CS 598 EVS: Tensor Computations

Matrix Computations Background

Edgar Solomonik

University of Illinois, Urbana-Champaign

Matrices and Tensors

- What is a matrix?
  - *A collection of numbers arranged into an array of dimensions $m \times n$, e.g., $\boldsymbol{M} \in \mathbb{R}^{m \times n}$*
  - *A linear operator $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}$*
  - *A bilinear form $\boldsymbol{x}^T \boldsymbol{M} \boldsymbol{y}$*
- What is a tensor?
  - *A collection of numbers arranged into an array of a particular order, with dimensions $l \times m \times n \times \cdots$, e.g., $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{l \times m \times n}$ is order 3*
  - *A multilinear operator $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$*

$$z_i = \sum_{j,k} t_{ijk} x_j y_k$$

  - *A multilinear form $\sum_{i,j,k} t_{ijk} x_i y_j z_k$*

Matrix Norms

- **Properties of matrix norms**:

$$\begin{aligned} \|\boldsymbol{A}\| &\geqslant 0 \\ \|\boldsymbol{A}\| = 0 \quad &\Leftrightarrow \quad \boldsymbol{A} = \boldsymbol{0} \\ \|\alpha \boldsymbol{A}\| &= |\alpha| \cdot \|\boldsymbol{A}\| \\ \|\boldsymbol{A} + \boldsymbol{B}\| &\leqslant \|\boldsymbol{A}\| + \|\boldsymbol{B}\| \quad \textit{(triangle inequality)} \end{aligned}$$

- **Frobenius norm**:

$$\|\mathbf{A}\|_F = \left(\sum_{i,j} a_{ij}^2 \right)^{1/2}$$

- **Operator/induced/subordinate matrix norms**:

  For any vector norm $\|\cdot\|_p$, the induced matrix norm is

$$\|\boldsymbol{A}\|_p = \max_{\boldsymbol{x} \neq \boldsymbol{0}} \|\boldsymbol{A}\boldsymbol{x}\|_p / \|\boldsymbol{x}\|_p = \max_{\|\boldsymbol{x}\|_p = 1} \|\boldsymbol{A}\boldsymbol{x}\|_p$$

Existence of SVD

- **Consider any maximizer $\boldsymbol{x}_1 \in \mathbb{R}^n$ with $\|\boldsymbol{x}_1\|_2 = 1$ to $\|\boldsymbol{A}\boldsymbol{x}_1\|_2$**

  Let $\boldsymbol{y}_1 = \boldsymbol{A}\boldsymbol{x}_1 / \|\boldsymbol{A}\boldsymbol{x}_1\|_2$ and $\sigma_1 = \boldsymbol{y}_1^T \boldsymbol{A}\boldsymbol{x}_1 = \|\boldsymbol{A}\boldsymbol{x}_1\|_2$, then consider any maximizer $\boldsymbol{x}_2$ of

$$\left\|(\boldsymbol{A} - \sigma_1 \boldsymbol{y}_1 \boldsymbol{x}_1^T)\boldsymbol{x}_2\right\|_2 .$$

  We can see that $\boldsymbol{x}_1 \perp \boldsymbol{x}_2$ since, otherwise, we have $\boldsymbol{x}_2 = \alpha \boldsymbol{x}_1 + \tilde{\boldsymbol{x}}_2$ with $\tilde{\boldsymbol{x}}_2 \perp \boldsymbol{x}_1$ and $\|\tilde{\boldsymbol{x}}_2\|_2 < \|\boldsymbol{x}_2\|_2$ and

$$\left\|(\boldsymbol{A} - \sigma_1 \boldsymbol{y}_1 \boldsymbol{x}_1^T)(\alpha \boldsymbol{x}_1 + \tilde{\boldsymbol{x}}_2)\right\|_2 = \left\|(\boldsymbol{A} - \sigma_1 \boldsymbol{y}_1 \boldsymbol{x}_1^T)\tilde{\boldsymbol{x}}_2\right\|_2 .$$

  Hence we have a contradiction, since

$$\left\|(\boldsymbol{A} - \sigma_1 \boldsymbol{y}_1 \boldsymbol{x}_1^T)\boldsymbol{x}_2\right\|_2 < (1/\|\tilde{\boldsymbol{x}}_2\|_2) \left\|(\boldsymbol{A} - \sigma_1 \boldsymbol{y}_1 \boldsymbol{x}_1^T)\tilde{\boldsymbol{x}}_2\right\|_2 .$$

  More generally, we can see that any maximizer $\boldsymbol{x}_{i+1}$ to

$$\|(\boldsymbol{A} - \begin{bmatrix}\boldsymbol{y}_1 & \cdots & \boldsymbol{y}_i\end{bmatrix} \begin{bmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_i\end{bmatrix} \begin{bmatrix}\boldsymbol{x}_1 & \cdots & \boldsymbol{x}_i\end{bmatrix}^T)\boldsymbol{x}_{i+1}\|_2$$

  is orthogonal to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$ and similar for $\boldsymbol{y}_{i+1}$.

Singular Value Decomposition

- The singular value decomposition (SVD)

  We can express any matrix $\boldsymbol{A}$ as

  $$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

  where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal, and $\boldsymbol{\Sigma}$ is square nonnegative and diagonal,

  $$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{max} & & \\ & \ddots & \\ & & \sigma_{min} \end{bmatrix}$$

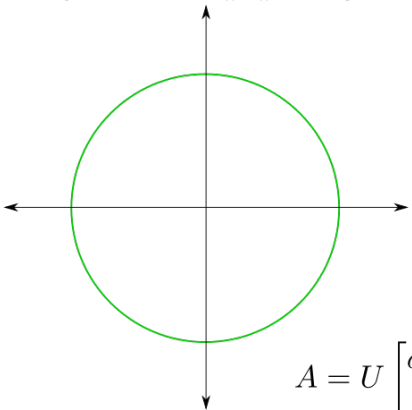
  The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are left and right singular vectors of $\boldsymbol{A}$, i.e.,

  $$\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i, \quad \boldsymbol{u}_i^T \boldsymbol{A} = \sigma_i \boldsymbol{v}_i^T$$

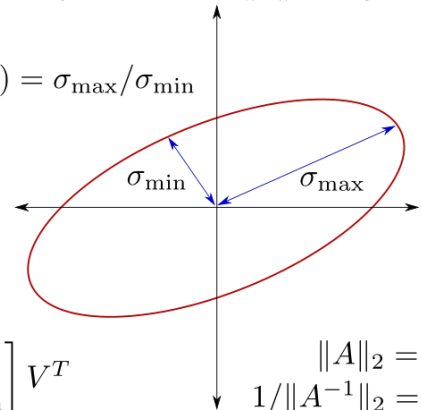
- Condition number in terms of singular values
  - *We have that $\|\boldsymbol{A}\|_2 = \sigma_{max}$ and if $\boldsymbol{A}^{-1}$ exists, $\|\boldsymbol{A}^{-1}\|_2 = 1/\sigma_{min}$*
  - *Consequently, $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \|\boldsymbol{A}^{-1}\|_2 = \sigma_{max}/\sigma_{min}$*

Visualization of Matrix Conditioning

$$\{x : x \in \mathbb{R}^2, \|x\|_2 = 1\} \xrightarrow{A} \{Ax : x \in \mathbb{R}^2, \|x\|_2 = 1\}$$



$$\kappa(A) = \sigma_{\max}/\sigma_{\min}$$



$$A = U \begin{bmatrix} \sigma_{\max} & \\ & \sigma_{\min} \end{bmatrix} V^T$$

$$\|A\|_2 = \sigma_{\max}$$
$$1/\|A^{-1}\|_2 = \sigma_{\min}$$

Matrix Condition Number

- The matrix condition number $\kappa(\boldsymbol{A})$ is the ratio between the max and min distance from the surface to the center of the unit ball (norm-1 vectors) transformed by $\boldsymbol{A}$:
  - *The max distance to center is given by the vector maximizing* $\max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|_2$.
  - *The min distance to center is given by the vector minimizing* $\min_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|_2 = 1/(\max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}^{-1}\boldsymbol{x}\|_2)$.
  - *Thus, we have that* $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|_2 \|\boldsymbol{A}^{-1}\|_2$
- The matrix condition number bounds the worst-case amplification of error in a matrix-vector product: *Consider* $\boldsymbol{y} + \boldsymbol{\delta y} = \boldsymbol{A}(\boldsymbol{x} + \boldsymbol{\delta x})$, *assume* $\|\boldsymbol{x}\|_2 = 1$
  - *In the worst case,* $\|\boldsymbol{y}\|_2$ *is minimized, that is* $\|\boldsymbol{y}\|_2 = 1/\|\boldsymbol{A}^{-1}\|_2$
  - *In the worst case,* $\|\boldsymbol{\delta y}\|_2$ *is maximized, that is* $\|\boldsymbol{\delta y}\|_2 = \|\boldsymbol{A}\|_2 \|\boldsymbol{\delta y}\|_2$
  - *So* $\|\boldsymbol{\delta y}\|_2 / \|\boldsymbol{y}\|_2$ *is at most* $\kappa(\boldsymbol{A}) \|\boldsymbol{\delta x}\|_2 / \|\boldsymbol{x}\|_2$

Linear Systems

- Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with rank n, consider solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ given $\boldsymbol{b}$
- The SVD allows explicit inversion of $\boldsymbol{A}$

$$\boldsymbol{A}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T$$

- However, Gaussian elimination is more computationally efficient
  - *Can factorize arbitrary $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{U}$ for permutation matrix $\boldsymbol{P}$ and triangular $\boldsymbol{L}, \boldsymbol{U}$*
  - *For symmetric $\boldsymbol{A}$ LDLT factorization is $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T\boldsymbol{P}^T$, where $\boldsymbol{D}$ has diagonal entries of 2-by-2 anti-diagonal symmetric blocks*
  - *If positive definite, Cholesky requires no pivoting/permutation*
  - *Suffices to solve linear systems in $O(n^2)$ cost using triangular solve*
- Given a factorization of $\boldsymbol{A}$, solving a system with $\boldsymbol{A} + \boldsymbol{u}\boldsymbol{v}^T$ has cost $O(n^2)$ via the Sherman-Morrison-Woodbury formula

Linear Least Squares

- Find $\boldsymbol{x}^\star = \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$:

  Since $m \geqslant n$, the minimizer generally does not attain a zero residual $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$. We can rewrite the optimization problem constraint via

$$\boldsymbol{x}^\star = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 = \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^n} \left[(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\right]$$

- Given the SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ we have $\boldsymbol{x}^\star = \underbrace{\boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T}_{\boldsymbol{A}^\dagger}\boldsymbol{b}$, where $\boldsymbol{\Sigma}^\dagger$ contains the reciprocal of all nonzeros in $\boldsymbol{\Sigma}$, and more generally $\dagger$ denotes pseudoinverse:
  - *The minimizer satisfies $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x}^\star \cong \boldsymbol{b}$ and consequently also satisfies*

$$\boldsymbol{\Sigma}\boldsymbol{y}^\star \cong \boldsymbol{d} \quad \textit{where } \boldsymbol{y}^\star = \boldsymbol{V}^T\boldsymbol{x}^\star \textit{ and } \boldsymbol{d} = \boldsymbol{U}^T\boldsymbol{b}.$$

  - *The minimizer of the reduced problem is $\boldsymbol{y}^\star = \boldsymbol{\Sigma}^\dagger\boldsymbol{d}$, so $y_i = d_i/\sigma_i$ for $i \in \{1,\ldots,n\}$ and $y_i = 0$ for $i \in \{n+1,\ldots,m\}$.*

Normal Equations

Demo: *Normal equations vs Pseudoinverse*
Demo: *Issues with the normal equations*

- *Normal equations* are given by solving $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$:

  If $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$ *then*

$$
\begin{aligned}
(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)^T\boldsymbol{b} \\
\boldsymbol{\Sigma}^T\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= \boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{b} \\
\boldsymbol{V}^T\boldsymbol{x} &= (\boldsymbol{\Sigma}^T\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}^T\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T\boldsymbol{b} \\
\boldsymbol{x} &= \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T\boldsymbol{b} = \boldsymbol{x}^\star
\end{aligned}
$$

- However, solving the normal equations is a more ill-conditioned problem then the original least squares algorithm

  Generally we have $\kappa(\boldsymbol{A}^T\boldsymbol{A}) = \kappa(\boldsymbol{A})^2$ *(the singular values of* $\boldsymbol{A}^T\boldsymbol{A}$ *are the squares of those in* $\boldsymbol{A}$*). Consequently, solving the least squares problem via the normal equations may be unstable because it involves solving a problem that has worse conditioning than the initial least squares problem.*

Solving the Normal Equations

- If $\boldsymbol{A}$ is full-rank, then $\boldsymbol{A}^T\boldsymbol{A}$ is symmetric positive definite (SPD):
  - *Symmetry is easy to check* $(\boldsymbol{A}^T\boldsymbol{A})^T = \boldsymbol{A}^T\boldsymbol{A}$.
  - $\boldsymbol{A}$ *being full-rank implies* $\sigma_{min} > 0$ *and further if* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ *we have*

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}^T\boldsymbol{\Sigma}^2\boldsymbol{V}$$

    which implies that rows of $\boldsymbol{V}$ *are the eigenvectors of* $\boldsymbol{A}^T\boldsymbol{A}$ *with eigenvalues* $\boldsymbol{\Sigma}^2$ *since* $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{\Sigma}^2$.

- Since $\boldsymbol{A}^T\boldsymbol{A}$ is SPD we can use Cholesky factorization, to factorize it and solve linear systems:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$$

QR Factorization

- If $\boldsymbol{A}$ is full-rank there exists an orthogonal matrix $\boldsymbol{Q}$ and a unique upper-triangular matrix $\boldsymbol{R}$ with a positive diagonal such that $\boldsymbol{A} = \boldsymbol{QR}$
  - *Given $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$, we can take $\boldsymbol{R} = \boldsymbol{L}^T$ and obtain $\boldsymbol{Q} = \boldsymbol{A}\boldsymbol{L}^{-T}$, since $\underbrace{\boldsymbol{L}^{-1}\boldsymbol{A}^T}_{\boldsymbol{Q}^T}\underbrace{\boldsymbol{A}\boldsymbol{L}^{-T}}_{\boldsymbol{Q}} = \boldsymbol{I}$ implies that $\boldsymbol{Q}$ has orthonormal columns.*
- A reduced QR factorization (unique part of general QR) is defined so that $\boldsymbol{Q} \in \mathbb{R}^{m\times n}$ has orthonormal columns and $\boldsymbol{R}$ is square and upper-triangular

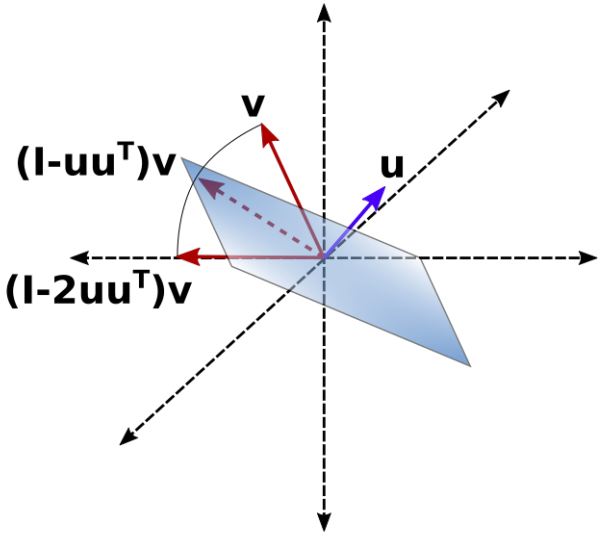
  A full QR factorization gives $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, but since $\boldsymbol{R}$ is upper triangular, the latter $m-n$ columns of $\boldsymbol{Q}$ are only constrained so as to keep $\boldsymbol{Q}$ orthogonal. The reduced QR factorization is given by taking the first n columns $\boldsymbol{Q}$ and $\hat{\boldsymbol{Q}}$ the upper-triangular block of $\boldsymbol{R}$, $\hat{\boldsymbol{R}}$ giving $\boldsymbol{A} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$.
- We can solve the normal equations (and consequently the linear least squares problem) via reduced QR as follows

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b} \quad \Rightarrow \quad \hat{\boldsymbol{R}}^T\underbrace{\hat{\boldsymbol{Q}}^T\hat{\boldsymbol{Q}}}_{\boldsymbol{I}}\hat{\boldsymbol{R}}\boldsymbol{x} = \hat{\boldsymbol{R}}^T\hat{\boldsymbol{Q}}^T\boldsymbol{b} \quad \Rightarrow \quad \hat{\boldsymbol{R}}\boldsymbol{x} = \hat{\boldsymbol{Q}}^T\boldsymbol{b}$$

Computing the QR Factorization

- The Cholesky-QR algorithm uses the normal equations to obtain the QR factorization
  - *Compute $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$, take $\boldsymbol{R} = \boldsymbol{L}^T$, and solve for $\boldsymbol{Q}$ triangular linear systems $\boldsymbol{L}\boldsymbol{Q}^T = \boldsymbol{A}^T$*
  - *If $\boldsymbol{A}$ is $m \times n$, forming $\boldsymbol{A}^T\boldsymbol{A}$ has cost mn^2, computing Cholesky factorization has cost $(2/3)n^3$, and solving the triangular systems (if $\boldsymbol{Q}$ is needed) costs mn^2, yielding total cost $2mn^2 + (2/3)n^3$*
  - *However, this method is unstable since $\boldsymbol{A}^T\boldsymbol{A}$ is ill-conditioned. This is addressible by iterating on the computed (nearly-orthogonal) $\boldsymbol{Q}$ factor (CholeskyQR2).*
- Orthogonalization-based methods are most efficient and stable for QR factorization of dense matrices
  - *Apply a sequence of orthogonal transformations $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_k$ to reduce $\boldsymbol{A}$ to triangular form $(\boldsymbol{Q}_1 \cdots \boldsymbol{Q}_k)^T\boldsymbol{A} = \boldsymbol{R}$*
  - *Householder QR uses rank-1 perturbations of the identity matrix (reflectors) $\boldsymbol{Q}_i = \boldsymbol{I} - 2\boldsymbol{u}_i\boldsymbol{u}_i^T$ to zero-out each sub-column of $\boldsymbol{A}$*
  - *Givens rotations zero-out a single entry at a time*
  - *Both approaches have cost $O(mn^2)$ with similar constant to Cholesky-QR*

Householder orthogonalization



Eigenvalue Decomposition

- If a matrix $\boldsymbol{A}$ is diagonalizable, it has an *eigenvalue decomposition*

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1}$$

where $\boldsymbol{X}$ are the right eigenvectors, $\boldsymbol{X}^{-1}$ are the left eigenvectors and $\boldsymbol{D}$ are eigenvalues

$$\boldsymbol{A}\boldsymbol{X} = \begin{bmatrix}\boldsymbol{A}\boldsymbol{x}_1 & \cdots \boldsymbol{A}\boldsymbol{x}_n\end{bmatrix} = \boldsymbol{X}\boldsymbol{D} = \begin{bmatrix}d_{11}\boldsymbol{x}_1 & \cdots & d_{nn}\boldsymbol{x}_n\end{bmatrix}.$$

  - *If $\boldsymbol{A}$ is symmetric, its right and left singular vectors are the same, and consequently are its eigenvectors.*
  - *More generally, any normal matrix, $\boldsymbol{A}^H\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^H$, has unitary eigenvectors.*
- $\boldsymbol{A}$ and $\boldsymbol{B}$ are *similar*, if there exist $\boldsymbol{Z}$ such that $\boldsymbol{A} = \boldsymbol{Z}\boldsymbol{B}\boldsymbol{Z}^{-1}$
  - *Normal matrices are unitarily similar ($\boldsymbol{Z}^{-1} = \boldsymbol{Z}^H$) to diagonal matrices*
  - *Symmetric real matrices are orthogonally similar ($\boldsymbol{Z}^{-1} = \boldsymbol{Z}^T$) to real diagonal matrices*
  - *Hermitian matrices are unitarily similar to real diagonal matrices*

Similarity of Matrices

Invertible similarity transformations $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A}\boldsymbol{X}^{-1}$

matrix ($\boldsymbol{A}$)	*reduced form* ($\boldsymbol{Y}$)
arbitrary	bidiagonal
diagonalizable	diagonal

Unitary similarity transformations $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^{H}$

matrix ($\boldsymbol{A}$)	*reduced form* ($\boldsymbol{Y}$)
arbitrary	triangular (Schur)
normal	diagonal
Hermitian	real diagonal

Orthogonal similarity transformations $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{A}\boldsymbol{Q}^{T}$

matrix ($\boldsymbol{A}$)	*reduced form* ($\boldsymbol{Y}$)
real	Hessenberg
real symmetric	real diagonal
real SPD	real positive diagonal

Field of Values

- For any square matrix $\boldsymbol{A}$ and vector $\boldsymbol{x}$ the *Rayleigh quotient* is

$$\rho_{\boldsymbol{A}}(\boldsymbol{x}) = \frac{\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^H \boldsymbol{x}}$$

- Its magnitude is bounded by the singular values as

$$1/\|\boldsymbol{A}\|_2^{-1} \leqslant \rho_{\boldsymbol{A}}(\boldsymbol{x})| \leqslant \|\boldsymbol{A}\|_2$$

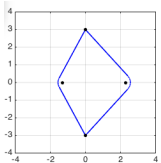
- If $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}$, so $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$ or $\boldsymbol{x}^H\boldsymbol{A} = \lambda\boldsymbol{x}^H$, then

$$\rho_{\boldsymbol{A}}(\boldsymbol{x}) = \lambda$$

- The set $\mathcal{F}_{\boldsymbol{A}} = \{\rho_{\boldsymbol{A}}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{C}^n, \boldsymbol{x} \neq 0\}$ is the *field of values* of $\boldsymbol{A}$

Field of Values and Eigenvalues

- Clearly any eigenvalue λ of $\boldsymbol{A}$ is in $\mathcal{F}_{\boldsymbol{A}}$
- For the matrix $\boldsymbol{A} = \begin{bmatrix} & 3 & & \\ -3 & & & \\ & & & 3 \\ & & 1 & 1 \end{bmatrix}$, $\mathcal{F}_{\boldsymbol{A}}$ is[1]



- The field of values of a normal matrix is easy to characterize
  - *If $\boldsymbol{A}$ is normal, $\mathcal{F}_{\boldsymbol{A}}$ is the convex hull of the eigenvalues.*
  - *If $\boldsymbol{A}$ is Hermitian and positive definite, $\mathcal{F}_{\boldsymbol{A}} = [\sigma_{min}, \sigma_{max}]$*
- In general, eigenvectors are obtained from critical points of the Rayleigh quotient on the unit circle

$$\mathcal{L}_{\boldsymbol{A}}(\boldsymbol{x}, \lambda) = \boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x} + \lambda(1 - \boldsymbol{x}^H \boldsymbol{x})$$

$$\nabla \mathcal{L}_{\boldsymbol{A}}(\boldsymbol{x}, \lambda) = \begin{bmatrix} 2\boldsymbol{A}\boldsymbol{x} - 2\lambda\boldsymbol{x} \\ 1 - \boldsymbol{x}^H \boldsymbol{x} \end{bmatrix} = 0,$$

[1] Credit to `https://www.chebfun.org/examples/linalg/FieldOfValues.html`

Singular Vectors as Critical Points

- Like eigenvectors, we can also derive singular vectors from an optimization (critical point) perspective
  - *Again, consider the critical points of the Lagrangian function of an optimization problem on the unit-sphere,*

$$\mathcal{L}_{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v}, \lambda_1, \lambda_2) = 2\boldsymbol{u}^H \boldsymbol{A}\boldsymbol{v} + \lambda_1(1 - \boldsymbol{u}^H\boldsymbol{u}) + \lambda_2(1 - \boldsymbol{v}^H\boldsymbol{v})$$

$$\nabla\mathcal{L}_{\boldsymbol{A}}(\boldsymbol{u}, \boldsymbol{v}, \lambda_1, \lambda_2) = \begin{bmatrix} 2\boldsymbol{A}\boldsymbol{v} - 2\lambda_1\boldsymbol{u} \\ 2\boldsymbol{A}^H\boldsymbol{u} - 2\lambda_2\boldsymbol{v} \\ 1 - \boldsymbol{u}^H\boldsymbol{u} \\ 1 - \boldsymbol{v}^H\boldsymbol{v} \end{bmatrix} = 0,$$

  - *At a critical point, we can see that $\lambda_1 = \lambda_2$, since $\boldsymbol{u}^H\boldsymbol{A}\boldsymbol{v} = \lambda_1 = \lambda_2$.*

Matrix Functions

- Consider a polynomial p, for a diagonalizable matrix $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1}$,

$$p(\boldsymbol{A}) = \boldsymbol{X}p(\boldsymbol{D})\boldsymbol{X}^{-1}$$

$$\begin{aligned} p(\boldsymbol{A}) &= \sum_{i=0}^{deg(p)} c_i \boldsymbol{A}^i = \sum_{i=0}^{deg(p)} c_i \prod_{j=1}^{i} \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1} \\ &= \sum_{i=0}^{deg(p)} c_i \boldsymbol{X}\boldsymbol{D}^i\boldsymbol{X}^{-1} = \boldsymbol{X}\left(\sum_{i=0}^{deg(p)} c_i \boldsymbol{D}^i\right)\boldsymbol{X}^{-1} \end{aligned}$$

- The above definition readily extends to other analytic functions f, but non-diagonalizable matrices require a more sophisticated definition

Crouzeix's conjecture

- So far, we have seen close connections between the matrix 2-norm and singular values, and between the Rayleigh quotient and the eigenvalues
- An important open problem in numerical analysis that relates the norm with the Rayleigh quotient is Crouzeix's conjecture
  - *For any polynomial p and complex matrix $\boldsymbol{A}$,*

$$\|p(\boldsymbol{A})\|_2 \leqslant 2 \max_{z \in \mathcal{F}_A} |p(z)|$$

  - *The conjecture is known to hold for some subclasses of matrices and with constant 11.08 instead of 2 (Crouzeix's theorem)*
  - *If valid, the bound of 2 is tight, including for $p(\boldsymbol{A}) = \boldsymbol{A}$, by choosing $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$*

Computing Eigenvalue and Singular Value Decompositions

- Direct methods for eigenvalue problems start by reducing the matrix to upper-Hessenberg form
  - *Seek a sequence of unitary similarity transformations* $\boldsymbol{H} = \boldsymbol{Q}_k \cdots \boldsymbol{Q}_1 \boldsymbol{A} \boldsymbol{Q}_1^T \cdots \boldsymbol{Q}_k^T$ *so that* $\boldsymbol{H}$ *is zero below the first subdiagonal (upper-Hessenberg)*
  - *Can pick each* $\boldsymbol{Q}_i$ *as a Householder transformation acting on the last* $n - i$ *rows*
  - $O(n^3)$ *cost to reduce to upper-Hessenberg or tridiagonal if symmetric*
  - *To obtain singular vectors, can work with* $\boldsymbol{A}^T \boldsymbol{A}$ *or perform 'bidiagonal reduction'*
  - *If matrix is sparse, fill may be introduced*
- Iterative methods are generally based on products with the matrix
  - *Power iteration converges to the largest eigenvalue eigenvectors of* $\boldsymbol{A}$
  - *Convergence rate is linear and depends on ratio of two largest eigenvalues*
  - *Integrating diagonal shifts and inversion yields other methods: inverse iteration, Rayleigh-quotient iteration*
  - *Most iterative methods involve only products with* $\boldsymbol{A}$ *or a related matrix*

Introduction to Krylov Subspace Methods

- *Krylov subspace methods* work with information contained in the $n \times k$ matrix

$$\boldsymbol{K}_k = \begin{bmatrix} \boldsymbol{x_0} & \boldsymbol{A x_0} & \cdots & \boldsymbol{A}^{k-1}\boldsymbol{x_0} \end{bmatrix}$$

We seek to best use the information from the matrix vector product results (columns of $\boldsymbol{K}_k$) to solve eigenvalue problems.

- Assuming $\boldsymbol{K}_n$ is invertible, the matrix $\boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n$ is a *companion matrix* $\boldsymbol{C}$:

Letting $\boldsymbol{k}_n^{(i)} = \boldsymbol{A}^{i-1}\boldsymbol{x}$, we observe that

$$\boldsymbol{A}\boldsymbol{K}_n = \begin{bmatrix} \boldsymbol{A}\boldsymbol{k}_n^{(1)} & \cdots & \boldsymbol{A}\boldsymbol{k}_n^{(n-1)} & \boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{k}_n^{(2)} & \cdots & \boldsymbol{k}_n^{(n)} & \boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix},$$

therefore premultiplying by $\boldsymbol{K}_m^{-1}$ transforms the first $n-1$ columns of $\boldsymbol{A}\boldsymbol{K}_n$ into the last $n-1$ columns of $\boldsymbol{I}$,

$$\begin{aligned} \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n &= \begin{bmatrix} \boldsymbol{K}_n^{-1}\boldsymbol{k}_n^{(2)} & \cdots & \boldsymbol{K}_n^{-1}\boldsymbol{k}_n^{(n)} & \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{e}_2 & \cdots & \boldsymbol{e}_n & \boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{k}_n^{(n)} \end{bmatrix} \end{aligned}$$

Krylov Subspaces

- Given $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{K}_k$, we obtain an orthonormal basis for the Krylov subspace,

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{x}_0) = \text{span}(\boldsymbol{Q}_k) = \{p(\boldsymbol{A})\boldsymbol{x}_0 : \deg(p) < k\},$$

  where p is any polynomial of degree less than k.

- The Krylov subspace includes the $k-1$ approximate dominant eigenvectors generated by $k-1$ steps of power iteration:
  - *The approximation obtained from $k-1$ steps of power iteration starting from $\boldsymbol{x}_0$ is given by the Rayleigh-quotient of $\boldsymbol{y} = \boldsymbol{A}^k \boldsymbol{x}_0$.*
  - *This vector is within the Krylov subspace, $\boldsymbol{y} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{x}_0)$.*
  - *Consequently, Krylov subspace methods will generally obtain strictly better approximations of the dominant eigenpair than power iteration.*

Rayleigh-Ritz Procedure

- The eigenvalues/eigenvectors of $\boldsymbol{H}_k$ are the *Ritz values/vectors*:

$$\boldsymbol{H}_k = \boldsymbol{X}\boldsymbol{D}\boldsymbol{X}^{-1}$$

eigenvalue approximations based on Ritz vectors $\boldsymbol{X}$ are given by $\boldsymbol{Q}_k\boldsymbol{X}$.

- The Ritz vectors and values are the *ideal approximations* of the actual eigenvalues and eigenvectors based on only $\boldsymbol{H}_k$ and $\boldsymbol{Q}_k$:

Assuming $\boldsymbol{A}$ is a symmetric matrix with positive eigenvalues, the largest Ritz value $\lambda_{max}(\boldsymbol{H}_k)$ will be the maximum Rayleigh quotient of any vector in $\mathcal{K}_k = span(\boldsymbol{Q}_k)$,

$$\max_{\boldsymbol{x}\in span(\boldsymbol{Q}_k)} \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \max_{\boldsymbol{y}\neq 0} \frac{\boldsymbol{y}^T\boldsymbol{Q}_k^T\boldsymbol{A}\boldsymbol{Q}_k\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \max_{\boldsymbol{y}\neq 0} \frac{\boldsymbol{y}^T\boldsymbol{H}_k\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} = \lambda_{max}(\boldsymbol{H}_k),$$

which is the best approximation to $\lambda_{max}(\boldsymbol{A}) = \max_{\boldsymbol{x}\neq 0} \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ available in $\mathcal{K}_k$. The quality of the approximation can also be shown to be optimal for other eigenvalues/eigenvectors.

Arnoldi Iteration

- Arnoldi iteration computes the ith column of $\boldsymbol{H}_n$, $\boldsymbol{h}_i$ and the ith column of $\boldsymbol{Q}_n$ directly using the recurrence $\boldsymbol{A}\boldsymbol{q}_i = \boldsymbol{Q}_n\boldsymbol{h}_i = \sum_{j=1}^{i+1} h_{ji}\boldsymbol{q}_j$
  - *Note that*
    $$\boldsymbol{q}_i^T\boldsymbol{A}\boldsymbol{q}_j = \boldsymbol{q}_i^T(\boldsymbol{Q}_n\boldsymbol{H}_n\boldsymbol{Q}_n^T)\boldsymbol{q}_j = \boldsymbol{e}_i^T\boldsymbol{H}_n\boldsymbol{e}_j = h_{ij}.$$
  - *The Arnoldi algorithm computes $\boldsymbol{q}_{i+1}$ from $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_i$ by first computing $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{q}_i$ then orthogonalizing,*

    $$\boldsymbol{q}_{i+1}h_{i+1,i} = \boldsymbol{u}_i - \sum_{j=1}^{i} \boldsymbol{q}_j h_{ji}, \quad h_{ji} = \boldsymbol{q}_j^T\boldsymbol{u}_i$$

    then computing the norm of the vector to obtain $h_{i+1,i}$, yielding the ith column of $\boldsymbol{H}_n$.

Multidimensional Optimization

- Minimize $f(\boldsymbol{x})$
  - *In the context of constrained optimization, also have equality and or inequality constraints, e.g., $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ or $\boldsymbol{x} > 0$*
  - *Unconstrained local optimality holds if $\nabla f(\boldsymbol{x}^*) = 0$ and $H_f(\boldsymbol{x}^*)$ is positive semi-definite*
  - *Reduces to solving nonlinear equations via optimality condition*
  - *Unconstrained local optimality conditions are looser, need the gradient to be zero or positive in all unconstrained directions at $\boldsymbol{x}^*$*
  - *The condition $\nabla f(\boldsymbol{x}^*) = 0$ implies poor conditioning, perturbations that change the function value in the kth digit can change the sollution in the $(k/2)$th digit*
- Quadratic optimization $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$
  - *Quadratic optimization problems can provide local approximations to general nonlinear optimization problems via Newton's method (where $\boldsymbol{A}$ is the Hessian and $\boldsymbol{b}^T$ is the gradient)*
  - *Equivalent to solving linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by optimality condition*
  - *Accordingly, conditioning relative to perturbation in $\boldsymbol{b}$ is $\kappa(\boldsymbol{A})$*

Basic Multidimensional Optimization Methods

- Steepest descent: minimize f in the direction of the negative gradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$$

such that $f(\boldsymbol{x}_{k+1}) = \min_{\alpha_k} f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k))$*, i.e. perform a line search (solve 1D optimization problem) in the direction of the negative gradient.*

- Given quadratic optimization problem $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x}$ where $\boldsymbol{A}$ is symmetric positive definite, the error $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ satisfies

$$||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}} = \boldsymbol{e}_{k+1}^T \boldsymbol{A} \boldsymbol{e}_{k+1} = \frac{\sigma_{\mathsf{max}}(\boldsymbol{A}) - \sigma_{\mathsf{min}}(\boldsymbol{A})}{\sigma_{\mathsf{max}}(\boldsymbol{A}) + \sigma_{\mathsf{min}}(\boldsymbol{A})} ||\boldsymbol{e}_k||_{\boldsymbol{A}}$$

  - When sufficiently close to a local minima, general nonlinear optimization problems are described by such an SPD quadratic problem.
  - Convergence rate depends on the conditioning of $\boldsymbol{A}$, since

$$\frac{\sigma_{max}(\boldsymbol{A}) - \sigma_{min}(\boldsymbol{A})}{\sigma_{max}(\boldsymbol{A}) + \sigma_{min}(\boldsymbol{A})} = \frac{\kappa(\boldsymbol{A}) - 1}{\kappa(\boldsymbol{A}) + 1}.$$

Gradient Methods with Extrapolation

- We can improve the constant in the linear rate of convergence of steepest descent by leveraging *extrapolation methods*, which consider two previous iterates (maintain *momentum* in the direction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$$

- The *heavy ball method*, which uses constant $\alpha_k = \alpha$ and $\beta_k = \beta$, achieves better convergence than steepest descent:

$$||\boldsymbol{e}_{k+1}||_{\boldsymbol{A}} = \frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1} ||\boldsymbol{e}_k||_{\boldsymbol{A}}$$

Nesterov's gradient optimization method is another instance of an extrapolation method that provides further improved optimality guarantees.

Conjugate Gradient Method

- The *conjugate gradient method* is capable of making the optimal (for a quadratic objective) choice of α_k and β_k at each iteration of an extrapolation method:

$$(\alpha_k, \beta_k) = \underset{\alpha_k, \beta_k}{\operatorname{argmin}} \left[f\Big(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) + \beta_k (\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \Big) \right]$$

  - *For SPD quadratic programming problems, conjugate gradient is an optimal first order method, converging in n iterations.*
  - *It implicitly computes Lanczos iteration, searching along $\boldsymbol{A}$-orthogonal directions at each step.*

- *Parallel tangents* implementation of the method proceeds as follows

  1. *Perform a step of steepest descent to generate $\hat{\boldsymbol{x}}_k$ from $\boldsymbol{x}_k$.*
  2. *Generate $\boldsymbol{x}_{k+1}$ by minimizing over the line passing through $\boldsymbol{x}_{k-1}$ and $\hat{\boldsymbol{x}}_k$.*

The method is equivalent to CG for a quadratic objective function.

Krylov Optimization

- Conjugate gradient (CG) finds the minimizer of $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ (which satisfies optimality condition $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$) within the Krylov subspace of $\boldsymbol{A}$:
  - *It constructs Krylov subspace* $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) = span(\boldsymbol{b}, \boldsymbol{A}\boldsymbol{b}, \ldots, \boldsymbol{A}^{r-1}\boldsymbol{b})$.
  - *At the* k*th step conjugate gradient yields iterate*

$$\boldsymbol{x}_k = ||\boldsymbol{b}||_2 \boldsymbol{Q}_k \boldsymbol{T}_k^{-1} \boldsymbol{e}_1,$$

where $\boldsymbol{Q}_k$ *is an orthogonal basis for Krylov subspace* $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ *and* $\boldsymbol{T}_k = \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k$.

  - *This choice of* $\boldsymbol{x}_k$ *minimizes* $f(\boldsymbol{x})$ *since*

$$\begin{aligned}
\min_{\boldsymbol{x} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})} f(\boldsymbol{x}) &= \min_{\boldsymbol{y} \in \mathbb{R}^k} f(\boldsymbol{Q}_k \boldsymbol{y}) \\
&= \min_{\boldsymbol{y} \in \mathbb{R}^k} \boldsymbol{y}^T \boldsymbol{Q}_k^T \boldsymbol{A} \boldsymbol{Q}_k \boldsymbol{y} - \boldsymbol{b}^T \boldsymbol{Q}_k \boldsymbol{y} \\
&= \min_{\boldsymbol{y} \in \mathbb{R}^k} \boldsymbol{y}^T \boldsymbol{T}_k \boldsymbol{y} - ||\boldsymbol{b}||_2 \boldsymbol{e}_1^T \boldsymbol{y}
\end{aligned}$$

is minimized by $\boldsymbol{y} = ||\boldsymbol{b}||_2 \boldsymbol{T}_k^{-1} \boldsymbol{e}_1$.

Conjugate Gradient Method: Optimized Form

After initialization $\boldsymbol{x}_0 = 0$, $\boldsymbol{r}_0 = \boldsymbol{b}$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$, the kth iteration of CG computes

$$\begin{aligned}
\boldsymbol{q}_k &= \boldsymbol{A}\boldsymbol{p}_k \\
\alpha_k &= \frac{\boldsymbol{r}_k^T \boldsymbol{r}_k}{\boldsymbol{q}_k^T \boldsymbol{p}_k} \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \\
\boldsymbol{r}_{k+1} &= \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k
\end{aligned}$$

At this point if the residual norm ($\|\boldsymbol{r}_{k+1}\|$) is small, terminate, otherwise prepare for next iteration,

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \frac{\boldsymbol{r}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{r}_k^T \boldsymbol{r}_k} \boldsymbol{p}_k$$

See Jonathan Shewchuk 1994 notes on CG or James Demmel's book for the derivation of this form of the algorithm.

Conjugate Gradient Convergence Analysis

- In previous discussion, we assumed $\boldsymbol{K}_n$ is invertible, which may not be the case if $\boldsymbol{A}$ has $k < n$ distinct eigenvalues, however, then CG converges in $k-1$ iterations (in exact arithmetic)
  - *To prove this, we can analyze the 'minimizing' polynomials in the Krylov subspace in terms of the (real and positive) eigenvalues of $\boldsymbol{A}$*
  - *The approximate solution $\boldsymbol{x}_k$ obtained by CG after $k-1$ iterations is given by minimizing $\boldsymbol{z} \in \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$, which means $\boldsymbol{z} = \boldsymbol{\rho}_{k-1}(\boldsymbol{A})\boldsymbol{b}$ for some polynomial ρ_{k-1} of degree $k-1$*
  - *Now, consider the residual*

$$\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = (\boldsymbol{A}\boldsymbol{\rho}_{k-1}(\boldsymbol{A}) - \boldsymbol{I})\boldsymbol{b}$$

  - *Choosing ρ_{k-1} as a polynomial interpolant so that $\rho_{k-1}(\lambda) = 1/\lambda$ for $\lambda \in \lambda(\boldsymbol{A})$, results in a zero residual since then $\boldsymbol{\rho}_{k-1}(\boldsymbol{A}) = \boldsymbol{A}^{-1}$.*

Round-off Error in Conjugate Gradient

- CG provides strong convergence guarantees for SPD matrices in exact arithmetic
  - Classically, CG was viewed as a direct method, since its guaranteed to convergence in n iterations
  - In practice, round-off error prevents CG from achieving this for realistic matrices, so CG was actually abandoned for a while due to being viewed as unstable
  - Later, it was realized that CG is highly competitive as an iterative method
- Due to round-off CG may stagnate / have plateaus in convergence
  - A formal analysis of round-off error[2] reveals that CG with round-off is equivalent to exact CG on a matrix of larger dimension, whose eigenvalues are clustered around those of $\boldsymbol{A}$
  - Using this view, CG convergence plateaus may be explained by the polynomial q_k developing more and more zeros near the same eigenvalue of $\boldsymbol{A}$

[2] A. Greenbaum and Z. Strakos, SIMAX 1992

Preconditioning

- Convergence of iterative methods for $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ depends on $\kappa(\boldsymbol{A})$, the goal of a preconditioner $\boldsymbol{M}$ is to obtain $\boldsymbol{x}$ by solving

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

  with $\kappa(\boldsymbol{M}^{-1}\boldsymbol{A}) < \kappa(\boldsymbol{A})$
  - need not form $\boldsymbol{M}^{-1}\boldsymbol{A}$ but only compute matrix-vector products $\boldsymbol{M}^{-1}(\boldsymbol{A}\boldsymbol{x})$
  - want $\boldsymbol{M}^{-1}\boldsymbol{x}$ to be easy to compute (easier than $\boldsymbol{A}^{-1}\boldsymbol{x}$)
  - so generally one extracts some $\boldsymbol{M} \approx \boldsymbol{A}$ that is easy to solve linear systems with
  - however, $\boldsymbol{M} \approx \boldsymbol{A}$ may be insufficient/unnecessary, primary goal is to improve conditioning to accelerate iterative methods, i.e., want $\kappa(\boldsymbol{M}^{-1}\boldsymbol{A}) \ll \kappa(\boldsymbol{A})$
- Common preconditioners select parts of $\boldsymbol{A}$ or perform inexact factorization
  - (block-)Jacobi preconditioner takes $\boldsymbol{M}$ to be (block-)diagonal of $\boldsymbol{A}$
  - incomplete LU (ILU) preconditioners compute $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{U} \approx \boldsymbol{A}$ (+pivoting)
  - ILU variants constraint sparsity of $\boldsymbol{L}$ and $\boldsymbol{U}$ factors during factorization to be the same or not much more than that of $\boldsymbol{A}$
  - good problem-specific preconditioners are often available in practice and theory, applying also to problems beyond linear systems (eigenvalue problems, optimization, approximate graph algorithms)

Newton's Method

- Newton's method in n dimensions is given by finding minima of n-dimensional quadratic approximation using the gradient and Hessian of f:

$$f(\boldsymbol{x}_k + \boldsymbol{s}) \approx \hat{f}(\boldsymbol{s}) = f(\boldsymbol{x}_k) + \boldsymbol{s}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2} \boldsymbol{s}^T \boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s}.$$

The minima of this function can be determined by identifying critical points

$$\boldsymbol{0} = \nabla \hat{f}(\boldsymbol{s}) = \nabla f(\boldsymbol{x}_k) + \boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s},$$

thus to determine $\boldsymbol{s}$ we solve the linear system,

$$\boldsymbol{H}_f(\boldsymbol{x}_k) \boldsymbol{s} = -\nabla f(\boldsymbol{x}_k).$$

Assuming invertibility of the Hessian, we can write the Newton's method iteration as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \underbrace{\boldsymbol{H}_f(\boldsymbol{x}_k)^{-1} \nabla f(\boldsymbol{x}_k)}_{\boldsymbol{s}}.$$

Quadratic convergence follows by fixed point function analysis, beyond smoothness, a sufficient assumption is that $\boldsymbol{H}_f(\boldsymbol{x}^*)$ is SPD.

Nonlinear Least Squares

- An important special case of multidimensional optimization is *nonlinear least squares*, the problem of fitting a nonlinear function $f_{\boldsymbol{x}}(t)$ so that $f_{\boldsymbol{x}}(t_i) \approx y_i$:
  For example, consider fitting $f_{[x_1,x_2]}(t) = x_1 \sin(x_2 t)$ *so that*

$$\begin{bmatrix} f_{[x_1,x_2]}(1.5) \\ f_{[x_1,x_2]}(1.9) \\ f_{[x_1,x_2]}(3.2) \end{bmatrix} \approx \begin{bmatrix} -1.2 \\ 4.5 \\ 7.3 \end{bmatrix}.$$

- We can cast nonlinear least squares as an optimization problem to minimize residual error and solve it by Newton's method:
  Define residual vector function $\boldsymbol{r}(\boldsymbol{x})$ *so that* $r_i(\boldsymbol{x}) = y_i - f_{\boldsymbol{x}}(t_i)$ *and minimize*

$$\phi(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{r}(\boldsymbol{x})||_2^2 = \frac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x}).$$

  Now the gradient is $\nabla\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})$ *and the Hessian is*

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

Gauss-Newton Method

- The Hessian for nonlinear least squares problems has the form:

$$\boldsymbol{H}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x})\boldsymbol{H}_{r_i}(\boldsymbol{x}).$$

The second term is small when the residual function $\boldsymbol{r}(\boldsymbol{x})$ is small, so approximate

$$\boldsymbol{H}_\phi(\boldsymbol{x}) \approx \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}) = \boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x})\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}).$$

- The *Gauss-Newton* method is Newton iteration with an approximate Hessian:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \hat{\boldsymbol{H}}_\phi(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k) = \boldsymbol{x}_k - (\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k))^{-1}\boldsymbol{J}_{\boldsymbol{r}}^T(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

Recognizing the normal equations, we interpret the Gauss-Newton method as solving linear least squares problems $\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{x}_k)\boldsymbol{s}_k \cong \boldsymbol{r}(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{s}_k$.

Low Rank Matrix Approximation

- Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ seek rank $r < m, n$ approximation
  - *Given by matrices $\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times r}$ so*

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}^T$$

  - *Reduces memory footprint and cost of applying $\boldsymbol{A}$ from mn to $mr + nr$*
  - *This factorization is nonunique, $\boldsymbol{U}\boldsymbol{V}^T = (\boldsymbol{U}\boldsymbol{M})(\boldsymbol{V}\boldsymbol{M}^{-T})^T$*
- Eckart-Young (optimal low-rank approximation by SVD) theorem
  - *Truncated SVD approximates $\boldsymbol{A}$ as*

$$\boldsymbol{A} \approx \tilde{\boldsymbol{A}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

    where $\sigma_1, \ldots, \sigma_r$ are the largest r singular values, while $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the associated left and right singular vectors
  - *Eckart-Young theorem demonstrates that the truncated SVD minimizes*

$$\underbrace{\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_2}_{\sigma_{r+1}} \quad \textit{and} \quad \underbrace{\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_F^2}_{\sum_{i=r+1}^{\min(m,n)} \sigma_i^2}$$

Rank Revealing Matrix Factorizations

- Computing the SVD
  - *Can compute full SVD with $O(mn\min(m,n))$ cost via bidiagonalization*
    - *unconditionally stable and accurate*
    - *inefficient for low r or if $\boldsymbol{A}$ is sparse*
  - *Given any low-rank approximation composed of $\boldsymbol{U}$ and $\boldsymbol{V}$, compute QR of each and SVD of product of $\boldsymbol{R}$ factors to obtain SVD with total cost $O((m+n)r^2)$*
- QR with column pivoting
  - *By selecting columns of largest norm in the trailing matrix during QR factorization, we obtain a pivoted factorization with permutation matirx $\boldsymbol{P}$*

$$\boldsymbol{AP} = \boldsymbol{QR}$$

  - *Truncating this factorization can be done after applying r Householder reflectors (or another QR algorithm on r columns), with cost $O((m+n)r)$*
  - *Approximation is somewhat suboptimal in theory, but in practice almost always as accurate as truncated SVD*

Simultaneous and Orthogonal Iteration

- Orthogonal iteration computing many eigenvectors at once in an iterative way
  - *Initialize $\boldsymbol{X}_0 \in \mathbb{R}^{n \times k}$ to be random, orthogonalize it to obtain $\boldsymbol{Q}_0$, then iterate via*

$$\boldsymbol{Q}_{i+1}\boldsymbol{R}_{i+1} = \boldsymbol{A}\boldsymbol{Q}_i$$

  - *For random starting guess, with high probability, $\lim_{i \to \infty} span(\boldsymbol{X}_i) = \mathbb{S}$ where $\mathbb{S}$ is the subspace spanned by the k eigenvectors of $\boldsymbol{A}$ with the largest eigenvalues in magnitude.*
  - *Can use this to compute the right singular vectors of matrix $\boldsymbol{M}$ by using $\boldsymbol{A} = \boldsymbol{M}^T\boldsymbol{M}$ (no need to form $\boldsymbol{A}$, just multiply $\boldsymbol{Q}_i$ by $\boldsymbol{M}^T$ then $\boldsymbol{M}$).*
  - *QR has cost $O(nk^2)$ while product has cost $O(n^2k)$ (or more generally, k products with $\boldsymbol{A}$) per iteration.*
  - *QR iteration performs orthogonal iteration implicitly when $n = k$*

Orthogonal Iteration Convergence

- If $\boldsymbol{A}$ has distinct eigenvalues and $\boldsymbol{R}_i$ has positive decreasing diagonal, the jth column of $\boldsymbol{Q}_i$ converges to the jth Schur vector of $\boldsymbol{A}$ linearly with rate $\max(|\lambda_{j+1}/\lambda_j|, |\lambda_j/\lambda_{j-1}|)$.
  - *Convergence of the first column of $\boldsymbol{Q}_i$ follows by analogy to power iteration*
  - *Span of first j columns of $\boldsymbol{Q}_i$ converges to the span of the first j Schur vectors with rate $|\lambda_{j+1}/\lambda_j|$*
  - *Hence orthogonal iteration converges similarly to k instances of inverse iteration with shifts chosen near the k largest magnitude eigenvalues*
  - *Block-Krylov methods, which consider span$\{\boldsymbol{X}_0, \boldsymbol{A}\boldsymbol{X}_0, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{X}_0\}$ provide some improvement over orthogonal iteration for low rank SVD (see works by Ming Gu and others)*

Randomized SVD

- Orthogonal iteration for SVD can also be viewed as a randomized algorithm
  - *Suppose that we have an exact low-rank factorization* $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ *with* $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$
  - *If* $\boldsymbol{Q}^{(0)}$ *is a random orthogonal matrix, so is* $\boldsymbol{V}^T\boldsymbol{Q}^{(0)}$
  - *Consequently,* $\boldsymbol{A}\boldsymbol{Q}^{(0)}$ *is a set of* r *random linear combinations of columns of* $\boldsymbol{U}\boldsymbol{\Sigma}$
  - *Further, after the QR* $\boldsymbol{Q}^{(1)}\boldsymbol{R}^{(1)} = \boldsymbol{A}\boldsymbol{Q}^{(0)}$,

$$\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{Q}^{(1)}\boldsymbol{Q}^{(1)T}$$

    holds with probability 1 (suffices to have $\boldsymbol{A}\boldsymbol{Q}^{(0)}$ *full rank)*
  - *Consequently, we can compute SVD of* $\boldsymbol{Q}^{(1)T}\boldsymbol{A}$ *(with cost* $O(nr^2)$*) and recover* $\boldsymbol{U}$ *by premultiplying the computed left singular vectors by* $\boldsymbol{Q}^{(1)}$
  - *When* $\boldsymbol{A}$ *is not exactly low-rank, span of leading singular vectors can be captured by oversampling (e.g., selecting each* $\boldsymbol{Q}^{(i)}$ *to have* $r + 10$ *columns)*
  - *Initial guess* $\boldsymbol{Q}^{(0)}$ *need not be orthogonal (Gaussian random performs well, structured pseudo-random enables* $O(mn \log n)$ *complexity for one-shot randomized SVD), but better accuracy is obtained with orthogonality*

Generalized Nyström Algorithm

- The generalized Nyström algorithm provides an efficient way of computing a low-rank factorization given an approximation of its span[3]
  - *Given matrices $\boldsymbol{S}_1 \in \mathbb{R}^{k \times n}$ and $\boldsymbol{S}_2 \in \mathbb{R}^{k \times m}$ the rank k factorization of a matrix $\boldsymbol{A} \in \mathbb{R}m \times n$ is obtained via*

$$\hat{\boldsymbol{A}}_k = \boldsymbol{A}\boldsymbol{S}_1^T(\boldsymbol{S}_2\boldsymbol{A}\boldsymbol{S}_1^T)^\dagger \boldsymbol{S}_2\boldsymbol{A}$$

  - *The truncated SVD is recovered if $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ contain the largest eigenvectors*
  - *Generally, we expect $\boldsymbol{S}_2\boldsymbol{A}\boldsymbol{S}_1^T$ to be full rank, otherwise factorization is rank-deficient*
  - *If $\boldsymbol{S}_2\boldsymbol{A}\boldsymbol{S}_1^T$ is invertible, $\forall \boldsymbol{u}, \boldsymbol{A}\boldsymbol{S}_1^T\boldsymbol{u} = \hat{\boldsymbol{A}}_k\boldsymbol{S}_1^T\boldsymbol{u}$*
  - *The skeleton decomposition is obtained by choosing both $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ to be sampling matrices (each row being a unit vector)*
  - *Instead, $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ may be chosen as random 'sketch matrices'*
  - *The interpolative decomposition is obtained by choosing either of the two to be a sampling matrix.*

[3] Nakatsukasa, Yuji, Fast and stable randomized low-rank matrix approximation, 2020.

Analysis of Generalized Nyström Algorithm

- Consider $\boldsymbol{F}_1 = \boldsymbol{A}\boldsymbol{S}_1^T$ and $\boldsymbol{F}_2 = \boldsymbol{A}\boldsymbol{S}_2^T$, derive the minimizer $\boldsymbol{Z}$ to

$$\|\boldsymbol{A} - \boldsymbol{F}_1\boldsymbol{Z}\boldsymbol{F}_2^T\|_F$$

$$\begin{aligned}
\operatorname{vec}(\boldsymbol{A} - \boldsymbol{F}_1\boldsymbol{Z}\boldsymbol{F}_2^T) &= \operatorname{vec}(\boldsymbol{A}) - \operatorname{vec}(\boldsymbol{F}_1 \otimes \boldsymbol{F}_2)\operatorname{vec}(\boldsymbol{Z}) \\
\operatorname{vec}(\boldsymbol{Z}) &= \operatorname{vec}(\boldsymbol{F}_1 \otimes \boldsymbol{F}_2)^+ \operatorname{vec}(\boldsymbol{A}) \\
&= \operatorname{vec}(\boldsymbol{F}_1^+ \otimes \boldsymbol{F}_2^+)\operatorname{vec}(\boldsymbol{A}) \\
&= \boldsymbol{F}_1^+\boldsymbol{A}(\boldsymbol{F}_2^+)^T
\end{aligned}$$

- The generalized Nyström algorithm may be interpreted as applying a two-sided oblique projection of $\boldsymbol{A}$
  - *Optimal solution above is given by orthogonal projections $\boldsymbol{F}_1\boldsymbol{F}_1^+$ and $\boldsymbol{F}_2\boldsymbol{F}_2^+$*
  - *Generalized Nyström approximation instead uses the oblique projections*

$$\boldsymbol{P}_1 = \boldsymbol{A}\boldsymbol{S}_1(\boldsymbol{S}_2\boldsymbol{A}\boldsymbol{S}_1^T)^+\boldsymbol{S}_2, \boldsymbol{P}_2 = \boldsymbol{S}_1^T(\boldsymbol{S}_2\boldsymbol{A}\boldsymbol{S}_1^T)^+\boldsymbol{S}_2\boldsymbol{A}$$

where $\boldsymbol{P}_1\boldsymbol{P}_1 = \boldsymbol{P}_1$ and $\boldsymbol{P}_2\boldsymbol{P}_2 = \boldsymbol{P}_2$, while the approximation obtained via $\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{P}_2$