ para todo i. Definimos então

$$\lim_{x \to x_0} F(x) = L = (L_1, L_2, \ldots, L_n)^t$$

se e somente se $\lim_{x \to x_0} f_i(x) = L_i$ para todo $i = 1, 2, \ldots, n$.

□

Agora, vamos enunciar o teorema que que relaciona a continuidade de uma função de n variáveis em um ponto com as derivadas parciais da função nesses pontos.

Teorema 6.4 Seja f uma função de $D \subset \mathbb{R}^n$ em $\mathbb{R}$ e $x_0 \in D$. Suponha que todas as derivadas parciais de f existem e constantes $\delta > 0$ e $K > 0$ existem, tais que, para todo $||x - x_0|| < \delta$ e $x \in D$, temos

$$\left| \frac{\partial f(x)}{\partial x_j} \right| \leq K, \ j = 1, 2, \ldots, n.$$

Então, f é contínua em x_0.

□

Agora que temos bagagem necessária sobre funções de várias variáveis, vamos falar sobre o método do ponto fixo em $\mathbb{R}^n$.

Definição 6.5 Uma função G de $D \subset \mathbb{R}^n$ em $\mathbb{R}^n$ tem um **ponto fixo** em $p \in D$ se $G(p) = p$.

□

O seguinte teorema expande o teorema do método do ponto fixo para casos n-dimensionais.

Teorema 6.6 Seja $D = \{(x_1, x_2, \ldots, x_n)^t | a_i \leq x_i \leq b_i, \ i = 1, 2, \ldots, n\}$ para um conjunto de constantes $a_1, a_2, \ldots, a_n$ e $b_1, b_2, \ldots, b_n$. Suponha que G é uma função contínua de $D \subset \mathbb{R}^n$ em $\mathbb{R}^n$ com a propriedade que $G(x) \in D$ para todo $x \in D$. Então G tem um ponto fixo em D.

Ainda, suponha que todos os componentes da função G tem derivadas parciais contínuas e existe uma constante $K < 1$ que

$$\left| \frac{\partial g_i(x)}{\partial x_j} \right| \leq \frac{K}{n}, \ \forall x \in D$$

para todo $j = 1, 2, \ldots, n$ e para cada componente da função g_i. Então, a sequência $\{x^{(k)}\}_{k=0}^{\infty}$ definida por um $x^{(0)}$ arbitrário em D e gerada por

$$x^{(k)} = G(x^{(k-1)}), \forall k \geq 1$$

converge para um único ponto fixo $p \in D$ e

$$||x^{(k)} - p||_\infty \leq \frac{K^k}{1 - k} ||x^{(1)} - x^{(0)}||_\infty.$$

□

Dado as definições e teoremas acima apresentados, temos a implicação matemática que o método exige que todas as funções $f_1, f_2, \ldots, f_n$ sejam diferenciáveis em torno de x_0. Além disso, o sistema precisa ser posto na forma de ponto fixo, ou seja, a j-ésima função precisa ser escrita na forma $x_j = f(x_1, x_2, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$, o que pode não ser uma tarefa simples, bem como, devemos escrever as derivadas parciais para cada uma das equações, mostrando que há um único ponto fixo dentro do intervalo selecionado para as iterações.

Para cada passo iterativo, deve-se avaliar todas as funções a respeito do vetor x^k, até que haja a convergência para $|x^{(k)} - x^{(k-1)}|_\infty < \epsilon$.

6.2 Método de Newton

Para construir o método de ponto-fixo, encontramos uma função ϕ com a propriedade que $g(x) = x - \phi(x)f(x)$, nos dando convergência quadrática para o ponto fixo p para a função g. Disso, o método de Newton está envolvido em encontrar uma função $\phi(x) = \frac{1}{f'(x)},\ f'(x) \neq 0$. Em um caso n-dimensional, envolveremos a matriz

$$A(x) = \begin{bmatrix} a_{11}(x) & a_{12}(x) & \cdots & a_{1n}(x) \\ a_{21}(x) & a_{22}(x) & \cdots & a_{2n}(x) \\ \vdots & \vdots & & \vdots \\ a_{n1}(x) & a_{n2}(x) & \cdots & a_{nn}(x) \end{bmatrix} \tag{9}$$

onde cada entrada $a_{ij}(x)$ é uma função $\mathbb{R}^n$ em $\mathbb{R}$. Isso requer que $A(x)$ seja encontrado de forma que

$$G(x) = x - A(x)^{-1}F(x)$$

nos dando uma convergência quadrática para a solução de $F(x) = 0$, assumindo que $A(x)$ é não singular no ponto fixo p em G.

Teorema 6.7 Seja p uma solução de $G(x) = x$. Suponha que existe um número $\delta > 0$ que:

(i) $\frac{\partial g_i}{\partial x_j}$ é contínua em $N_\delta = \{x|\ ||x - p|| < \delta\},\ \forall i = 1, 2, \ldots, n;\ \forall j = 1, 2, \ldots, n;$

(ii) $\frac{\partial^2 g_i(x)}{(\partial x_j \partial x_k)}$ é contínua e $|\frac{\partial^2 g_i(x)}{(\partial x_j \partial x_k)}| \leq M$ para alguma constante M, para todo $x \in N_\delta,\ \forall i = 1, 2, \ldots, n;\ \forall j = 1, 2, \ldots, n;\ \forall k = 1, 2, \ldots, n;$

(iii) $\frac{\partial g_i(p)}{\partial x_k} = 0\ \forall i = 1, 2, \ldots, n;\ \forall k = 1, 2, \ldots, n.$

Então existe um número $\hat{\delta} \leq \delta$ que a sequência gerada por $x^{(k)} = G(x^{(k-1)})$ converge quadraticamente para p para qualquer escolha de $x^{(0)}$, de forma que $||x^{(0)} - p|| < \hat{\delta}$. Ou ainda

$$||x^{(k)} - p||_\infty \leq \frac{n^2 M}{2}||x^{(k-1)} - p||_\infty^2,\ \forall k \geq 1.$$

□

Para aplicar o teorema 6.7, suponha que $A(x)$ é uma matriz $n \times n$ de funções de $\mathbb{R}^n$ em $\mathbb{R}$ na forma da equação (9) com entradas especificamente escolhidas. Assumindo também que $A(x)$ é não singular próximo da solução p de $F(x) = 0$. Utilizaremos para a matriz A a matriz Jacobiana, definida como

$$J(x) = A(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \frac{\partial f_1}{\partial x_2}(x) & \cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \frac{\partial f_2}{\partial x_1}(x) & \frac{\partial f_2}{\partial x_2}(x) & \cdots & \frac{\partial f_2}{\partial x_n}(x) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(x) & \frac{\partial f_n}{\partial x_2}(x) & \cdots & \frac{\partial f_n}{\partial x_n}(x) \end{bmatrix}$$

pois essa matriz satisfaz a condição (iii) do teorema. Com isso, a função G é definida por

$$G(x) = x - J(x)^-1F(x)$$

e a função de iteração, envolvendo para um selecionado $x^{(0)}$ e gerando, para $k \geq 1$,

$$x^{(k)} = G(x^{(k-1)}) = x^{(k-1)} - J(x^{(k-1)})^{-1}F(x^{(k-1)}$$

chegando, então ao método de Newton para sistemas não lineares, tendo uma convergência quadrática esperada, desde que $J(p)^{-1}$ exista. Contudo, uma fraqueza do método de Newton é que precisamos determinar a matriz inversa de $J(x)$ a cada iteração. Contudo, na prática, podemos excluir essa operação fazendo a operação em dois passos: primeiro, um vetor y é encontrado de forma a satisfazer $J(x^{(k-1)})y = -F(x^{(k-1)})$. Então, uma nova aproximação para $x^{(k)}$ é obtida adicionando y a $x^{(k-1)}$.

Com isso, temos o seguinte algoritmo, sendo $F(x)$ um conjunto de n equações e n variáveis, uma aproximação inicial $x = (x_1, \ldots, x_n)^t$, a tolerância TOL de parada e o número máximo de iterações N:

MÉTODO DE NEWTON$(F(x), x, TOL, N)$

1 $k = 1$
2 **while** $(k \leq N)$
3 Calcule $F(x)$ e $J(x)$, onde $J(x)_{ij} = (\partial f_i(x)/\partial x_j)$ para $1 \leq i,\ j \leq n$
4 Resolva o sistema linear $J(x)y = -F(x)$
5 $x = x + y$
6 **if** $||y|| < TOL$ **return** x
7 $k = k + 1$

Pelo método apresentado, temos as implicações matemáticas do cálculo da matriz jacobiana, que envolve n^2 derivadas parciais, e que pode ser muito custoso dependendo do tamanho do problema, além de haver a garantia que de existem todas essas derivadas parciais. Além disso, deveríamos, para cada passo iterativo, calcular a matriz inversa de $J(x)$. Contudo o método descrito isenta esse cálculo, resolvendo o sistema $J(x)y = -F(x)$.

Em termos de complexidade computacional é necessário, ao menos n^2 avaliações escalares de funções para aproximar a matriz Jacobiana em um determinado passo iterativo, mais n para avaliar a função F, e geralmente $O(n^3)$ operações para resolver o sistema linear envolvendo o Jacobiano aproximado.

6.3 Método de Broyden

O método de Newton tem a desvantagem de necessitar a determinação da matriz Jacobiana $J(x)$ para cada iteração. Contudo, na prática, podemos utilizar uma aproximação por diferenças finitas para as derivadas parciais, por exemplo:

$$\frac{\partial f_j}{\partial x_k}(x^{(i)}) \approx \frac{f_j(x^(i) + e_k h) - f_j(x^{(i)})}{h}$$

onde h é pequeno em módulo e e_k é um vetor cuja única entrada não nula é 1 na k-ésima coordenada.

O método de Broyden requer apenas n avaliações de função por iteração e reduz a complexidade geral para $O(n^2)$ por iteração. Ele está contido em uma classe de método denominada *Métodos Quasi-Newton*. Ele basicamente troca a matriz Jacobiana do método de Newton por uma aproximação, que é facilmente atualizada a cada iteração.

A desvantagem, em relação ao método de Newton, é que perdemos a propriedade de convergência quadrática, que agora passa a ser superlinear, implicando que

$$\lim_{i \to \infty} \frac{||x^{(i+1)} - p||}{||x^{(i)} - p||} = 0$$

onde p denota a solução para $F(x) = 0$ e $x^{(i)}$ e $x^{(i+1)}$ são aproximações consecutivas para p. Outra desvantagem é que o método não é auto-corretivo.

Para descrever o método, vamos partir de uma aproximação inicial $x^{(0)}$ que é uma solução de p quando $F(x) = 0$. Calculando a próxima iteração $x^{(1)}$ da mesma forma que pelo método de Newton. Para calcular $x^{(2)}$ partimos do método de Newton e examinamos o método das secantes para uma equação não linear. O método usa a aproximação

$$f'(x_1) \approx \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

como uma troca de $f'(x_1)$ no método de Newton.

Para sistemas não lineares $x^{(1)} - x^{(0)}$ é um vetor, e então trocamos a matriz J pela matriz A com a seguinte propriedade:

$$A_1(x^{(1)} - x^{(0)}) = F(x^{(1)}) - F(x^{(0)}).$$

Com isso, após algumas definições, uma vez que $x^{(i)}$ já foi determinado, $x^{(i+1)}$ será computado por

$$A_i = A_{i-1} + \frac{y_i - A_{i-1}s_i}{||s_i||_2^2} s_i^t \tag{10}$$

e

$$x^{(i+1)} = x^{(i)} - A_1^{-1}F(x^{(i)}) \tag{11}$$

onde $y_i = F(x^{(i)}) - F(x^{(i-1)})$ e $s_i = x^{(i)} - x^{(i-1)}$.

Abaixo é apresentado a Fórmula de Sherman-Morrison, que é uma melhoria que pode ser incorporada ao método, ao se aplicar a sua inversão de matriz.

Teorema 6.8 (Fórmula de Sherman-Morrison) Suponha que A é uma matriz não singular e que x e y são vetores tais que $y^tA^{-1}x \neq -1$. Então, $A + xy^t$ é não singular e

$$(A + xy^t)^{-1} = A^{-1} - \frac{A^{-1}xy^tA^{-1}}{1 + y^tA^{-1}x}$$

□

Com isso, podemos calcular A_i^{-1} diretamente a partir de A_{i-1}^{-1}, eliminando a necessidade de se calcular uma inversão a cada iteração.

Fazendo $A = A_{i-1}$, $x = (y_i - A_{i-1}s_i)/||s_i||_2^2$ e $y = s_i$ na equação (10) temos que

$$A_i^{-1} = A_{i-1}^{-1} + \frac{(s_i - A_{i-1}^{-1}y_i)s_i^tA_{i-1}^-1}{s_i^tA_{i-1}^{-1}y_i} \tag{12}$$

Esse cálculo envolve somente multiplicações matriz-vetor a cada passo iterativo, totalizando, apenas $O(n^2)$ cálculos aritméticos. O seguinte algoritmo apresenta o passo a passo, incorporando a equação 12. Seja $F(x)$ um conjunto de n equações e n variáveis, uma aproximação inicial $x = (x_1, \ldots, x_n)^t$, a tolerância TOL de parada e o número máximo de iterações N:

MÉTODO DE BROYDEN$(F(x), x, TOL, N)$

1 $A_0 = J(x)$ onde $J(x)_{ij} = (\partial f_i(x)/\partial x_j)$ para $1 \leq i,\ j \leq n$
2 $v = F(x^{(0)})$
3 $A = A_0^{-1}$
4 $s = -Av$ onde $s = s_1$
5 $x = x + s$ onde $x = x^{(1)}$
6 $k = 2$
7 **while** $(k \leq N)$
8 $\quad w = v$
9 $\quad v = F(x)$, note que $v = F(x^{(k)})$
10 $\quad y = v - w$, note que $y = y_k$
11 $\quad z = -Ay$, note que $z = -A_{k-1}^{-1}y_k$
12 $\quad p = -s^tz$, note que $p = s_k^tA_{k-1}^{-1}y_k$
13 $\quad u^t = s^tA$
14 $\quad A = A + \frac{1}{p}(s + z)u^t$, note que $A = A_k^{-1}$
15 $\quad s = -Av$, note que $s = -A_k^{-1}F(x^{(k)})$
16 $\quad x = x + s$, note que $x = x^{(k+1)}$
17 $\quad$ **if** $||s|| < TOL$ **return** x
18 $\quad k = k + 1$

O método ainda necessita a determinação da primeira matriz Jacobiana. Com isso, é necessário que todas as funções não lineares sejam parcialmente diferenciáveis. Porém, dado o primeiro cálculo, as iterações não necessitam de mais determinações.

Referências

[Burden and Faires, 2010] Burden, L. and Faires, J. D. (2010). *Numerical Analysis*. Cengage Learning, 9th edition.

[Demmel, 1997] Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. Society for Industrial and Aplied Mathematics, Philadelphia, PA, USA.

[G. H. Golub and Loan, 1979] G. H. Golub, S. N. and Loan, C. V. (1979). A hessenberg-schur method for the problem $ax + xb = c$. *IEEE Transactions on Automatic Control*, AC-24(6):909–913.

[Gene F. Franklin and Emami-Naeini, 2013] Gene F. Franklin, J. D. P. and Emami-Naeini, A. (2013). *Sistemas de Controle para Engenharia*. Bookman, 6th edition.

[Golub and Van Loan, 2013] Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, USA, 4th edition.

[Higham, 2014] Higham, N. J. (2014). Sylvester's influence on applied mathematics. *Mathematics Today*, pages 202–206.

[Horn and Johnson, 2013] Horn, R. A. and Johnson, C. R. (2013). *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 2nd edition.

[Pulino, 2015] Pulino, P. (2015). *Algebra Linear e suas Aplicações*. IMECC, UNICAMP, Campinas, Brasil.

[Sylvester, 1884] Sylvester, J. J. (1884). Sur l'equation en matrices $px = xq$. *Comptes Rendus de l'Académie des Sciences*, pages 67–71 et 115–116.

[Trefethen and III, 1997] Trefethen, L. N. and III, D. B. (1997). *Numerical Linear Algebra*. SIAM.

[Watkins, 2008] Watkins, D. S. (2008). The qr algorithm revisited. *SIAM*, 50(1):133–145.

[Watkins, 2010] Watkins, D. S. (2010). *Fundamentals of matrix computations*. Pure and applied mathematics. Wiley-Interscience, New York, 3rd edition.

[Wheeler, 2013] Wheeler, R. (2013). Investigation in mathematics singular value decomposition.