Term Projects

- Solving linear equations with Gaussian Elimination
- The QR Algorithm for Symmetric Eigenvalue Problem
- The QR Algorithm for The SVD
- Quasi-Newton Methods

Solving linear equations with Gaussian Elimination (I)

- Gaussian Elimination with Partial Pivoting (GEPP)
- Gaussian Elimination with Complete Pivoting (GECP)
- Iterative Refinement
- Comparison with `matlab` \ function.

Gaussian Elimination with partial pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

Gaussian Elimination with partial pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

Let E_j be the j-th row of A

- for $s = 1, 2, \cdots, n-1$:
  - **pivoting**: choose largest entry in absolute value:

$$\mathbf{piv}_s \stackrel{def}{=} \mathbf{argmax}_{s \le j \le n} |a_{js}|, \quad E_s \leftrightarrow E_{\mathbf{piv}_s}$$

Gaussian Elimination with partial pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

Let E_j be the j-th row of A

- for $s = 1, 2, \cdots, n-1$:
  - **pivoting**: choose largest entry in absolute value:

$$\mathbf{piv}_s \stackrel{def}{=} \mathbf{argmax}_{s \le j \le n} |a_{js}|, \quad E_s \leftrightarrow E_{\mathbf{piv}_s} \quad \text{(row interchange: permutation)}.$$

Gaussian Elimination with partial pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

Let E_j be the j-th row of A

- for $s = 1, 2, \cdots, n-1$:
  - **pivoting**: choose largest entry in absolute value:

$$\mathbf{piv}_s \stackrel{def}{=} \mathbf{argmax}_{s \le j \le n} |a_{js}|, \quad E_s \leftrightarrow E_{\mathbf{piv}_s} \quad \text{(row interchange: permutation)}.$$

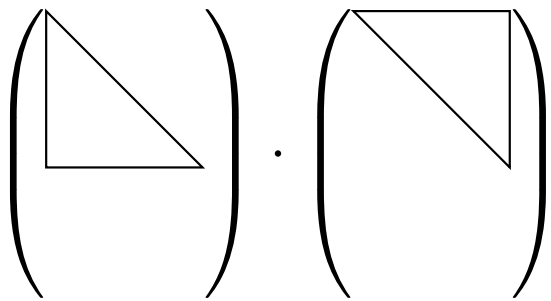
  - **eliminating** x_s from E_{s+1} through E_n:

$$\begin{aligned} l_{js} &= \frac{a_{js}}{a_{ss}}, \quad s+1 \le j \le n, \\ a_{jk} &= a_{jk} - l_{js}\, a_{sk}, \quad s+1 \le j,\, k \le n. \end{aligned}$$

GEPP as LU factorization

Theorem: Let $A = (a_{ij}) \in \mathbf{R}^{n \times n}$ be non-singular. Then GEPP computes an LU factorization with permutation matrix P such that

$$P \cdot A = L \cdot U = \left(\right) \cdot \left(\right) .$$



P is the accumulation of row interchanges, L is unit lower triangular, and U upper triangular.

Solving general linear equations with GEPP

$$A\,\mathbf{x} = \mathbf{b}, \quad P \cdot A = L \cdot U$$

- interchanging components in $\mathbf{b}$

$$P \cdot (A\,\mathbf{x}) = (P \cdot \mathbf{b}), \quad (L \cdot U)\,\mathbf{x} = (P \cdot \mathbf{b}).$$

- solving for $\mathbf{b}$ with forward and backward substitution

$$\begin{aligned} \mathbf{x} &= (L \cdot U)^{-1} (P \cdot \mathbf{b}) \\ &= \left(U^{-1} \left(L^{-1} (P \cdot \mathbf{b})\right)\right). \end{aligned}$$

Gaussian Elimination with complete pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

Gaussian Elimination with complete pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

- for $s = 1, 2, \cdots, n-1$:
  - **pivoting**: choose largest entry in absolute value:

$$(\mathbf{ip}, \mathbf{jp}) \stackrel{def}{=} \mathbf{argmax}_{s \le i,j \le n} |a_{ij}|, \quad E_s \leftrightarrow E_{\mathbf{ip}}.$$

swap matrix columns/ variables s and **jp**

Gaussian Elimination with complete pivoting

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

- for $s = 1, 2, \cdots, n-1$:
  - **pivoting**: choose largest entry in absolute value:

$$(\mathbf{ip}, \mathbf{jp}) \stackrel{def}{=} \mathbf{argmax}_{s \le i,j \le n} |a_{ij}|, \quad E_s \leftrightarrow E_{\mathbf{ip}}.$$

    swap matrix columns/ variables s and **jp**
  - **eliminating** x_s from E_{s+1} through E_n:

$$\mathbf{a}_{jk} = a_{jk} - \frac{a_{js}}{a_{ss}} a_{sk} \stackrel{overwrite}{\longrightarrow} a_{jk}, \quad s+1 \le j \le n, \quad s+1 \le k \le n,$$

More numerically stable, **too expensive**. GECP computes LU factorization with permutations $P_{\mathbf{row}}$ and $P_{\mathbf{column}}$,

$$P_{\mathbf{row}} \cdot A \cdot P_{\mathbf{column}} = L \cdot U = \begin{pmatrix} \blacksquare & & \\ & \ddots & \\ & & \end{pmatrix} \cdot \begin{pmatrix} \blacksquare & & \\ & \ddots & \\ & & \end{pmatrix}.$$

Error Bounds and Iterative Refinement

Assume that $\widehat{\mathbf{x}}$ is an approximation to the solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$.

- RESIDUAL $\widehat{\mathbf{r}} \stackrel{def}{=} \mathbf{b} - A\widehat{\mathbf{x}} = A\left(\mathbf{x} - \widehat{\mathbf{x}}\right)$. Thus small $\|\mathbf{x} - \widehat{\mathbf{x}}\|$ implies small $\|\widehat{\mathbf{r}}\|$.

Thm: Assume $\widehat{\mathbf{x}}$ is an approximate solution with $\widehat{\mathbf{r}} = b - A\widehat{\mathbf{x}}$. Then for any natural norm,

$$\begin{aligned} \|\widehat{\mathbf{x}} - \mathbf{x}\| &\leq \left\|A^{-1}\right\| \|\widehat{\mathbf{r}}\|, \\ \frac{\|\widehat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} &\leq \kappa(A) \frac{\|\widehat{\mathbf{r}}\|}{\|\mathbf{b}\|}, \end{aligned}$$

where $\kappa(A) \stackrel{def}{=} \|A\| \left\|A^{-1}\right\|$ is the CONDITION NUMBER of A □

Iterative Refinement (I)

- Let $A\,\mathbf{x} = \mathbf{b}$ with non-singular A and non-zero $\mathbf{b}$.
- Let $\mathcal{F}(\cdot)$ be an in-exact equation solver, so $\mathcal{F}(\mathbf{b})$ is approximate solution.
- Assume $\mathcal{F}(\cdot)$ is accurate enough that there exists a $\rho < 1$ so

$$\frac{\|\mathbf{b} - A\,\mathcal{F}(\mathbf{b})\|}{\|\mathbf{b}\|} \leq \rho \quad \text{for any } \mathbf{b} \neq \mathbf{0}.$$

Iterative Refinement (I)

- Let $A\,\mathbf{x} = \mathbf{b}$ with non-singular A and non-zero $\mathbf{b}$.
- Let $\mathcal{F}(\cdot)$ be an in-exact equation solver, so $\mathcal{F}(\mathbf{b})$ is approximate solution.
- Assume $\mathcal{F}(\cdot)$ is accurate enough that there exists a $\rho < 1$ so

$$\frac{\|\mathbf{b} - A\,\mathcal{F}(\mathbf{b})\|}{\|\mathbf{b}\|} \leq \rho \quad \text{for any } \mathbf{b} \neq \mathbf{0}.$$

For this project,

- $\mathcal{F}(\cdot)$ will be: LU factorization without pivoting, LU factorization with partial pivoting, and LU factorization with complete pivoting.

$$\mathcal{F}(\mathbf{b}) = U^{-1}\left(L^{-1}\,\mathbf{b}\right).$$

Iterative Refinement (II)

Given a tolerance $\tau > 0$ and $\mathbf{x}^{(0)}$

- Initialize $\mathbf{r}^{(0)} = \mathbf{b} - A\,\mathbf{x}^{(0)}$.
- for $k = 0, 1, \cdots$
  - Compute

$$\begin{aligned} \Delta\mathbf{x}^{(k)} &= \mathcal{F}\left(\mathbf{r}^{(k)}\right), \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}, \\ \mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - A\,\Delta\mathbf{x}^{(k)}. \end{aligned}$$

  - If $\left\|\mathbf{r}^{(k+1)}\right\| \leq \tau \, \|\mathbf{b}\|$ **stop**.

Project Requirements

- Implement GE, GEPP, GECP, with and without iterative refinement.
- Compare with `matlab` \ function in terms of accuracy (residual norm) and execution time.

The QR Algorithm for Symmetric Eigenvalue Problem

- Householder reduction to tridiagonal
- Implicit BULGE CHASING scheme for tridiagonal QR Algorithm.
- Recover eigenvectors.

Householder Reduction for symmetric $A \in \mathcal{R}^{n\times n}$

- Let $\mathbf{v}_j \in \mathbb{R}^{n-j}$ be the unit vector in the Householder Reflection matrix $\widehat{H}_j = I - 2\,\mathbf{v}_j\,\mathbf{v}_j^T$ so that

$$H_j \stackrel{def}{=} \begin{pmatrix} I_j & \\ & \widehat{H}_j \end{pmatrix} \in \mathcal{R}^{n\times n}, \quad j = 1, \cdots, n-1.$$

- Perform $n-1$ Householder reflections so that

$$H_{n-1} \cdots H_1\, A\, H_1^T \cdots H_{n-1}^T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix} \stackrel{def}{=} T.$$

- Householder's Method,

$$H^T A H = T, \quad \text{or} \quad A = H\,T\,H^T,$$

where $H \stackrel{def}{=} H_1^T \cdots H_{n-1}^T$.

Computing H

For $j = n-2, n-3, \cdots, 2, 1$, let

$$H_{j+1} \cdots H_{n-1} \stackrel{def}{=} \begin{pmatrix} I_j & & \\ & 1 & \\ & & \overline{H}_{j+1} \end{pmatrix}. \quad \text{Then}$$

$$H_j \cdot H_{j+1} \cdots H_{n-1} = \begin{pmatrix} I_j & \\ & \widehat{H}_j \end{pmatrix} \begin{pmatrix} I_j & & \\ & 1 & \\ & & \overline{H}_{j+1} \end{pmatrix} = \begin{pmatrix} I_j & \\ & \overline{H}_j \end{pmatrix},$$

where

$$\begin{aligned} \overline{H}_j &= \left(I - 2\,\mathbf{v}_j\,\mathbf{v}_j^T\right) \begin{pmatrix} 1 & \\ & \overline{H}_{j+1} \end{pmatrix} \\ &= \begin{pmatrix} 1 & \\ & \overline{H}_{j+1} \end{pmatrix} - (2\,\mathbf{v}_j) \left(\mathbf{v}_j^T \begin{pmatrix} 1 & \\ & \overline{H}_{j+1} \end{pmatrix}\right). \end{aligned}$$

4.6.2 The tridiagonal QR algorithm

In the symmetric case the Hessenberg QR algorithm becomes a tridiagonal QR algorithm. This can be executed in an **explicit** or an **implicit** way. In the explicit form, a QR step is essentially

---

1: Choose a shift μ
2: Compute the QR factorization $A - \mu I = QR$
3: Update A by $A = RQ + \mu I$.

Of course, this is done by means of plane rotations and by respecting the symmetric tridiagonal structure of A.

In the more elegant implicit form of the algorithm we first compute the first Givens rotation $G_0 = G(1, 2, \vartheta)$ of the QR factorization that zeros the $(2, 1)$ element of $A - \mu I$,

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a_{11} - \mu \\ a_{21} \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}, \qquad c = \cos(\vartheta_0), \quad s = \sin(\vartheta_0). \tag{4.12}$$

Performing a similary transformation with G_0 we have $(n = 5)$

$$G_0^* A G_0 = A' = \begin{bmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

Similar as with the double step Hessenberg QR algorithm we chase the bulge down the diagonal. In the 5×5 example this becomes

$$A \xrightarrow[= G(1,2,\vartheta_0)]{G_0} \begin{bmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \xrightarrow[= G(2,3,\vartheta_1)]{G_1} \begin{bmatrix} \times & \times & 0 & & \\ \times & \times & \times & + & \\ 0 & \times & \times & \times & \\ & + & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

$$\xrightarrow[= G(3,4,\vartheta_2)]{G_2} \begin{bmatrix} \times & \times & 0 & & \\ \times & \times & \times & & \\ & \times & \times & \times & + \\ & 0 & \times & \times & \times \\ & & + & \times & \times \end{bmatrix} \xrightarrow[= G(4,5,\vartheta_3)]{G_3} \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & 0 \\ & & \times & \times & \times \\ & & 0 & \times & \times \end{bmatrix} = \overline{A}.$$

The full step is given by

$$\overline{A} = Q^* A Q, \qquad Q = G_0 G_1 \cdots G_{n-2}.$$

Project Requirements

- Householder reduction to tridiagonal form
- Implicit Bulge Chasing tridiagonal QR Algorithm for all eigenvalues.
- Accumulate all Householder reflections and Givens rotations to recover all eigenvectors.
- Total cost $O(n^3)$ flops.
- Compare with `matlab` *eig* function in terms of accuracy (residual and orthogonality) and execution time.

Algorithm Details

`http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf`

The QR Algorithm for SVD

- Householder reduction to bidiagonal form
- Implicit BULGE CHASING scheme for bidiagonal SVD.
- Recover singular vectors.

Householder Bidiagonal Reduction for $A \in \mathbb{R}^{n \times n}$ (I)

Partition matrix $A = \begin{pmatrix} \mathbf{a}_1 & \widehat{A}_1 \end{pmatrix} \in \mathbb{R}^{n \times n}, \quad \widehat{A}_1 \in \mathbb{R}^{n \times (n-1)}$.

- Let $\mathbf{u}_1 \in \mathbb{R}^n$ be the unit vector in the Householder Reflection matrix $G_1 \stackrel{def}{=} \widehat{G}_1 = I - 2\,\mathbf{u}_1\,\mathbf{u}_1^T$ so that

$$\widehat{G}_1\,\mathbf{a}_1 = \begin{pmatrix} \pm\,\|\mathbf{a}_1\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_1 \\ \mathbf{0} \end{pmatrix}. \quad \text{and}$$

$$G_1\,A = \begin{pmatrix} \widehat{G}_1\,\mathbf{a}_1 & \widehat{G}_1\,\widehat{A}_1 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_1 & \mathbf{b}_1^T \\ \mathbf{0} & \overline{A}_1 \end{pmatrix}.$$

- Let $\mathbf{v}_1 \in \mathbb{R}^{n-1}$ be the unit vector in the Householder Reflection matrix $\widehat{H}_1 = I - 2\,\mathbf{v}_1\,\mathbf{v}_1^T$ so that,

$$\widehat{H}_1\,\mathbf{b}_1 = \begin{pmatrix} \pm\,\|\mathbf{b}_1\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \beta_1 \\ \mathbf{0} \end{pmatrix}. \quad \text{With} \quad H_1 = \begin{pmatrix} 1 & \\ & \widehat{H}_1 \end{pmatrix},$$

$$G_1\,A\,H_1 = \begin{pmatrix} \alpha_1 & \mathbf{b}_1^T\,\widehat{H}_1 \\ \mathbf{0} & \overline{A}_1\,\widehat{H}_1 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_1 & \beta_1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{a}_2 & \widehat{A}_2 \end{pmatrix}.$$

$$G_1\, A\, H_1 = \begin{pmatrix} \alpha_1 & \beta_1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{a}_2 & \widehat{A}_2 \end{pmatrix}, \quad \widehat{A}_2 \in \mathbb{R}^{(n-1)\times(n-2)}.$$

- Let $\mathbf{u}_2 \in \mathbb{R}^{n-1}$ be the unit vector in $\widehat{G}_2 = I - 2\,\mathbf{u}_2\,\mathbf{u}_2^T$ so that

$$\widehat{G}_2\,\mathbf{a}_2 = \begin{pmatrix} \pm\,\|\mathbf{a}_2\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_2 \\ \mathbf{0} \end{pmatrix}. \quad \text{With} \quad G_2 = \begin{pmatrix} 1 & \\ & \widehat{G}_2 \end{pmatrix},$$

$$G_2\, G_1\, A\, H_1 = \begin{pmatrix} \alpha_1 & \beta_1 & \mathbf{0}^T \\ \mathbf{0} & \widehat{G}_2\,\mathbf{a}_2 & \widehat{G}_2\,\widehat{A}_2 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_1 & \beta_1 & \mathbf{0}^T \\ 0 & \alpha_2 & \mathbf{b}_2^T \\ \mathbf{0} & \mathbf{0} & \overline{A}_2 \end{pmatrix}.$$

- Let $\mathbf{v}_2 \in \mathbb{R}^{n-2}$ be the unit vector in $\widehat{H}_2 = I - 2\,\mathbf{v}_2\,\mathbf{v}_2^T$ so that,

$$\widehat{H}_2\,\mathbf{b}_2 = \begin{pmatrix} \pm\,\|\mathbf{b}_2\|_2 \\ \mathbf{0} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \beta_2 \\ \mathbf{0} \end{pmatrix}. \quad \text{With} \quad H_2 = \begin{pmatrix} I_2 & \\ & \widehat{H}_2 \end{pmatrix}$$

$$G_2\, G_1\, A\, H_1\, H_2 = \begin{pmatrix} \alpha_1 & \beta_1 & \mathbf{0}^T \\ 0 & \alpha_2 & \mathbf{b}_2^T\,\widehat{H}_2 \\ \mathbf{0} & \mathbf{0} & \overline{A}_2\,\widehat{H}_2 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \mathbf{0}^T \\ 0 & \alpha_2 & \beta_2 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{a}_3 & \widehat{A}_3 \end{pmatrix}.$$

Householder Bidiagonal Reduction for $A \in \mathbb{R}^{n \times n}$ (III)

- After $n-1$ pairs of Householder Reflections,

$$G_{n-1} \cdots G_2 \, G_1 \, A \, H_1 \, H_2 \cdots H_{n-1} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix} \stackrel{def}{=} B.$$

  B is an upper bidiagonal matrix.
- Next step is QR algorithm for bidiagonal SVD.

Input matrix $B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix}$, tolerance τ

Algorithm 1 BidiSVD (B, τ)

while NOT YET CONVERGED **do**
  if $n = 1$ **then** return SVD of B
  for $j = 1, \cdots, n-1$ **do**
    if $|\beta_j| \leq \tau$ **then**
      return **BidiSVD** $(B(1:j, 1:j), \tau)$ and **BidiSVD** $(B(j+1:n, j+1:n+1), \tau)$
    end if
    if $|\alpha_j| \leq \tau$ **then**
      return **BidiSVD** $(B(1:j-1, 1:j), \tau)$ and **BidiSVD** $(B(j:n, j+1:n+1), \tau)$
    end if
  end for
  Compute shift σ. Bulge-chasing QR with σ.
end while

All forms Bidiagonal

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n & \beta_n \end{pmatrix}, \quad \widehat{B} = \begin{pmatrix} \beta_1 & & & & \\ \alpha_2 & \beta_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \end{pmatrix}$$

$$\widetilde{B} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix}, \quad \overline{B} = \begin{pmatrix} \beta_1 & & & & \\ \alpha_2 & \beta_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \beta_n \end{pmatrix}$$

All forms Bidiagonal

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n & \beta_n \end{pmatrix}, \quad \widehat{B} = \begin{pmatrix} \beta_1 & & & & \\ \alpha_2 & \beta_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \end{pmatrix}$$

$$\widetilde{B} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix}, \quad \overline{B} = \begin{pmatrix} \beta_1 & & & & \\ \alpha_2 & \beta_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \beta_n \end{pmatrix}$$

Next: QR for SVD of $B = \begin{pmatrix} \beta_1 & & & & \\ \alpha_2 & \beta_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \beta_n \end{pmatrix}$.

Bidiagonal SVD

- $B\,B^T$ is symmetric tridiagonal.

$$B\,B^T = \begin{pmatrix} \beta_1^2 & \alpha_2\beta_1 & & & \\ \alpha_2\beta_1 & \alpha_2^2+\beta_2^2 & \alpha_3\beta_2 & & \\ & \alpha_3\beta_2 & \alpha_3^2+\beta_3^2 & \ddots & \\ & & \ddots & \ddots & \alpha_n\beta_{n-1} \\ & & & \alpha_n\beta_{n-1} & \alpha_n^2+\beta_n^2 \end{pmatrix}.$$

- Bottom 2×2 matrix

$$\begin{pmatrix} \alpha_{n-1}^2+\beta_{n-1}^2 & \alpha_n\beta_{n-1} \\ \alpha_n\beta_{n-1} & \alpha_n^2+\beta_n^2 \end{pmatrix} = S\,S^T, \text{ with } S \stackrel{def}{=} \begin{pmatrix} \alpha_{n-1} & \beta_{n-1} \\ & \alpha_n & \beta_n \end{pmatrix}.$$

  Shift σ is the singular value of S closer to $\sqrt{\alpha_n^2+\beta_n^2}$.

- Bulge-chasing QR based on QR factorization of

$$B\,B^T - \sigma^2\,I = \begin{pmatrix} \beta_1^2-\sigma^2 & \alpha_2\beta_1 & & & \\ \alpha_2\beta_1 & \alpha_2^2+\beta_2^2-\sigma^2 & \ddots & & \\ & \ddots & \ddots & \alpha_n\beta_{n-1} & \\ & & \alpha_n\beta_{n-1} & \alpha_n^2+\beta_n^2-\sigma^2 & \end{pmatrix}.$$

Bulge-chasing QR for Bidiagonal SVD (I)

- Givens rotation on first column of $B\,B^T - \sigma^2\,I$:

$$G_1\,\left(B\,B^T - \sigma^2\,I\right) \stackrel{def}{=} \begin{pmatrix} c_1 & s_1 & & \\ -s_1 & c_1 & & \\ & & & I_{n-2} \end{pmatrix} \begin{pmatrix} \beta_1^2 - \sigma^2 \\ \alpha_2\beta_1 \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} * \\ 0 \\ \mathbf{0} \end{pmatrix}.$$

  G_1 is the first Givens rotation towards QR factorization of $B\,B^T - \sigma^2\,I$.

- Creating bulge: $\begin{pmatrix} d_1 & f_1 \\ d_2 & f_2 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} \beta_1 & 0 \\ \alpha_2 & \beta_2 \end{pmatrix}$.

$$G_1\,B = \begin{pmatrix} d_1 & f_1 & & & \\ d_2 & f_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & \alpha_n & \beta_n \end{pmatrix} = \begin{pmatrix} * & + & & & \\ * & * & & & \\ & * & \ddots & & \\ & & \ddots & * & \\ & & & * & * \end{pmatrix}.$$

- Next: Chase Bulge ($+$) down along the diagonals.

Bulge-chasing QR for Bidiagonal SVD (II)

$$G_1\, B = \begin{pmatrix} d_1 & f_1 & & & \\ & d_2 & f_2 & & \\ & & \alpha_3 & \ddots & \\ & & & \ddots & \beta_{n-1} \\ & & & & \alpha_n \quad \beta_n \end{pmatrix} = \begin{pmatrix} * & + & & & \\ & * & * & & \\ & & * & \ddots & \\ & & & \ddots & * \\ & & & & * \quad * \end{pmatrix}.$$

- Givens rotation on first row, $H_1 \stackrel{def}{=} \begin{pmatrix} \overline{c}_1 & \overline{s}_1 & \\ -\overline{s}_1 & \overline{c}_1 & \\ & & I_{n-2} \end{pmatrix}$

$$\begin{pmatrix} d_1 & f_1 & \mathbf{0}^T \end{pmatrix} H_1^T = \begin{pmatrix} \sqrt{d_1^2 + f_1^2} & 0 & \mathbf{0}^T \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \widehat{\beta}_1 & \mathbf{0}^T \end{pmatrix}.$$

- Compute

$$\begin{pmatrix} d_2 & f_2 \\ & \alpha_3 \end{pmatrix} \begin{pmatrix} \overline{c}_1 & \overline{s}_1 \\ -\overline{s}_1 & \overline{c}_1 \end{pmatrix}^T \stackrel{def}{=} \begin{pmatrix} \overline{d}_2 & \overline{f}_2 \\ \overline{d}_3 & \overline{f}_3 \end{pmatrix}.$$

Bulge-chasing QR for Bidiagonal SVD (III)

$$G_1 B H_1 = \begin{pmatrix} \widehat{\beta}_1 & & & & & \\ \overline{d}_2 & \overline{f}_2 & & & & \\ \overline{d}_3 & \overline{f}_3 & \beta_3 & & & \\ & & \alpha_4 & \ddots & & \\ & & & \ddots & \beta_{n-1} & \\ & & & & \alpha_n & \beta_n \end{pmatrix} = \begin{pmatrix} * & & & & & \\ * & * & & & & \\ + & * & * & & & \\ & & * & \ddots & & \\ & & & \ddots & * & \\ & & & & * & * \end{pmatrix}.$$

- Givens rotation on first column, $G_2 \stackrel{def}{=} \begin{pmatrix} 1 & & & \\ & c_2 & s_2 & \\ & -s_2 & c_2 & \\ & & & I_{n-3} \end{pmatrix}$

$$\begin{pmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{pmatrix} \begin{pmatrix} \overline{d}_2 & \overline{f}_2 & 0 \\ \overline{d}_3 & \overline{f}_3 & \beta_3 \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \widehat{\alpha}_2 & d_2 & f_2 \\ 0 & d_3 & f_3 \end{pmatrix}.$$

Bulge-chasing QR for Bidiagonal SVD (IV)

$$G_2\, G_1\, B\, H_1 = \begin{pmatrix} \widehat{\beta}_1 & & & & & & \\ \widehat{\alpha}_2 & d_2 & f_2 & & & & \\ & d_3 & f_3 & & & & \\ & & & \ddots & & & \\ & & & \ddots & \beta_{n-1} & & \\ & & & & \alpha_n & \beta_n \end{pmatrix} = \begin{pmatrix} * & & & & & & \\ * & * & + & & & & \\ & * & * & & & & \\ & & * & \ddots & & & \\ & & & \ddots & * & & \\ & & & & * & * \end{pmatrix}.$$

- New Givens rotation:

$$\begin{pmatrix} d_2 & f_2 \\ d_3 & f_3 \\ 0 & \alpha_4 \end{pmatrix} \begin{pmatrix} \overline{c}_2 & \overline{s}_2 \\ -\overline{s}_2 & \overline{c}_2 \end{pmatrix}^T = \begin{pmatrix} \widehat{\beta}_2 & 0 \\ \overline{d}_3 & \overline{f}_3 \\ \overline{d}_4 & \overline{f}_4 \end{pmatrix}.$$

- Repeat procedure until bulge disappears at the lower right corner.

Project Requirements

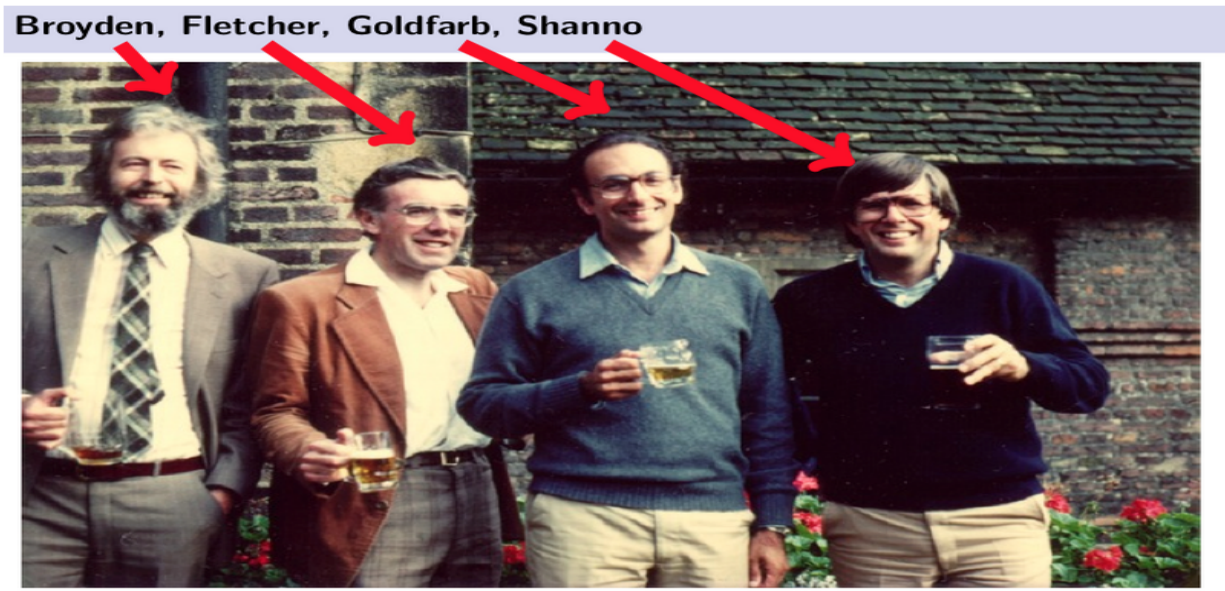
- Householder reduction to bidiagonal form
- Implicit Bulge Chasing bidiagonal QR Algorithm for all singular values.
- Accumulate all Householder reflections and Givens rotations to recover all singular vectors.
- Total cost $O(n^3)$ flops.
- Compare with `matlab` *svd* function in terms of accuracy (residual and orthogonality) and execution time.

Quasi-Newton Methods (BFGS) for $\mathbf{min_x}\ g(\mathbf{x})$

- BFGS Method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha\, \mathcal{B}_k^{-1} \nabla g\left(\mathbf{x}^{(k)}\right), \quad \text{where} \quad \mathcal{B}_k \in \mathcal{R}^{n\times n}.$$

- Desired Property I: "Mimics" $\mathcal{B}\left(\mathbf{x}^{(k)}\right)$ in some sense.
- Desired Property II: "Easy" to compute $\mathcal{B}_{k+1}^{-1}$ from $\mathcal{B}_k^{-1}$.
- Ensure symmetry in $\mathcal{B}_k$
- Step size selection as in Steepest Descent.



BFGS Method: Derivation

- Assume for some step $k \geq 0$, we have available $\mathbf{x}^{(k)} \in \mathcal{R}^n$ and $\mathcal{B}_k \in \mathcal{R}^{n \times n}$. By BFGS Method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathcal{B}_k^{-1} \nabla g\left(\mathbf{x}^{(k)}\right).$$

- Secant equation

$$\nabla g\left(\mathbf{x}^{(k+1)}\right) - \nabla g\left(\mathbf{x}^{(k)}\right) \approx \mathcal{H}\left(\mathbf{x}^{(k)}\right)\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right).$$

- BFGS ideas:
  - Approximate secant equation: $\mathcal{B}_{k+1}\,\mathbf{s}_{k+1} = \mathbf{y}_{k+1}$, where

$$\mathbf{y}_{k+1} \stackrel{def}{=} \nabla g\left(\mathbf{x}^{(k+1)}\right) - \nabla g\left(\mathbf{x}^{(k)}\right), \quad \mathbf{s}_{k+1} \stackrel{def}{=} \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

  - Choose a special, symmetric $\mathcal{B}_{k+1}$ that does not differ much from $\mathcal{B}_k$

$$\mathcal{B}_{k+1} = \mathcal{B}_k + \frac{\mathbf{y}_{k+1}\mathbf{y}_{k+1}^T}{\mathbf{y}_{k+1}^T\,\mathbf{s}_{k+1}} - \frac{\mathcal{B}_k\,\mathbf{s}_{k+1}\,\mathbf{s}_{k+1}^T\,\mathcal{B}_k}{\mathbf{s}_{k+1}^T\,\mathcal{B}_k\,\mathbf{s}_{k+1}}$$

BFGS Method

- Initialization: Given $\mathbf{x}^{(0)} \in \mathcal{R}^n$.
- Choose symmetric $\mathcal{B}_0 \in \mathcal{R}^{n \times n}$, typically SPD.
- For $k = 0, 1, \cdots,$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \, \mathcal{B}_k^{-1} \, \nabla g \left(\mathbf{x}^{(k)} \right),$$

$$\mathcal{B}_{k+1} = \mathcal{B}_k + \frac{\mathbf{y}_{k+1} \mathbf{y}_{k+1}^T}{\mathbf{y}_{k+1}^T \, \mathbf{s}_{k+1}} - \frac{\mathcal{B}_k \, \mathbf{s}_{k+1} \, \mathbf{s}_{k+1}^T \, \mathcal{B}_k}{\mathbf{s}_{k+1}^T \, \mathcal{B}_k \, \mathbf{s}_{k+1}}, \quad \text{with}$$

$$\mathbf{y}_{k+1} = \nabla g \left(\mathbf{x}^{(k+1)} \right) - \nabla g \left(\mathbf{x}^{(k)} \right), \quad \mathbf{s}_{k+1} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

BFGS Method

- Initialization: Given $\mathbf{x}^{(0)} \in \mathcal{R}^n$.
- Choose symmetric $\mathcal{B}_0 \in \mathcal{R}^{n \times n}$, typically SPD.
- For $k = 0, 1, \cdots,$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \, \mathcal{B}_k^{-1} \nabla g\left(\mathbf{x}^{(k)}\right),$$

$$\mathcal{B}_{k+1} = \mathcal{B}_k + \frac{\mathbf{y}_{k+1}\mathbf{y}_{k+1}^T}{\mathbf{y}_{k+1}^T \mathbf{s}_{k+1}} - \frac{\mathcal{B}_k \, \mathbf{s}_{k+1} \, \mathbf{s}_{k+1}^T \, \mathcal{B}_k}{\mathbf{s}_{k+1}^T \, \mathcal{B}_k \, \mathbf{s}_{k+1}}, \quad \text{with}$$

$$\mathbf{y}_{k+1} = \nabla g\left(\mathbf{x}^{(k+1)}\right) - \nabla g\left(\mathbf{x}^{(k)}\right), \quad \mathbf{s}_{k+1} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

Practical details:

- $\mathcal{B}_k$ remains SPD for CONVEX problems.
- Cholesky factorization of $\mathcal{B}_k$ can help to compute Cholesky factorization of $\mathcal{B}_{k+1}$? (updating Cholesky factorization)
- BFGS inverse update:

$$\mathcal{B}_{k+1}^{-1} = \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right) \mathcal{B}_k^{-1} \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)^T + \frac{\mathbf{s}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}.$$

Limited Memory BFGS (L-BFGS) Method (I)

$\mathcal{B}_{k+1}^{-1}$ maybe too large to fit in memory

- L-BFGS: Mimics BFGS, but with linear memory.

BFGS inverse update, from $\mathcal{B}_k^{-1}$ to $\mathcal{B}_{k+1}^{-1}$:

$$\mathcal{B}_{k+1}^{-1} = \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}} \right) \mathcal{B}_k^{-1} \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}} \right)^T + \frac{\mathbf{s}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}.$$

Big Idea: Go back only $m+1$ steps in inverse update

Limited Memory BFGS (L-BFGS) Method (II)

$$
\begin{aligned}
\mathcal{B}_{k+1}^{-1} \;=\; & \frac{\mathbf{s}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}} + \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-m}\,\mathbf{y}_{k+1-m}^T}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}\right) \\
& \times\mathcal{B}_{k-m}^{-1}\left(I - \frac{\mathbf{y}_{k+1-m}\,\mathbf{s}_{k+1-m}^T}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}\right)\cdots\left(I - \frac{\mathbf{y}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right) \\
& + \sum_{j=0}^{m-1}\left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-j}\,\mathbf{y}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right) \\
& \times\left(\frac{\mathbf{s}_{k-j}\,\mathbf{s}_{k-j}^T}{\mathbf{s}_{k-j}^T\,\mathbf{y}_{k-j}}\right)\left(I - \frac{\mathbf{y}_{k+1-j}\,\mathbf{s}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right)\cdots\left(I - \frac{\mathbf{y}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right).
\end{aligned}
$$

Big Idea: Go back only $m+1$ steps in inverse update: Set $\mathcal{B}_{k-m}^{-1} = \mathcal{B}_0^{-1}$

Computing $\mathcal{B}_{k+1}^{-1}\,\mathbf{h}$ for any vector $\mathbf{h}$ (I)

$$\mathcal{B}_{k+1}^{-1}\,\mathbf{h} = \alpha_{k+1}\,\mathbf{s}_{k+1} + \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-m}\,\mathbf{y}_{k+1-m}^T}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}\right)\mathcal{B}_0^{-1}\,\mathbf{q}_{k+1-m}$$
$$+\sum_{j=0}^{m-1}\left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-j}\,\mathbf{y}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right)$$
$$\times\left(\frac{\mathbf{s}_{k-j}\,\mathbf{s}_{k-j}^T}{\mathbf{s}_{k-j}^T\,\mathbf{y}_{k-j}}\right)\mathbf{q}_{k+1-j},$$

where $\alpha_{k+1} = \dfrac{\mathbf{s}_{k+1}^T\,\mathbf{h}}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}$, $\mathbf{q}_{k+1} = \mathbf{h} - \alpha_{k+1}\,\mathbf{y}_{k+1}$,

$$\mathbf{q}_t \overset{def}{=} \left(I - \frac{\mathbf{y}_t\,\mathbf{s}_t^T}{\mathbf{s}_t^T\,\mathbf{y}_t}\right)\cdots\left(I - \frac{\mathbf{y}_{k+1}\,\mathbf{s}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\mathbf{h}$$
$$= \mathbf{q}_{t+1} - \alpha_t\,\mathbf{y}_t, \quad \alpha_t = \frac{\mathbf{s}_t^T\,\mathbf{q}_{t+1}}{\mathbf{s}_t^T\,\mathbf{y}_t}, \quad t = k, \cdots, k+1-m.$$

Computing $\mathcal{B}_{k+1}^{-1}\,\mathbf{h}$ for any vector $\mathbf{h}$ (II)

Let $\mathbf{z}_{k+1-m} = \mathcal{B}_0^{-1}\,\mathbf{q}_{k+1-m}$. Then

$$\mathcal{B}_{k+1}^{-1}\,\mathbf{h} = \alpha_{k+1}\,\mathbf{s}_{k+1} + \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-m}\,\mathbf{y}_{k+1-m}^T}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}\right)\,\mathbf{z}_{k+1-m}$$
$$+ \sum_{j=0}^{m-1}\left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-j}\,\mathbf{y}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right)\times\left(\alpha_{k-j}\,\mathbf{s}_{k-j}\right)$$

Computing $\mathcal{B}_{k+1}^{-1}\,\mathbf{h}$ for any vector $\mathbf{h}$ (II)

Let $\mathbf{z}_{k+1-m} = \mathcal{B}_0^{-1}\,\mathbf{q}_{k+1-m}$. Then

$$\begin{aligned}
\mathcal{B}_{k+1}^{-1}\,\mathbf{h} &= \alpha_{k+1}\,\mathbf{s}_{k+1} + \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-m}\,\mathbf{y}_{k+1-m}^T}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}\right)\,\mathbf{z}_{k+1-m} \\
&\quad + \sum_{j=0}^{m-1}\left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-j}\,\mathbf{y}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right)\times\left(\alpha_{k-j}\,\mathbf{s}_{k-j}\right) \\
&= \alpha_{k+1}\,\mathbf{s}_{k+1} + \left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+2-m}\,\mathbf{y}_{k+2-m}^T}{\mathbf{s}_{k+2-m}^T\,\mathbf{y}_{k+2-m}}\right)\,\mathbf{z}_{k+2-m} \\
&\quad + \sum_{j=0}^{m-2}\left(I - \frac{\mathbf{s}_{k+1}\,\mathbf{y}_{k+1}^T}{\mathbf{s}_{k+1}^T\,\mathbf{y}_{k+1}}\right)\cdots\left(I - \frac{\mathbf{s}_{k+1-j}\,\mathbf{y}_{k+1-j}^T}{\mathbf{s}_{k+1-j}^T\,\mathbf{y}_{k+1-j}}\right)\times\left(\alpha_{k-j}\,\mathbf{s}_{k-j}\right),
\end{aligned}$$

where
$$\begin{aligned}
\mathbf{z}_{k+2-m} &= \mathbf{z}_{k+1-m} + \left(\alpha_{k+1-m} - \beta_{k+1-m}\right)\mathbf{s}_{k+1-m}, \\
\beta_{k+1-m} &= \frac{\mathbf{y}_{k+1-m}^T\,\mathbf{z}_{k+1-m}}{\mathbf{s}_{k+1-m}^T\,\mathbf{y}_{k+1-m}}
\end{aligned}$$

Algorithm 2 Computing $\mathcal{B}_{k+1}^{-1}\,\mathbf{h}$ for any vector $\mathbf{h}$

Set $\mathbf{q} = \mathbf{h}$.
for $t = k+1, k, \cdots, k+1-m$ **do**
  Compute $\alpha_t = \frac{\mathbf{s}_t^T\,\mathbf{q}}{\mathbf{s}_t^T\,\mathbf{y}_t}$, $\mathbf{q} = \mathbf{q} - \alpha_t\,\mathbf{y}_t$.
end for
Compute $\mathbf{z} = \mathcal{B}_0^{-1}\,\mathbf{q}$
for $t = k+1-m, \cdots, k+1$ **do**
  Compute $\beta = \frac{\mathbf{y}_t^T\,\mathbf{z}}{\mathbf{s}_t^T\,\mathbf{y}_t}$, $\mathbf{z} = \mathbf{z} + (\alpha_t - \beta)\,\mathbf{s}_t$.
end for
Return $\mathbf{z}$

Step-size Selection

- Ensure reduction in $g(\mathbf{x})$: Algorithm 3 will halt with an $\alpha_3 > 0$ so that $g\left(\mathbf{x}^{(k)} + \alpha_3 \mathbf{d}^{(k)}\right) < g\left(\mathbf{x}^{(k)}\right)$.

Algorithm 3 $g(\mathbf{x})$ Reduction

Set $\alpha_3 = 1$.
while $g\left(\mathbf{x}^{(k)} + \alpha_3 \mathbf{d}^{(k)}\right) \geq g\left(\mathbf{x}^{(k)}\right)$ **do**
  Set $\alpha_3 = \frac{\alpha_3}{2}$.
end while

Step-size Selection

- Ensure reduction in $g(\mathbf{x})$: Algorithm 3 will halt with an $\alpha_3 > 0$ so that $g\left(\mathbf{x}^{(k)} + \alpha_3 \mathbf{d}^{(k)}\right) < g\left(\mathbf{x}^{(k)}\right)$.

Algorithm 3 $g(\mathbf{x})$ Reduction

Set $\alpha_3 = 1$.
while $g\left(\mathbf{x}^{(k)} + \alpha_3 \mathbf{d}^{(k)}\right) \geq g\left(\mathbf{x}^{(k)}\right)$ **do**
  Set $\alpha_3 = \frac{\alpha_3}{2}$.
end while

- Ensure SUFFICIENT reduction in $g(\mathbf{x})$ with $\alpha^{(k)}$:

Algorithm 4 $g(\mathbf{x})$ Sufficient Reduction

Set $\alpha_2 = \alpha_3/2$. Compute coefficients h_0, h_1, h_2 so $\mathbf{P}(\alpha) = h_0 + h_1 \alpha + h_2 \alpha (\alpha - \alpha_2)$ interpolates $g\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right)$ at $\alpha = 0, \alpha_2, \alpha_3$. Set $\alpha_0 = \frac{1}{2}(\alpha_2 - h_1/h_2)$

$$\alpha^{(k)} = \mathbf{argmin}_{\alpha \in [\alpha_0, \alpha_2, \alpha_3]} g\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right).$$

BFGS Method for $\mathbf{min}_{\mathbf{x}}\, g(\mathbf{x})$

- Initialization: Given $\mathbf{x}^{(0)} \in \mathcal{R}^n$.
- Choose symmetric $\mathcal{B}_0 \in \mathcal{R}^{n \times n}$, typically SPD.
- For $k = 0, 1, \cdots$,
  - Compute $\mathbf{x}^{(d)} = -\mathcal{B}_k^{-1} \nabla g\left(\mathbf{x}^{(k)}\right)$.
  - Compute step size α.
  -

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \alpha\, \mathcal{B}_k^{-1} \nabla g\left(\mathbf{x}^{(k)}\right), \\ \mathcal{B}_{k+1} &= \mathcal{B}_k + \frac{\mathbf{y}_{k+1}\mathbf{y}_{k+1}^T}{\mathbf{y}_{k+1}^T\, \mathbf{s}_{k+1}} - \frac{\mathcal{B}_k\, \mathbf{s}_{k+1}\, \mathbf{s}_{k+1}^T\, \mathcal{B}_k}{\mathbf{s}_{k+1}^T\, \mathcal{B}_k\, \mathbf{s}_{k+1}}, \quad \text{with} \end{aligned}$$

$$\mathbf{y}_{k+1} = \nabla g\left(\mathbf{x}^{(k+1)}\right) - \nabla g\left(\mathbf{x}^{(k)}\right), \quad \mathbf{s}_{k+1} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

Project Requirements

- Implement BFGS and L-BFGS
- Run BFGS and L-BFGS with logistic objective function