SVD e o algoritmo Golub-Reinsch

João Vitor de Oliveira Silva

University of Colorado - Denver

27 de Maio de 2024

Apresentação

- Bacharel em Ciência da Computação (UFRJ, 2013-2018)
- Mestre em Modelagem Computacional (LNCC, 2018-2021)
- Professor Substituto (UFRJ/IC, 2020-2022)
- Programador Científico (PUC-Rio/LMMP, 2022-2023)
- Doutorando em Matemática Aplicada
  (University of Colorado - Denver, 2023-202Y, $7 \leq Y \leq 9$)

Introdução

Problemas principais da Álgebra Linear Computacional:

1. Solução de sistemas lineares: $A\mathbf{x} = \mathbf{b}$
2. Solução aproximada de sistemas lineares: $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2$
3. Solução de problemas de autovalor: $A\mathbf{x} = \lambda\mathbf{x}$
4. Solução de problemas de valores singulares: $A\mathbf{v} = \sigma\mathbf{u}$

Normalmente é realizada uma **decomposição** da matriz A para resolver os problemas listados.

Introdução

Problemas principais da Álgebra Linear Computacional:

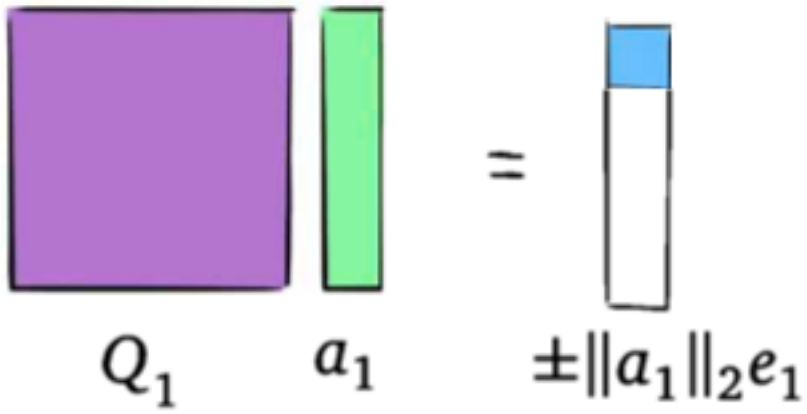
3. Solução de problemas de autovalor: $A\mathbf{x} = \lambda\mathbf{x}$
   i. A é uma matriz não simétrica (LAPACK: *dgeev*)
   ii. A é uma matriz simétrica (LAPACK: *dsyev*)
4. Solução de problemas de valores singulares: $A\mathbf{v} = \sigma\mathbf{u}$ (LAPACK: *dgesvd)*

Preliminares: Reflexão de Householder

- Reflexões que possuem o objetivo de introduzir zeros em um determinado vetor, **sem alterar sua norma.**

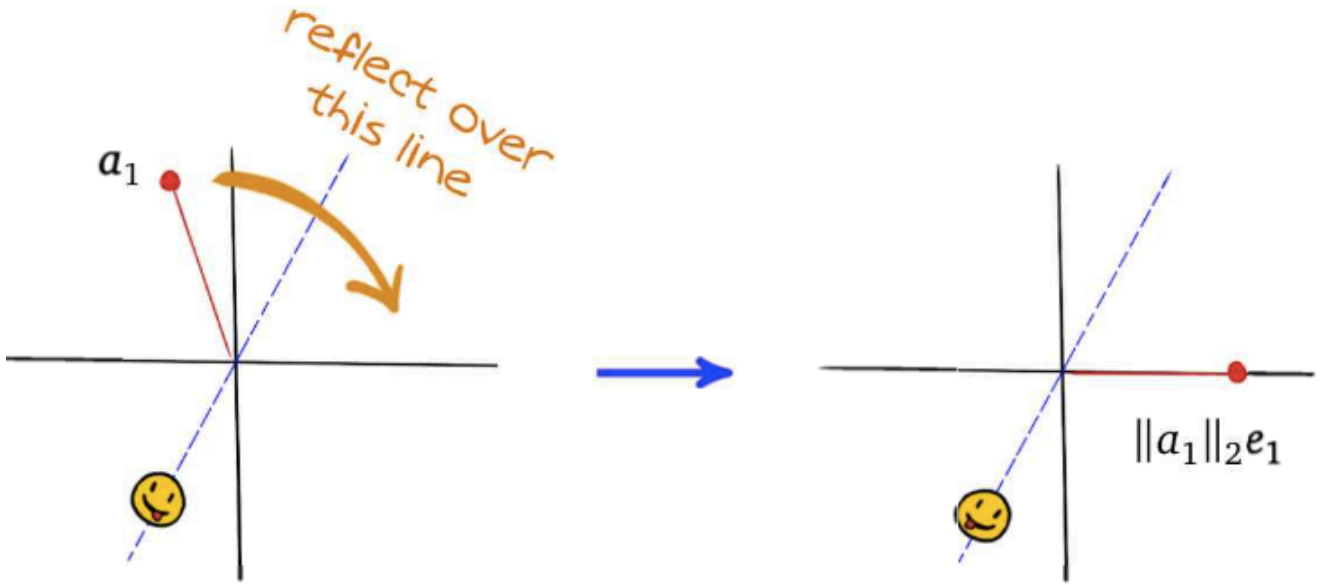
Preliminares: Reflexão de Householder

- Reflexões que possuem o objetivo de introduzir zeros em um determinado vetor, **sem alterar sua norma.**

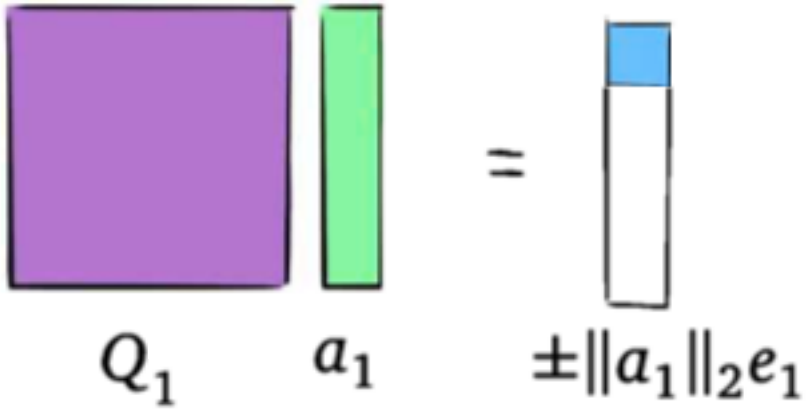


Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Preliminares: Reflexão de Householder



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

$$\mathbf{v} = \mathbf{a}_1 - \|\mathbf{a}_1\|\mathbf{e}_1$$

$$Q_1 = I - \beta \mathbf{v}\mathbf{v}^T$$

$$\text{onde } \beta = \frac{2}{\mathbf{v}^T\mathbf{v}}$$

Preliminares: Reflexão de Householder

- Aplicação direta: é possível iterar esse processo, obtendo uma matriz triangular superior no final (fatoração QR)

$$\begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix} \xrightarrow{Q_1} \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix} \xrightarrow{Q_2} \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} \xrightarrow{Q_3} \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix}$$

onde $$Q_2 = \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & & & \\ 0 & & \widetilde{Q}_2 & \\ 0 & & & \end{array}\right), \quad Q_3 = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & & \\ 0 & 0 & & \widetilde{Q}_3 \end{array}\right)$$

Preliminares: Reflexão de Householder

- Aplicação direta: é possível iterar esse processo, obtendo uma matriz triangular superior no final (fatoração QR)

$$\overset{A}{\begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}} \xrightarrow{Q_1} \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix} \xrightarrow{Q_2} \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} \xrightarrow{Q_3} \overset{R}{\begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{pmatrix}}$$

onde $Q_2 = \left(\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline 0 & & & \\ 0 & & \widetilde{Q}_2 & \\ 0 & & & \end{array}\right)$, $Q_3 = \left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & & \\ 0 & 0 & & \widetilde{Q}_3 \end{array}\right)$

Fatoração QR:

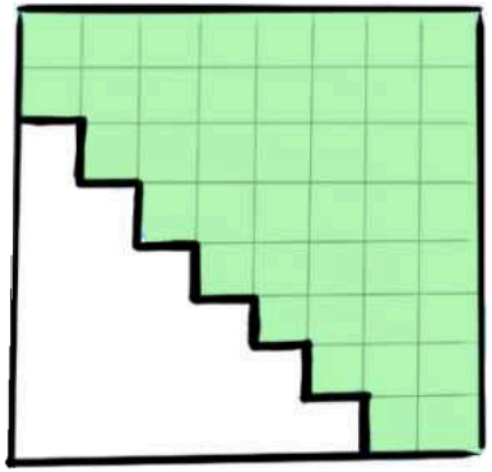
$$Q = Q_1^T Q_2^T Q_3^T,$$
$$A = QR$$

Preliminares: *Givens Rotations*

- Rotações que produzem exatamente um zero em um determinado vetor, **sem alterar a sua norma.**

Preliminares: *Givens Rotations*

- Rotações que produzem exatamente um zero em um determinado vetor, **sem alterar a sua norma.**
- Geralmente usado sobre matrizes Hessenberg:

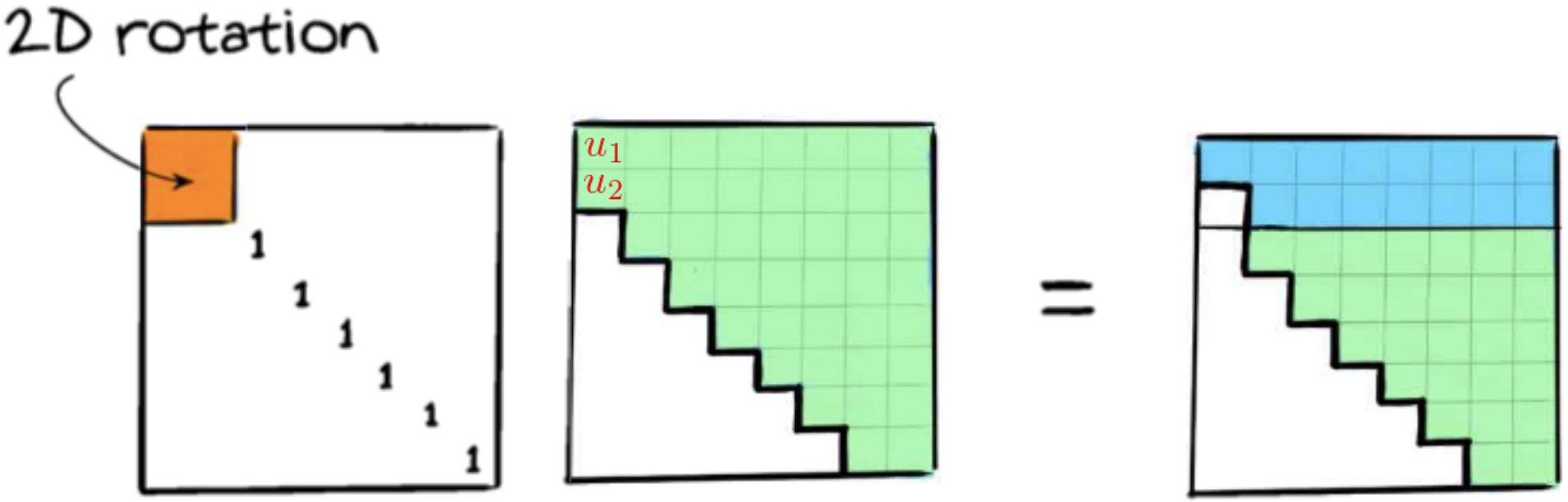


A

Matriz Hessenberg, “quase triangular superior”.
Veremos mais detalhes sobre adiante.

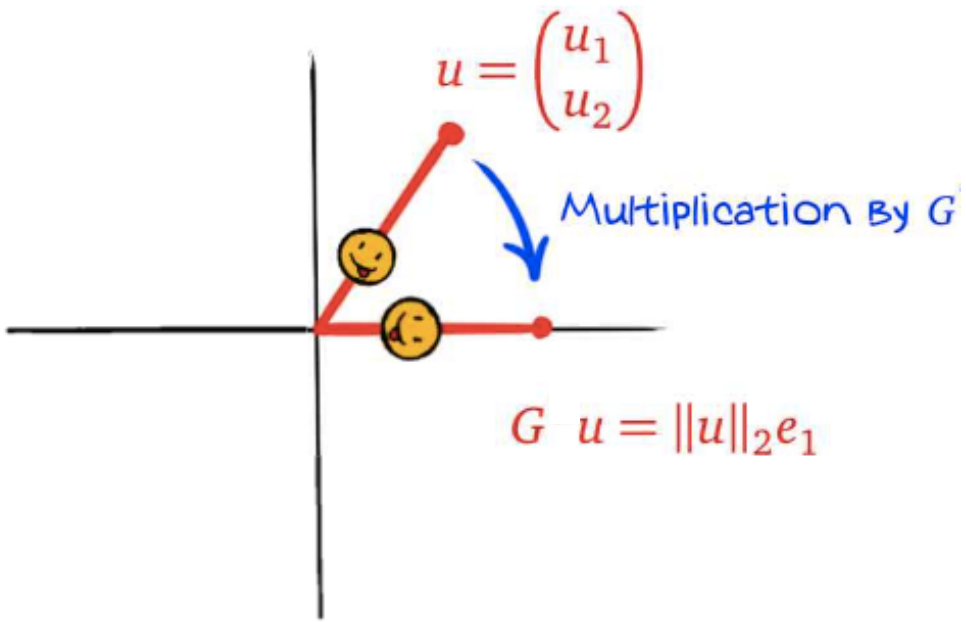
Preliminares: *Givens Rotations*

- Propriedade: uma *Givens Rotations* age apenas sobre duas linhas de uma matriz



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Preliminares: *Givens Rotations*



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

$$G = \left(\begin{array}{cc|ccc} c & s & 0 & \cdots & 0 \\ -s & c & 0 & \cdots & 0 \\ \hline 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{array}\right)$$

$$c = \frac{u_1}{\|u\|}\ , s = -\frac{u_2}{\|u\|}$$

Preliminares: Forma de Schur

- Para cálculo de autovalores de uma matriz quadrada, obter a forma de Schur de uma matriz é uma possibilidade (autovalores na diagonal principal)

Preliminares: Forma de Schur

- Para cálculo de autovalores de uma matriz quadrada, obter a forma de Schur de uma matriz é uma possibilidade (autovalores na diagonal principal)

$$A = STS^H$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & s_{1n} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2n} \\ s_{31} & s_{32} & s_{33} & \cdots & s_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & s_{n3} & \cdots & s_{nn} \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ 0 & t_{22} & t_{23} & \cdots & t_{2n} \\ 0 & 0 & t_{33} & \cdots & t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & t_{nn} \end{bmatrix} \begin{bmatrix} \overline{s_{11}} & \overline{s_{21}} & \overline{s_{31}} & \cdots & \overline{s_{n1}} \\ \overline{s_{12}} & \overline{s_{22}} & \overline{s_{32}} & \cdots & \overline{s_{n2}} \\ \overline{s_{13}} & \overline{s_{23}} & \overline{s_{33}} & \cdots & \overline{s_{n3}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{s_{1n}} & \overline{s_{2n}} & \overline{s_{3n}} & \cdots & \overline{s_{nn}} \end{bmatrix}$$

Preliminares: Forma de Schur

$$A = STS^H$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & s_{1n} \\ s_{21} & s_{22} & s_{23} & \cdots & s_{2n} \\ s_{31} & s_{32} & s_{33} & \cdots & s_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & s_{n3} & \cdots & s_{nn} \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ 0 & t_{22} & t_{23} & \cdots & t_{2n} \\ 0 & 0 & t_{33} & \cdots & t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & t_{nn} \end{bmatrix} \begin{bmatrix} \overline{s_{11}} & \overline{s_{21}} & \overline{s_{31}} & \cdots & \overline{s_{n1}} \\ \overline{s_{12}} & \overline{s_{22}} & \overline{s_{32}} & \cdots & \overline{s_{n2}} \\ \overline{s_{13}} & \overline{s_{23}} & \overline{s_{33}} & \cdots & \overline{s_{n3}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \overline{s_{1n}} & \overline{s_{2n}} & \overline{s_{3n}} & \cdots & \overline{s_{nn}} \end{bmatrix}$$

- Problema: Não há algoritmo para se obter esta forma em número finito de passos (teorema de Abel)

Preliminares: Forma de Hessenberg

- Há um algoritmo com **número finito de passos** para obter forma de Hessenberg, “quase triangular superior”

Preliminares: Forma de Hessenberg

- Há um algoritmo com **número finito de passos** para obter forma de Hessenberg, “quase triangular superior”

$$A = QHQ^T$$

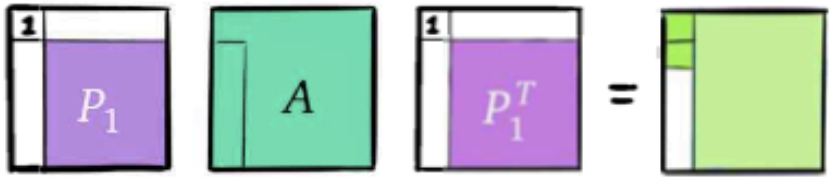
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & \cdots & q_{1n} \\ q_{21} & q_{22} & q_{23} & \cdots & q_{2n} \\ q_{31} & q_{32} & q_{33} & \cdots & q_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & q_{n3} & \cdots & q_{nn} \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & \cdots & h_{3n} \\ \vdots & 0 & h_{43} & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n,n-1} & h_{nn} \end{bmatrix} \begin{bmatrix} q_{11} & q_{21} & q_{31} & \cdots & q_{n1} \\ q_{12} & q_{22} & q_{32} & \cdots & q_{n2} \\ q_{13} & q_{23} & q_{33} & \cdots & q_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{1n} & q_{2n} & q_{3n} & \cdots & q_{nn} \end{bmatrix}$$

Preliminares: Forma de Hessenberg

Step 1: Find Q_1 to put zeros in the first column:

To do this, find the Householder reflection P_1 so that $P_1 v = \|v\| e_1$.

Observe that $Q_1 A Q_1^T$ has the property we want!



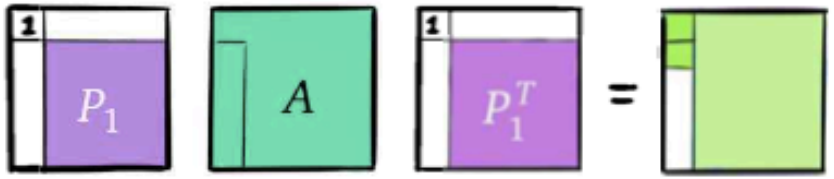
Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Preliminares: Forma de Hessenberg

Step 1: Find Q_1 to put zeros in the first column:

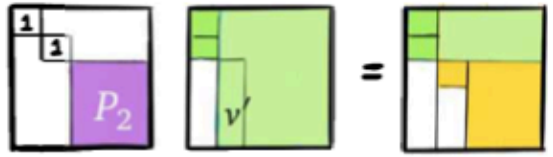
To do this, find the Householder reflection P_1 so that $P_1 v = \|v\| e_1$.

Observe that $Q_1 A Q_1^T$ has the property we want!

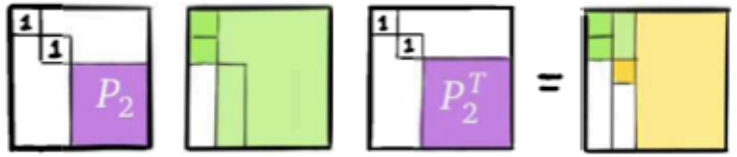


Step 2: Find Q_2 to put zeros in the second column:

To do this, find the Householder reflection P_2 so that $P_2 v' = \|v'\| e_1$.



Observe that $Q_2 Q_1 A Q_1^T Q_2^T$ has the property we want!



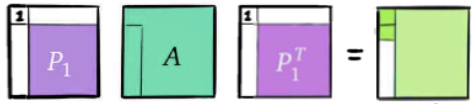
Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Preliminares: Forma de Hessenberg

Step 1: Find Q_1 to put zeros in the first column:

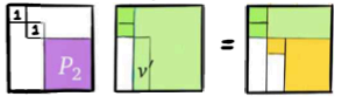
To do this, find the Householder reflection P_1 so that $P_1 v = \|v\| e_1$.

Observe that $Q_1 A Q_1^T$ as the property we want!

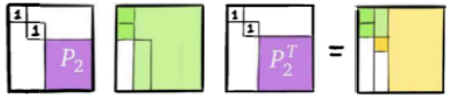


Step 2: Find Q_2 to put zeros in the second column:

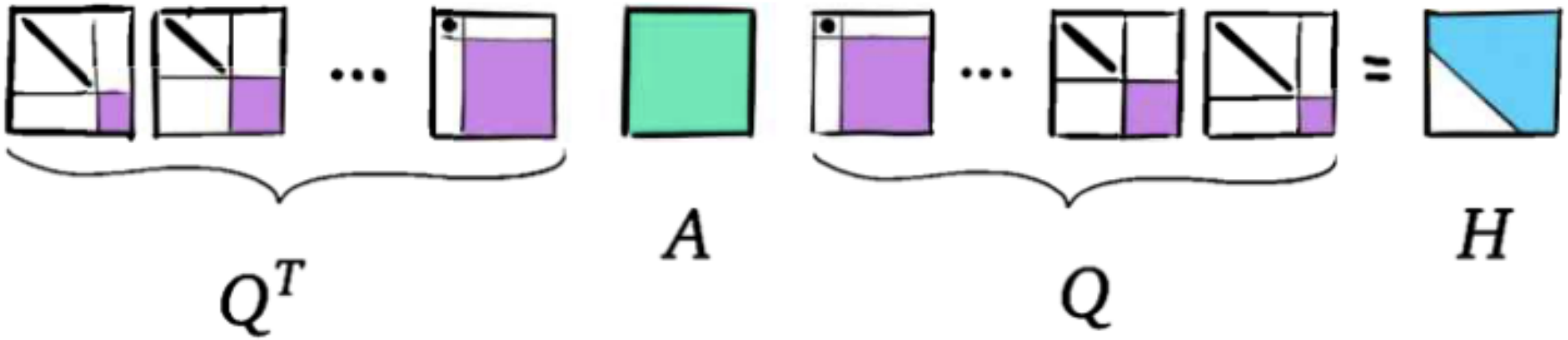
To do this, find the Householder reflection P_2 so that $P_2 v' = \|v'\| e_1$.



Observe that $Q_2 Q_1 A Q_1^T Q_2^T$ has the property we want!



Finally:



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Preliminares: Forma de Hessenberg

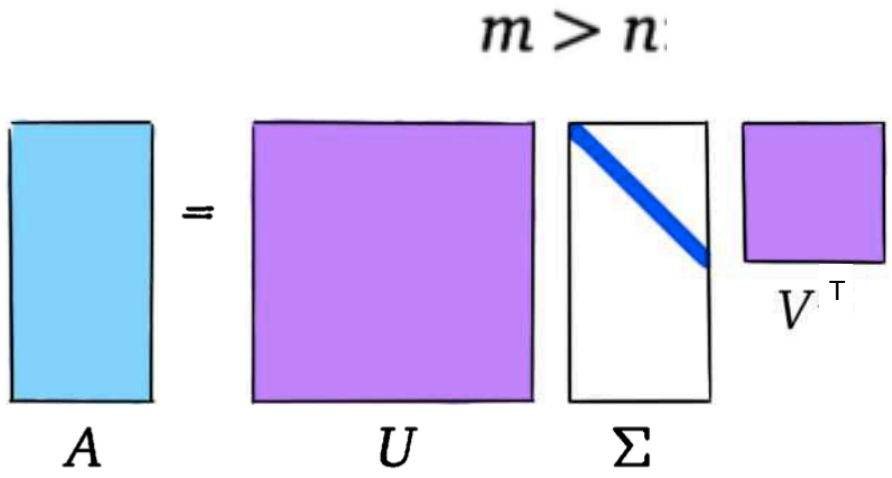
- Uma propriedade muito útil da forma de Hessenberg é que ela é preservada durante o algoritmo QR (principal algoritmo usado para cálculo de autovalores)

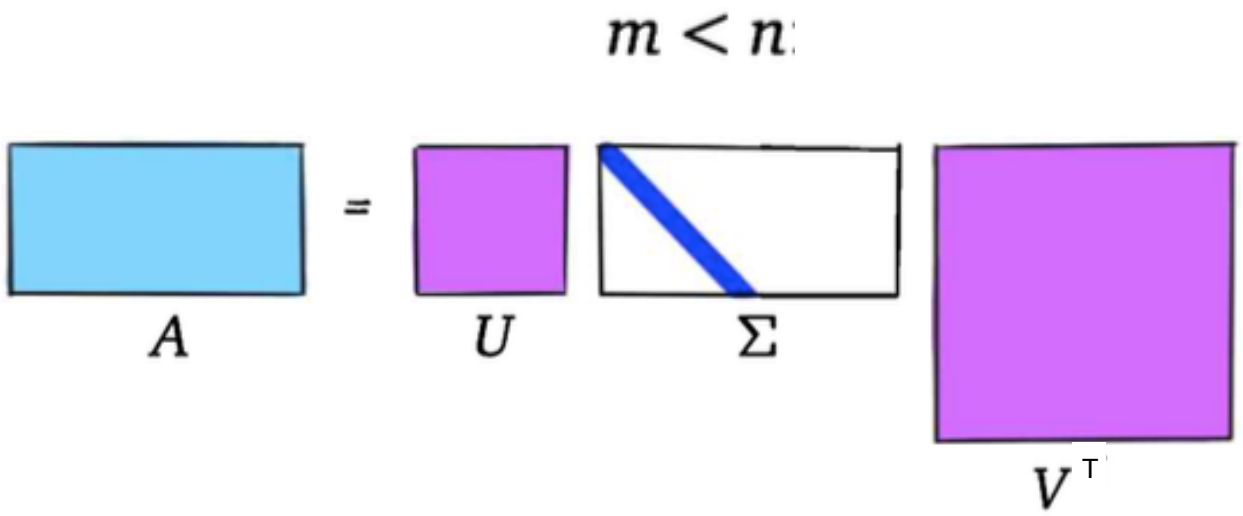
$$H^{(0)} = Q^{(0)} R^{(0)}$$
$$H^{(1)} = (Q^{(0)})^T H^{(0)} Q^{(0)} = R^{(0)} Q^{(0)}$$

Também é matriz Hessenberg!

Preliminares: decomposição em valores singulares (SVD)

- Sempre que nos referirmos a decomposição SVD de uma matriz retangular m x n, estaremos nos referindo ao SVD completo (*full* SVD)





Preliminares: decomposição em valores singulares (SVD)

- Propriedade: se uma matriz for multiplicada pela esquerda ou pela direita por uma matriz ortogonal, **seus valores singulares não são alterados**

$$QA = QU\Sigma V^T = \tilde{U}\Sigma V^T \qquad AQ = U\Sigma V^T Q = U\Sigma \tilde{V}^T$$

O algoritmo Golub-Kahan (finalmente!)

- Algoritmo dividido em 2 grandes etapas
  1. **Bidiagonalização** da matriz A de entrada em $A = P_l B P_r^T$, onde P_l e P_r são matrizes ortogonais e B é uma matriz bidiagonal

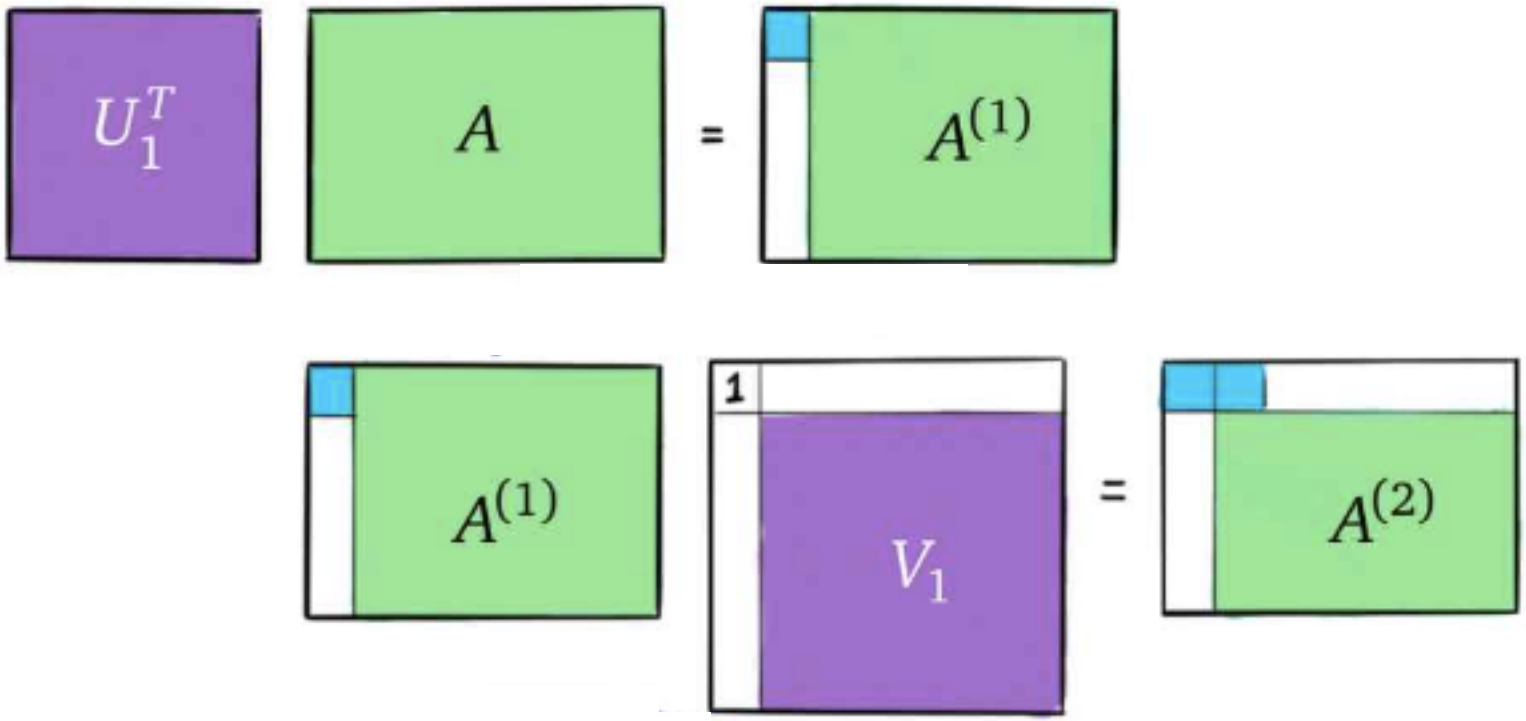
O algoritmo Golub-Kahan (finalmente!)

- Algoritmo dividido em 2 grandes etapas
  1. **Bidiagonalização** da matriz A de entrada em $A = P_l B P_r^T$, onde P_l e P_r são matrizes ortogonais e B é uma matriz bidiagonal (de mesmo tamanho que A)
  2. Aplicação de repetida de um passo do algoritmo **Golub-Kahan** (algoritmo QR com shifts/algoritmo QR de Francis aplicado implicitamente sobre $B^T B$) até convergência

Bidiagonalização

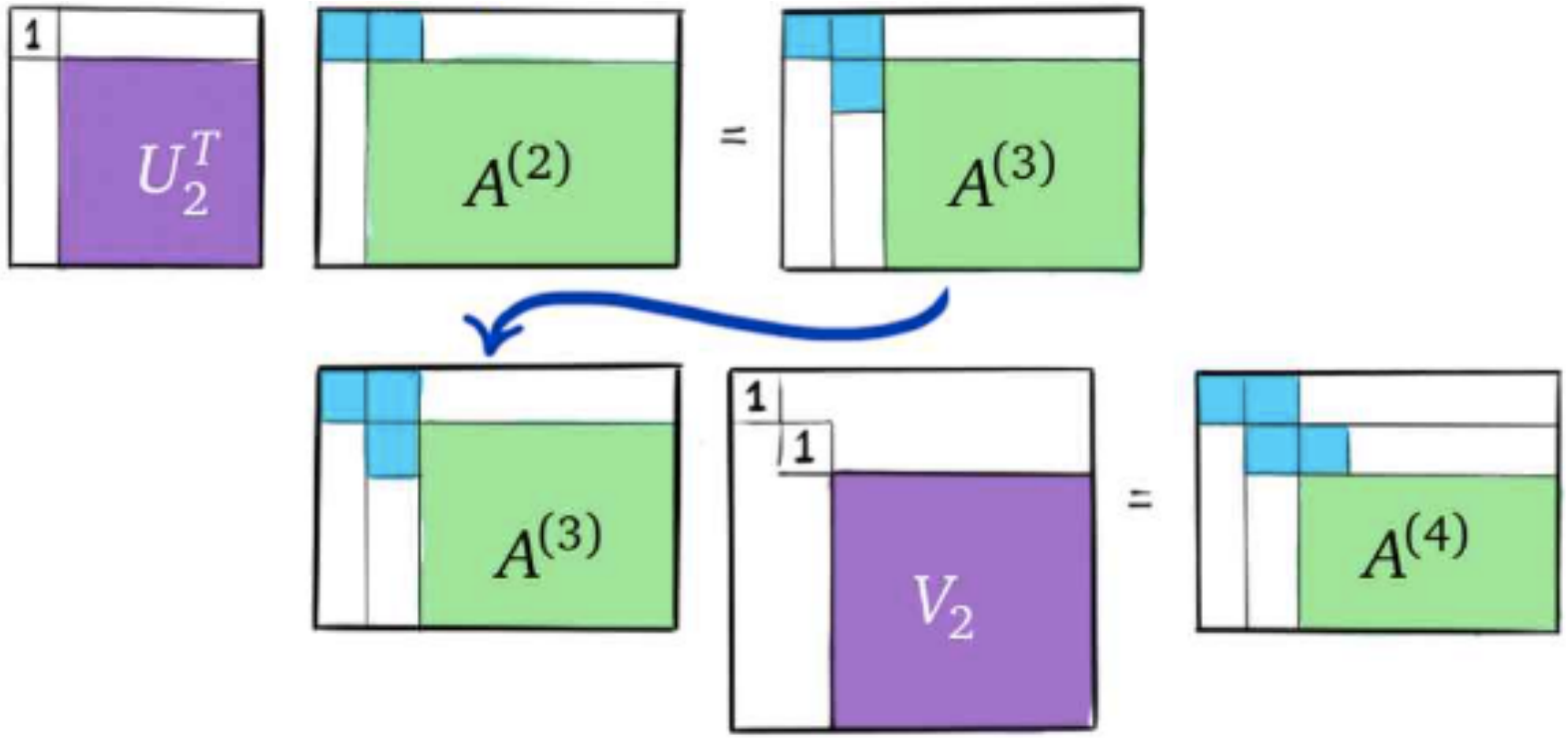
- Algoritmo baseado na redução na forma de Hessenberg, mas vai além por agir sobre colunas e linhas
- Número finito de passos

Bidiagonalização



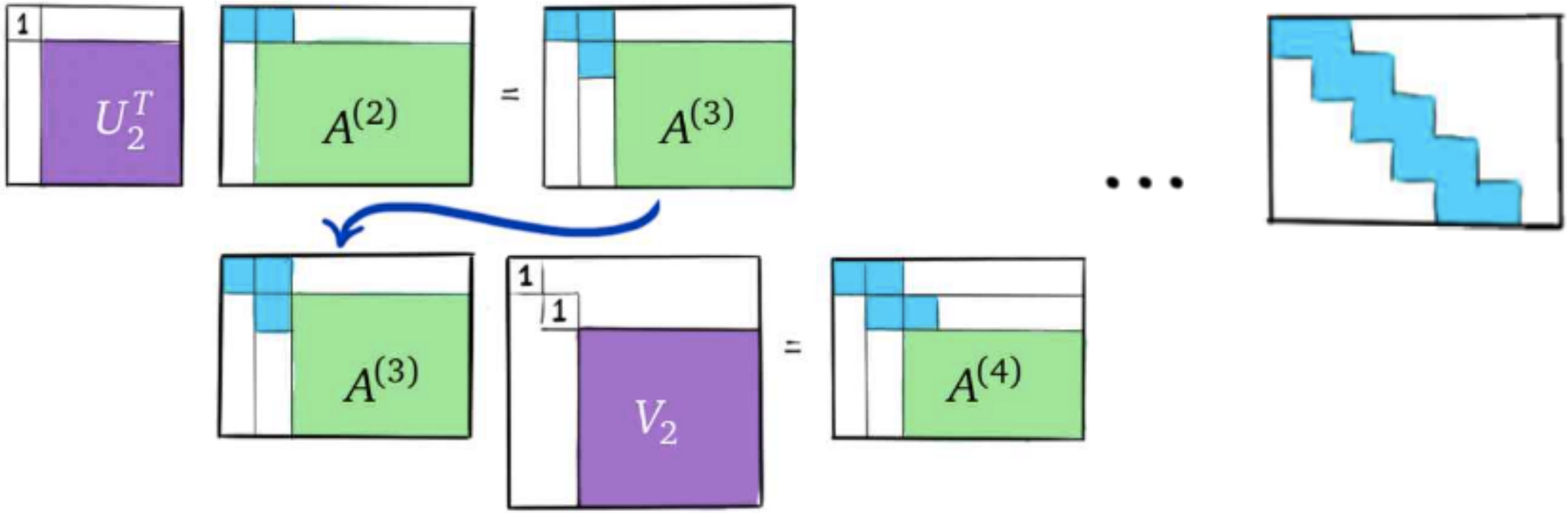
Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Bidiagonalização



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Bidiagonalização



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (explícito, ingênuo)

1. Determinar fatoração QR: $Q_V R \leftarrow B^T B - \alpha I$, onde α é um shift para o valor singular de menor valor baseado no algoritmo QR do Francis
2. Obter matriz atualizada $V \leftarrow R Q_V + \alpha I$

Passo Golub-Kahan (explícito, ingênuo)

1. Determinar fatoração QR: $Q_V R \leftarrow B^T B - \alpha I$, onde α é um shift para o valor singular de menor valor baseado no algoritmo QR do Francis
2. Obter matriz atualizada $V \leftarrow R Q_V + \alpha I$

- Custo computacional de construção de $B^T B$
- Matriz $B^T B$ pode não ser bem representada numericamente

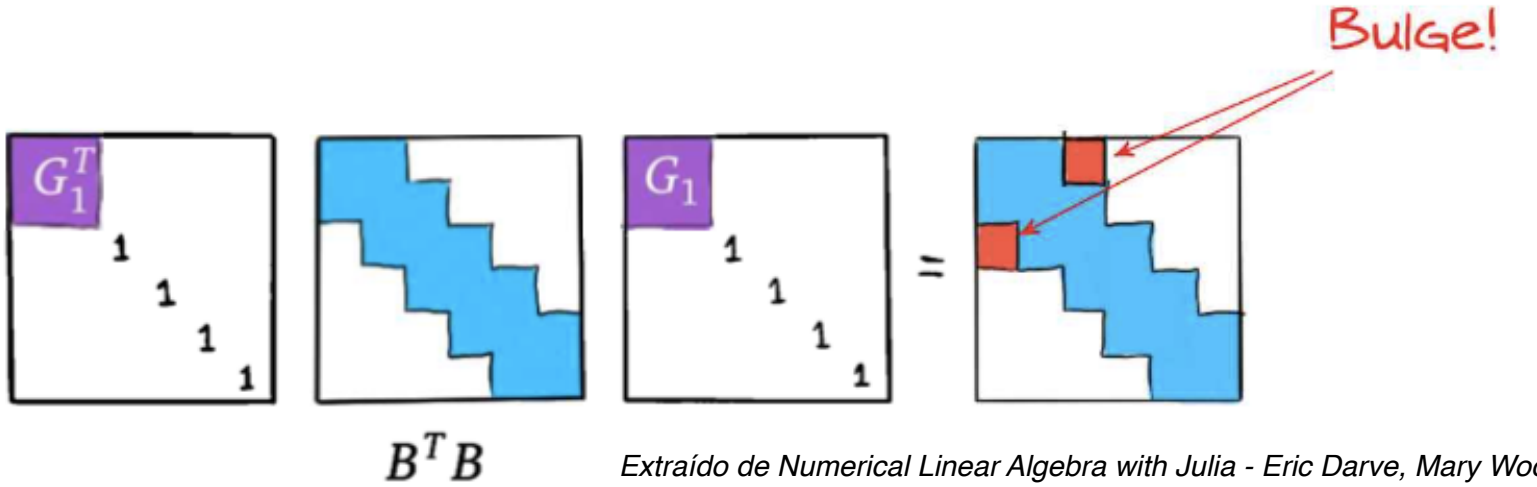
Passo Golub-Kahan (implícito)

- Vamos supor que iremos aplicar um passo do algoritmo QR (com shift, tal qual é feito no algoritmo de Francis) sobre $B^T B$
- Para isso, queremos uma *Givens Rotation* que introduza um zero sobre a primeira coluna de $B^T B$, ou seja

$$G_1^T \begin{bmatrix} b_{11}^2 - \alpha \\ b_{11} b_{12} \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Passo Golub-Kahan (implícito)

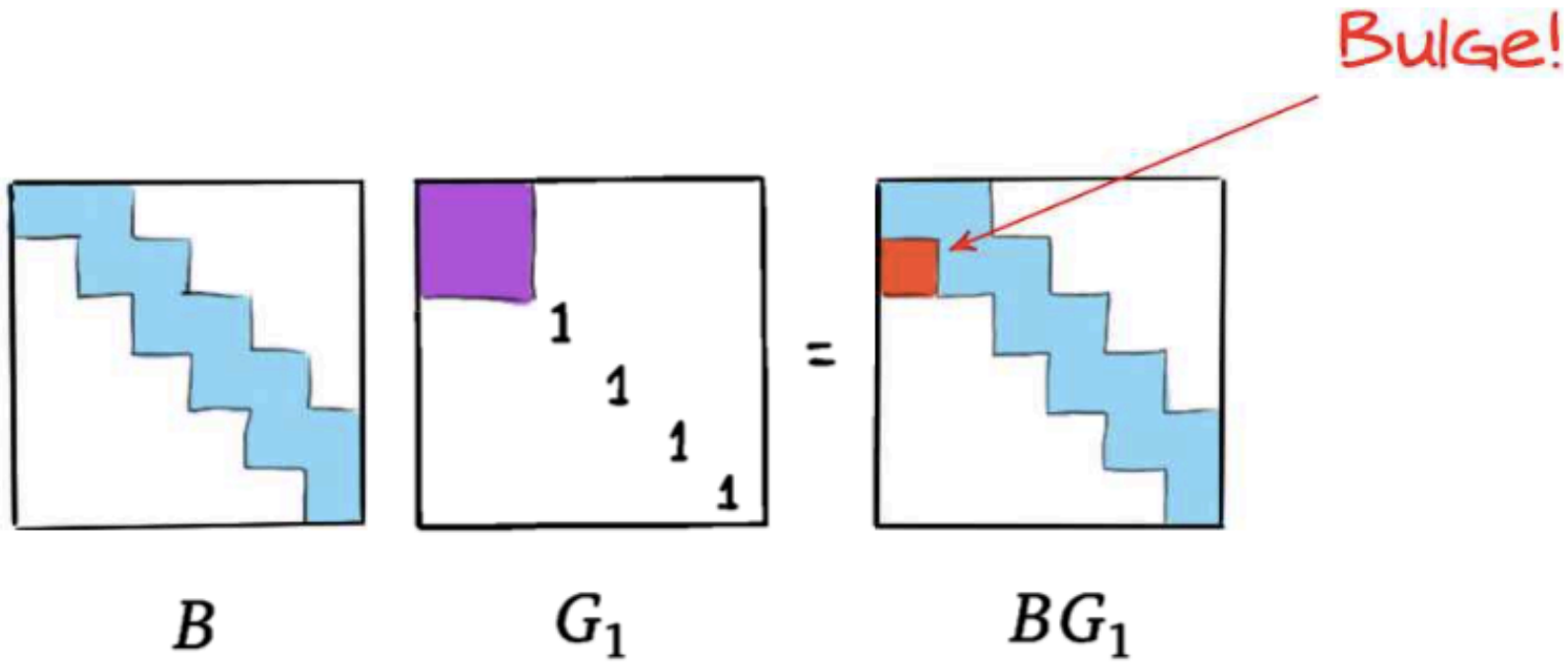
- Como a matriz $B^T B$ não possui esse shift efetivamente, isso introduz um *budge* (elemento não nulo na matriz)
- Assim como no algoritmo QR de Francis, é possível realizar um *budge-chasing* para retornar a uma matriz com as mesmas propriedades que a original



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (implícito)

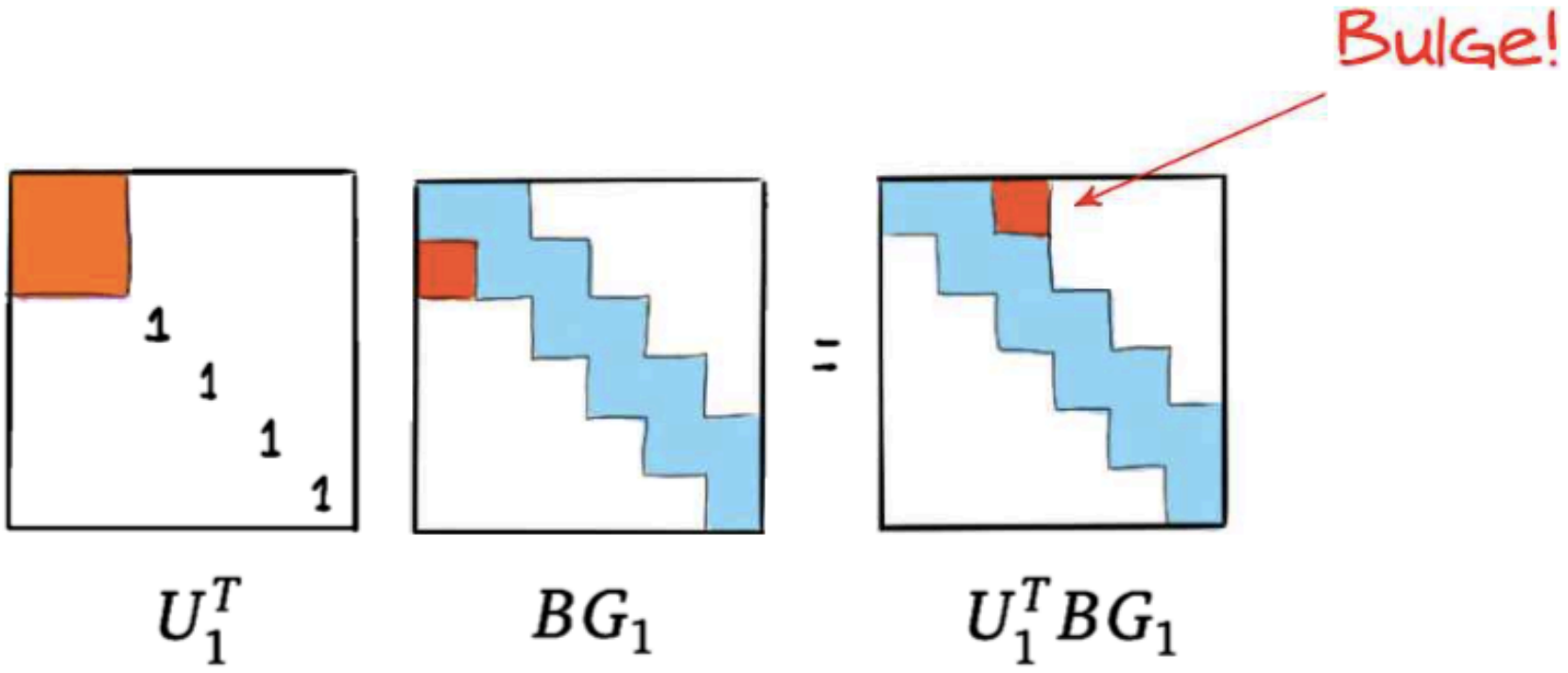
- Vamos considerar apenas o que é multiplicado por B, pois não queremos trabalhar com $B^T B$



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (implícito)

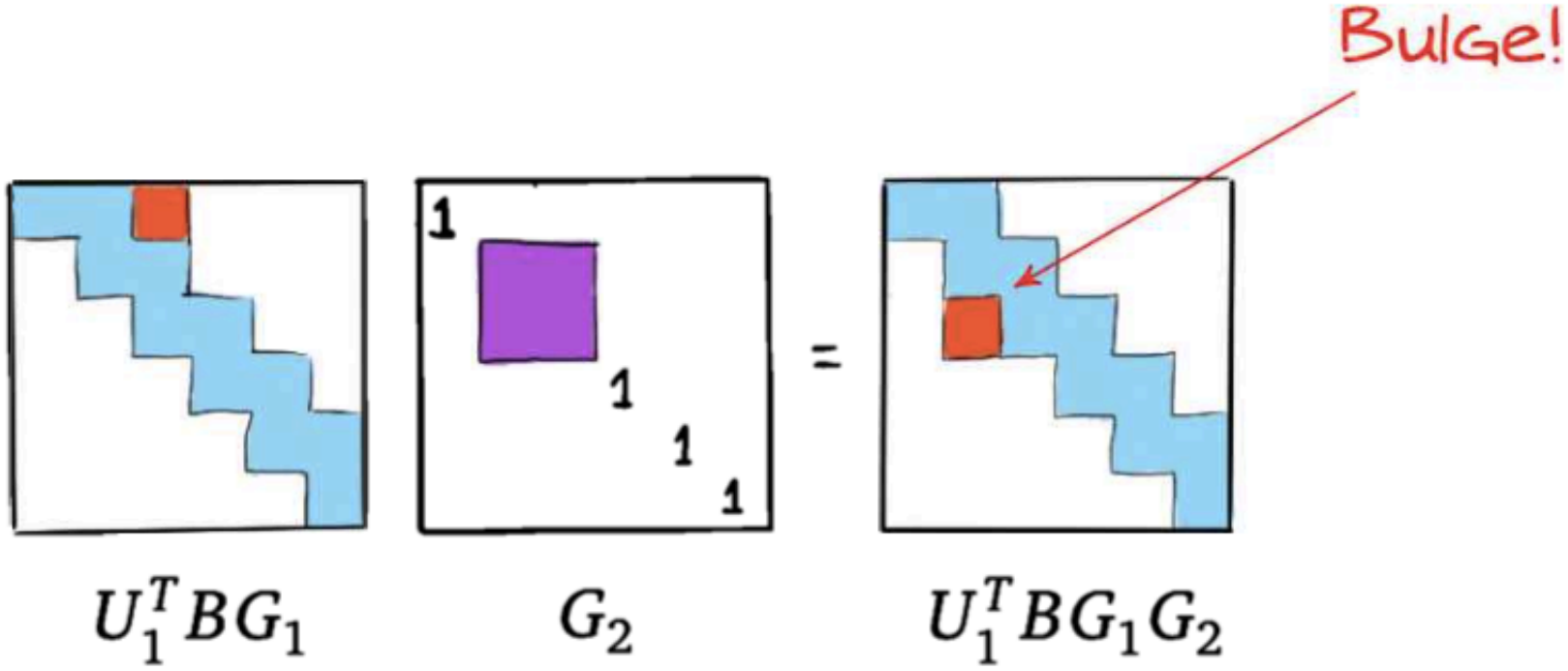
- Corrigindo o *budge* anterior com uma rotação pela esquerda



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (implícito)

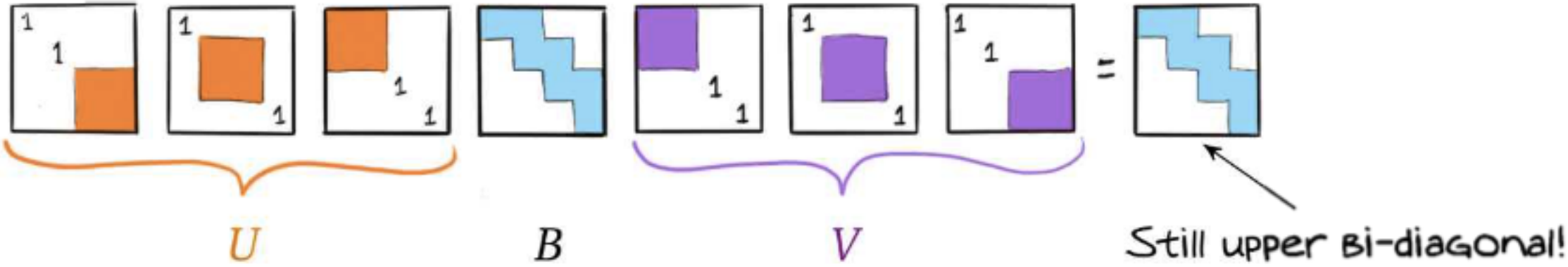
- Corrigindo novo *budge,* com uma rotação pela direita



Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (implícito)

- Repete-se esse processo iterativamente

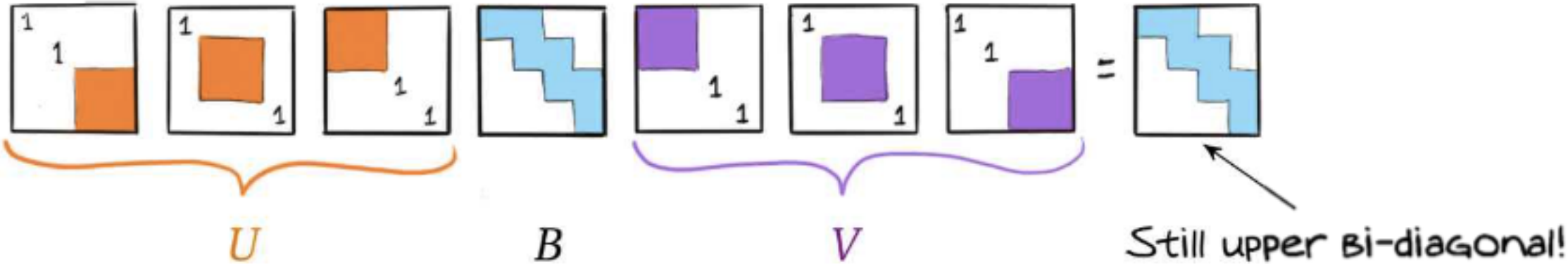


Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters

Passo Golub-Kahan (implícito)

- Repete-se esse processo iterativamente

Extraído de Numerical Linear Algebra with Julia - Eric Darve, Mary Wooters



- Nova matriz bidiagonal tem elementos acima da diagonal principal menores (irão convergir a zero), principalmente o associado ao valor singular mais próximo do shift α

Passo Golub-Kahan (implícito)

- Custo por iteração do passo Golub-Kahan é apenas $O(n)$, uma vez que B é originalmente bidiagonal
- É possível escanear a subdiagonal da matriz para verificar quais valores singulares já convergiram e alterar shift, garantindo **convergência cúbica** (iteração de Rayleigh)

Algoritmo Golub-Reinsch

- É possível escanear a matriz após cada iteração do algoritmo anterior e dividi-la em blocos, de modo que o algoritmo Golub-Kahan seja aplicada apenas em uma submatriz da matriz B

$$B = \begin{bmatrix} \begin{vmatrix} d_0 & e_1 & & \\ & d_1 & \ddots & \\ & & \ddots & e_{\ell-1} \\ & & & d_{\ell-1} \end{vmatrix} & & 0 \\ & \begin{vmatrix} d_\ell & e_{\ell+1} & & \\ & d_{\ell+1} & \ddots & \\ & & \ddots & e_k \\ & & & d_k \end{vmatrix} & \\ 0 & & \begin{vmatrix} d_{k+1} & \ddots & \\ & \ddots & 0 \\ & & d_{n-1} \end{vmatrix} \end{bmatrix}$$

Matriz B_{11}, inativa

Matriz B_{22}, aplicar Golub-Kahan até ficar diagonal

Matriz B_{33}, já diagonal

Algoritmo Golub-Reinsch

- Enfim, olhemos o código…

Referências

- Numerical Linear Algebra with Julia. *Eric Darve, Mary Wooters* (SIAM, livro, 2021)
- Handbook of linear algebra. *Alan Kaylor Cline, Inderjit S. Dhillon* (Chapman and Hall/CRC, livro, 2006)
- Singular value decomposition in extended double precision arithmetic. Christian *Reinsch, Mathias Richter* (Numerical Algorithms 93.3, 2023)

Agradecimentos

Perguntas?