Midterm Review

15-369/669/769: Numerical Computing

Instructor: Minchen Li

Midterm Exam

- **Time:** 80 min, in-class on Wednesday, Oct 8
- 7 questions in total
- 1 double-sided letter-size cheat sheet allowed (can be printed)
- Calculator allowed

- A highly relevant practice exam with answers will be provided today.
- Assignment answers will be provided today.

1. Mathematics Review

Vector Spaces and Linearity

- **Examples:**
  - $\mathbb{R}^n$: standard Euclidean space.
  - $\text{span}\{(1,0,0),(0,1,0),(0,0,1)\} = \mathbb{R}^3$.

- **Vector space:** a set closed under vector addition and scalar multiplication.

$$\forall \mathbf{u}, \mathbf{v} \in V,\ \forall c \in \mathbb{R}, \quad \mathbf{u} + \mathbf{v} \in V,\ c\mathbf{u} \in V.$$

- **Linear combination:** $\mathbf{v} = \sum_i \alpha_i \mathbf{v}_i$. The set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is *linearly independent* if no nontrivial coefficients α_i satisfy the above.

- **Span:** all linear combinations of a set of vectors; forms a subspace.

- **Basis:** a minimal linearly independent set that spans V. Dimensionality = number of basis vectors.

1. Mathematics Review

Linear Maps and Matrix Representation

- A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is **linear** if

$$f(a\mathbf{x} + b\mathbf{y}) = af(\mathbf{x}) + bf(\mathbf{y}) \quad \forall a, b, \mathbf{x}, \mathbf{y}.$$

- Any linear map can be represented by a matrix A:

$$f(\mathbf{x}) = A\mathbf{x}, \quad A \in \mathbb{R}^{m \times n}.$$

- **Key matrix operations:**
  - Matrix-vector product: $A\mathbf{x}$
  - Matrix-matrix product: AB
  - Transpose: A^T
  - Identity: $I\mathbf{x} = \mathbf{x}$

- **Solving linear systems:** $A\mathbf{x} = \mathbf{b}$
  - If A invertible: $\mathbf{x} = A^{-1}\mathbf{b}$.
  - Residual norm: $\|A\mathbf{x} - \mathbf{b}\|_2$ measures error.

1. Mathematics Review

Differential Calculus and Nonlinearity

- **Derivative (1D):** measures rate of change: $f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$.
- **Taylor's theorem:** for small h, $f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3)$.
- **Gradient:** for $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T$.
- **Directional derivative:** rate of change along direction $\mathbf{v}$: $D_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x})^\top \mathbf{v}$.
- **Jacobian:** derivative of a vector-valued function $F : \mathbb{R}^n \to \mathbb{R}^m$: $J_{ij} = \frac{\partial F_i}{\partial x_j}$.
- Quadratic form: $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$, Hessian $= A + A^T$.

2. Numerics and Error Analysis

Representing Numbers in Computers

- **Floating-point representation:**
  - Inspired by scientific notation: $x = \pm(1.d_1 d_2 \ldots d_p)_b \times b^e$
  - Provides wide range of magnitudes and uniform relative precision.
  - Key parameters: Base b; precision p; exponent range $[e_{\min}, e_{\max}]$; machine precision ε_m: smallest $\varepsilon > 0$ such that $1 + \varepsilon$ is representable.
  - Widely used standard (base $b = 2$):
    - 1 sign bit, 52 fraction bits (mantissa), 11 exponent bits.
    - Double precision exponent range: -1022 to 1023.
    - Supports special values: $\pm\infty$, `NaN` ("not-a-number") for undefined results (e.g. $1/0$).

2. Numerics and Error Analysis

Representing Numbers in Computers (cont.)

- **Fixed-point representation:**
  - Stores integers with an implied decimal point (scaled by a power of two or ten).
  - Efficient: reuses integer arithmetic hardware.
  - Limitations: Limited precision; cannot represent irrational numbers (e.g.π, $1/3$).
  - Still used in embedded or real-time systems for efficiency.

- **Other representations:**
  - *Rational numbers:* exact arithmetic but expensive and nonunique.
  - *Error bars:* explicitly track uncertainty bounds.

2. Numerics and Error Analysis

Error Analysis and Conditioning

- **Sources of error:**
  - **Rounding error:** finite-digit representation (e.g. π not exact).
  - **Discretization error:** numerical approximation of continuous equations.
  - **Modeling error:** simplified physical or mathematical model.
  - **Input error:** inaccurate or noisy user-provided data.
- **Error types:**
  - Absolute error: $|x_{\text{est}} - x_{\text{true}}|$.
  - Relative error: $|x_{\text{est}} - x_{\text{true}}| / |x_{\text{true}}|$.
- **Forward vs. backward error:**
  - Backward error: how much input must change to make result exact.
  - Forward error: deviation of computed result from true result.
  - **Condition number:** amplification factor linking backward $\rightarrow$ forward error: $\kappa = \frac{|\text{forward error}|}{|\text{backward error}|}$.
    Small κ: well-conditioned, large κ: ill-conditioned.

3. Linear Systems and the LU Decomposition

Solving Linear Systems

- **Goal:** Solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$.
- **Solvability:**
  - **Geometric intuition:** $A\mathbf{x}$ is a linear combination of columns of A. The equation is solvable only if $\mathbf{b}$ lies within their span.
  - Three cases:
    - Overdetermined: more equations than unknowns.
    - Underdetermined: fewer equations than unknowns.
    - Square, nonsingular: unique solution (our focus in this lecture).

3. Linear Systems and the LU Decomposition

Gaussian Elimination and Row Operations

- **Gaussian elimination:** iteratively eliminate variables to reach upper-triangular form.
- Equivalent to performing *row operations:*
  - Row permutation (swap two rows).
  - Row scaling (multiply by nonzero scalar).
  - Row addition (add multiple of one row to another).
- Matrix form: $M_k M_{k-1} \cdots M_1 A = U, \quad$ where U is upper triangular.
- Solving system: $A\mathbf{x} = \mathbf{b} \Rightarrow U\mathbf{x} = M_k M_{k-1} \cdots M_1 \mathbf{b}$.
- **Forward/backward substitution:** triangular systems can be solved in $O(n^2)$ time.

3. Linear Systems and the LU Decomposition

LU Factorization and Practical Use

- **LU factorization:** $A = LU, \quad L$ (lower triangular), U (upper triangular).
- Solving $A\mathbf{x} = \mathbf{b}$: $LU\mathbf{x} = \mathbf{b} \Rightarrow \begin{cases} L\mathbf{y} = \mathbf{b} & \text{(forward substitution)} \\ U\mathbf{x} = \mathbf{y} & \text{(backward substitution)} \end{cases}$
- Complexity: Factorization: $O(n^3)$; Each solve (after factorization): $O(n^2)$.
- Pivoting: improves numerical stability and sparsity control.
  - Row pivoting: $A = PLU$
  - Full pivoting (rare): $A = PLUQ$
- Used for multiple right-hand sides or iterative solvers as a precomputation.

4. Designing Linear Systems

Parametric Regression

- **Goal:** predict outcomes $f(x)$ for new x given observed data pairs (x_i, y_i).
- **Linear regression:**
  - Model, e.g. polynomial regression, $f(x) \approx a_0 + a_1 x + \cdots + a_n x^n = \mathbf{a}^T \mathbf{x}$.
  - Data points collected into a design matrix A and target vector $\mathbf{b}$.
  - Solve for parameters $\mathbf{a}$ using least squares: $A\mathbf{a} = \mathbf{b}$
- **Nonlinear regression:**
  - $f(x)$ depends nonlinearly on model parameters.
  - Requires iterative solvers (e.g. Newton's method).

4. Designing Linear Systems

Least-Squares Formulation and Normal Equations

- **Motivation:** Interpolation forces the fitted curve through all data points – poor for noisy data.
- Instead, allow an approximate fit minimizing squared residuals:

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2.$$

- If A has full column rank, normal equations yield:

$$A^T A\mathbf{x} = A^T\mathbf{b} \quad \Rightarrow \quad \mathbf{x} = (A^T A)^{-1} A^T\mathbf{b}$$

- Used throughout data fitting, statistics, and inverse problems.

4. Designing Linear Systems

Regularization and Applications

- **Tikhonov (Ridge) Regularization:** $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \alpha\|\mathbf{x}\|_2^2$.
  - Equivalent linear system: $(A^T A + \alpha I)\mathbf{x} = A^T \mathbf{b}$.
  - Stabilizes inversion when A is ill-conditioned or rank-deficient.
  - $\alpha > 0$ controls the tradeoff between fit accuracy and smoothness.
- **Applications:**
  - **Image alignment:** match corresponding points between two images. Solve for transformation A, b minimizing $\sum_k \|A\mathbf{x}_k + \mathbf{b} - \mathbf{y}_k\|_2^2$, leading to a linear least-squares system.
  - Common pattern: model relationships $\rightarrow$ form residuals $\rightarrow$ minimize squared norms $\rightarrow$ derive linear equations.

5. Analyzing Linear Systems

Positive Definiteness and Cholesky Factorization

- For a least-squares system $A^T A\mathbf{x} = A^T\mathbf{b}$:
  - $A^T A$ is always symmetric: $(A^T A)^T = A^T A$.
  - $A^T A$ is positive semi-definite (PSD): $\forall \mathbf{x} \neq \mathbf{0}, \quad \mathbf{x}^T A^T A\mathbf{x} = \|A\mathbf{x}\|_2^2 \geq 0$.
  - $A^T A$ is symmetric positive definite (SPD), and thus invertible, if A's columns are linearly independent.
- **Cholesky factorization:** $A = LL^T, \quad L$ lower triangular.
  - Faster and more memory-efficient than LU (half the storage, half the flops).
  - Must ensure A is SPD.
  - Efficient algorithm: $\ell_{kk} = \sqrt{c_{kk} - \|\boldsymbol{\ell}_k\|_2^2}, \quad \boldsymbol{\ell}_k = L_{11}^{-1}\mathbf{c}_k$.

5. Analyzing Linear Systems

Sparsity

- **Sparse matrix:** majority of entries are zero.
- **Example:** in image processing, each pixel depends only on neighbors.
- **Storage:** store only nonzero entries (e.g., triplet list: row, column, value).
- **Solvers:**
  - LU or Cholesky factorization may introduce new nonzeros (fill-in).
  - Specialized direct solvers (e.g., sparse LU, multifrontal) minimize fill.
  - Iterative solvers (CG, GMRES) avoid explicit factorization; require only matrix-vector products.
- **Tridiagonal systems:** can be solved in $O(n)$ using specialized algorithms.

5. Analyzing Linear Systems

Sensitivity Analysis and Conditioning

- **Goal:** measure how small perturbations in A, $\mathbf{b}$ affect solution $\mathbf{x}$ in $A\mathbf{x} = \mathbf{b}$.
- **Vector and matrix norms:**
  - Vector p-norms: $\|\mathbf{v}\|_p = \left(\sum_i |v_i|^p\right)^{1/p}$, with $\|\mathbf{v}\|_p \leq \|\mathbf{v}\|_q$ for $p > q$.
  - Matrix norms (induced): $\|A\|_2 = \max_{\|\mathbf{x}\|_2=1} \|A\mathbf{x}\|_2 = \sigma_{\max}(A)$.
- **Condition number:** $\text{cond}(A) = \|A\|\|A^{-1}\| = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$.
  - Always ≥ 1; scaling A doesn't change its condition number.
  - Large $\text{cond}(A)$: system is ill-conditioned — small perturbations cause large solution errors.
- **Error amplification:** $\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq |\epsilon|\text{cond}(A)\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right) + O(\epsilon^2)$.

6. Column Spaces and QR

QR Factorization and Orthogonal Matrices

- **Motivation:** Solving least-squares systems $\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2$ via normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$ may lead to numerical instability when columns of A are nearly dependent.
- **Idea:** Replace $A^T A$ with orthogonalization via $A = QR$, where Q's columns are orthonormal ($Q^T Q = I$), and R is upper-triangular matrix.
- **Orthogonal matrix properties:** $Q^{-1} = Q^T$; Preserves lengths and angles: $\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2$, $(Q\mathbf{x})^T(Q\mathbf{y}) = \mathbf{x}^T Q^T Q\mathbf{y} = \mathbf{x}^T\mathbf{y}$.
- **Least-squares solution:** $A = QR \Rightarrow A^T A = R^T R, \quad A^T\mathbf{b} = R^T Q^T\mathbf{b}$, hence $R\mathbf{x} = Q^T\mathbf{b}$ can be solved efficiently by back-substitution.
- Advantage: avoids squaring condition number as in $A^T A$.

6. Column Spaces and QR

Gram-Schmidt Orthogonalization

- **Goal:** Construct an orthonormal basis $\{\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \ldots, \hat{\mathbf{a}}_k\}$ for the column space of A.
- **Projection of a vector:** $\text{proj}_{\mathbf{a}}(\mathbf{b}) = \frac{\mathbf{a}^T\mathbf{b}}{\mathbf{a}^T\mathbf{a}}\mathbf{a}$.
- **Classical Gram-Schmidt:** $\hat{\mathbf{a}}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|}$, $\hat{\mathbf{a}}_2 = \frac{\mathbf{a}_2 - (\hat{\mathbf{a}}_1^T\mathbf{a}_2)\hat{\mathbf{a}}_1}{\|\mathbf{a}_2 - (\hat{\mathbf{a}}_1^T\mathbf{a}_2)\hat{\mathbf{a}}_1\|}$, $\ldots$
- **Result:** $A = QR$, where Q is a matrix with orthonormal columns, and $R = Q^T A$ is upper triangular.
- **Modified Gram-Schmidt:**
  - Projects each remaining column immediately after computing each new basis vector.
  - More numerically stable under rounding errors.

6. Column Spaces and QR

Householder Transformations and Reduced QR

- **Motivation:** Gram-Schmidt is conceptually simple but numerically unstable. Instead, use orthogonal reflections (Householder, 1958).

- **Householder reflection:** $H = I - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}, \quad H^T H = I$. Reflects a vector $\mathbf{b}$ across direction $\mathbf{v}$.

- **Algorithm:**

  - Iteratively pre-multiply A by orthogonal reflectors $H_1, H_2, \ldots$ to eliminate subdiagonal entries.

  - After k steps: $R = H_k \cdots H_1 A, \quad Q = (H_1 H_2 \cdots H_k)^T$.

- **Reduced QR Factorization:** $A = Q_1 R_1, \quad Q_1 \in \mathbb{R}^{m\times n},\ R_1 \in \mathbb{R}^{n\times n}$.
  Used when $m \gg n$, saves storage. Solve least squares via $R_1\mathbf{x} = Q_1^T\mathbf{b}$.

- Householder QR is stable and efficient ($O(mn^2)$) and preferred in practice.

7. Eigenvectors

Motivation and Applications of Eigenvalue Problems

- **Eigenvalue problem:** find nontrivial pairs $(\lambda, \mathbf{x})$ such that $A\mathbf{x} = \lambda\mathbf{x}$.
  Nonlinear in unknowns $\lambda, \mathbf{x}$ due to their product; constraint $\|\mathbf{x}\|_2 = 1$ avoids the trivial solution $\mathbf{x} = \mathbf{0}$.
- **Optimization form:** $\frac{\partial}{\partial \mathbf{x}} \left(\mathbf{x}^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1)\right) = 0 \Rightarrow A\mathbf{x} = \lambda\mathbf{x}$.
- **Example (PCA in statistics):**
  - Patient data (age, weight, BP, heart rate) in $\mathbb{R}^4$ may lie approximately in a lower-dimensional subspace.
  - The best 1D subspace spanned by $\mathbf{v}$ minimizes projection error: $\sum_i \|\mathbf{x}_i - (\mathbf{v}^T\mathbf{x}_i)\mathbf{v}\|^2$.
  - Equivalent to maximizing variance: $\max_{\|\mathbf{v}\|=1} \mathbf{v}^T XX^T\mathbf{v}$, whose solution $\mathbf{v}$ is the principal eigenvector of XX^T (the first principal component).

7. Eigenvectors

Properties of Eigenvectors

- **Scale invariance:** if $A\mathbf{x} = \lambda\mathbf{x}$, then $A(c\mathbf{x}) = \lambda(c\mathbf{x})$ for any scalar c. Thus eigenvectors are determined up to scale and sign.
- **Multiplicity and diagonalizability:**
  - An $n \times n$ matrix has at most n distinct eigenvalues. If fewer than n linearly independent eigenvectors exist, the matrix is *defective*.
  - The number of linearly independent eigenvectors for λ is its *geometric multiplicity*.
- **Similarity transformation:** Two matrices A and B are similar if there exists an invertible matrix T such that $A = T^{-1}BT$. Similar matrices have the same eigenvalues.
- **Spectral theorem:** For real symmetric (or Hermitian) A: $A = Q\Lambda Q^T$, where Q is orthogonal and Λ is real diagonal.

7. Eigenvectors

Computing Eigenvalues and Eigenvectors

- **Power iteration:** Start from arbitrary $\mathbf{v}_0$; repeat $\mathbf{v}_{k+1} = A\mathbf{v}_k / \|A\mathbf{v}_k\|$.
  Converges to eigenvector of largest $|\lambda|$ if $|\lambda_1| > |\lambda_2|$. Simple, but slow if eigenvalues are close in magnitude.

- **Inverse iteration:** Apply power iteration to A^{-1}; finds smallest $|\lambda|$.
  Each step requires solving a linear system, but can be pre-factorized.

- **Shifted inverse iteration:** Apply iteration to $(A - \sigma I)^{-1}$; converges to eigenvalue closest to σ.

- **Rayleigh quotient iteration:** Update shift each iteration with $\sigma_k = \frac{\mathbf{v}_k^T A \mathbf{v}_k}{\mathbf{v}_k^T \mathbf{v}_k}$.
  Faster convergence, though more expensive per iteration.

7. Eigenvectors

Finding Multiple Eigenvalues

- **Deflation:**
  - After finding $\mathbf{x}_1$ with eigenvalue λ_1, project out its contribution: $\mathbf{v} \leftarrow (I - \mathbf{x}_1 \mathbf{x}_1^T)\mathbf{v}$. Run power iteration again to find the next eigenvector.
  - Can be implemented via Householder reflections to robustly preserve symmetry.
- **QR iterations:**
  - Factor $A_k = Q_k R_k$, set $A_{k+1} = R_k Q_k$.
  - Sequence A_k converges to an upper-triangular matrix with eigenvalues on the diagonal.
  - Most efficient and robust method for all eigenvalues (after reducing to tridiagonal form).

8. Singular Value Decomposition

Core Concepts

- **Definition:** For any $A \in \mathbb{R}^{m \times n}$, $A = U\Sigma V^T$, where
  - $U \in \mathbb{R}^{m \times m}$: orthogonal matrix ($U^T U = I$),
  - $V \in \mathbb{R}^{n \times n}$: orthogonal matrix,
  - $\Sigma \in \mathbb{R}^{m \times n}$: diagonal with nonnegative entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.
- **Geometric interpretation:** A maps the unit sphere to an ellipsoid: $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$, scaling and rotating directions.
- **Relation to eigenproblems:** $A^T A = V\Sigma^T \Sigma V^T, \quad AA^T = U\Sigma\Sigma^T U^T$.
  Singular values are square roots of eigenvalues of $A^T A$ or AA^T.
- SVD always exists and is numerically stable.

8. Singular Value Decomposition

Applications: Pseudoinverse and Low-Rank Approximation

- **Solving linear systems via SVD:**
  - For full-rank A: $\mathbf{x} = V\Sigma^{-1}U^T\mathbf{b}$.
  - For rank-deficient or rectangular A: use the Moore–Penrose pseudoinverse $A^+ = V\Sigma^+U^T$, where Σ^+ inverts only nonzero singular values.
- **Low-rank approximation:** $A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, capturing the dominant structure of A.
- **Eckart–Young theorem:**

$$A_k = \arg \min_{\text{rank}(B)=k} \|A - B\|_2 = \sigma_{k+1}.$$

  Truncated SVD provides optimal compression under both 2-norm and Frobenius norm.

8. Singular Value Decomposition

Matrix Norms and PCA

- **Matrix norms from SVD:**
  - Frobenius norm: $\|A\|_F^2 = \sum_i \sigma_i^2$.
  - Spectral norm: $\|A\|_2 = \sigma_{\max}$.
  - Condition number: $\kappa(A) = \sigma_{\max}/\sigma_{\min}$.
- **Principal Component Analysis (PCA):**
  - Given data matrix $X \in \mathbb{R}^{n \times k}$ (mean-centered) and subspace spanned by $C \in \mathbb{R}^{n \times d}$.
  - Minimizes reconstruction error (maximizes variance) of projected data: $\min_{C^T C = I} \|X - CC^T X\|_F^2$.
  - Given the SVD: $X = U\Sigma V^T$, principal components = columns of U.
- SVD underlies modern data compression, noise reduction, and face recognition (eigenfaces).

9. Nonlinear Systems

Root-Finding in One Dimension

- **Goal:** Find x^* such that $f(x^*) = 0$.
- **Bisection method:** Find the root in $[a, b]$ with $f(a)f(b) < 0$; Repeatedly bisect interval and select subinterval where sign change occurs.
  - Guaranteed convergence but only linear rate.
- **Fixed-point iteration:** Reformulate as $x = g(x)$. Iterate $x_{k+1} = g(x_k)$.
  - Converges if $|g'(x^*)| < 1$; smaller $|g'(x^*)|$ leads to faster convergence.
- **Newton's method:** iterate $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.
  - Quadratic convergence near root if f' smooth and x_0 is close.
  - May diverge for poor initial guesses or discontinuous derivatives.

9. Nonlinear Systems

More Root-Finding Methods

- **Secant method:**
  - Uses finite-difference slope instead of $f'(x)$ and iterate $x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$.
  - Converges superlinearly if x_0 sufficiently close, requires no derivative evaluation.
- **Hybrid methods:**
  - Combine robustness of bisection with speed of Newton/Secant.
  - Example: Brent's method — guaranteed convergence, often near-quadratic.
- **Summary:**
  - Many schemes exist. Faster methods reduce iterations but typically increase per-step cost.
  - Choice depends on smoothness and reliability needs.

9. Nonlinear Systems

Multivariable Systems

- **Problem:** Find $\mathbf{x}^* \in \mathbb{R}^n$ such that $F(\mathbf{x}^*) = \mathbf{0}, \quad F : \mathbb{R}^n \to \mathbb{R}^n$.
- **Newton's method:** $J_F(\mathbf{x}_k)\Delta\mathbf{x}_k = -F(\mathbf{x}_k), \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$. J_F: Jacobian of F.
  - Quadratic convergence near solution if J_F is nonsingular and $\mathbf{x}_0$ is close.
- **Quasi-Newton (Broyden) methods:** Approximate the Jacobian update using rank-one correction:

$$B_{k+1} = B_k + \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}_k}, \quad \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k,\ \mathbf{y}_k = F(\mathbf{x}_{k+1}) - F(\mathbf{x}_k).$$

  - Lower cost per iteration, often superlinear convergence.
- **Conditioning:** $\|(J_F(\mathbf{x}^*))^{-1}\|$; Ill-conditioned J_F leads to unstable updates.

10. Unconstrained Optimization I

Motivation and Problem Setup

- **Goal:** Minimize or maximize $f : \mathbb{R}^n \to \mathbb{R}$ without constraints.
- **Examples:**
  - *Nonlinear least squares:* fit nonlinear model $f(x) = ce^{ax}$ to data (x_i, y_i):

$$E(a, c) = \sum_i (y_i - ce^{ax_i})^2.$$

  - *Maximum likelihood estimation:* fit parameters (μ, σ) for Gaussian $g(h; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(h-\mu)^2/2\sigma^2}$, maximizing the log-likelihood

$$\ell(\mu, \sigma) = \sum_i \log g(h_i; \mu, \sigma).$$

  - *Simulating elastic objects:* find equilibrium minimizing sum of elastic and gravitational energies.

10. Unconstrained Optimization I

Optimality Conditions

- **Global vs. local minima:**
  - Global minimum: $f(x^*) \leq f(x)$ for all x.
  - Local minimum: $f(x^*) \leq f(x)$ within small neighborhood.
- **First-order condition:** $\nabla f(x^*) = 0$. A necessary condition (no descent direction).
- **Second-order test:** $H_f(x^*) = \nabla^2 f(x^*)$.
  - $H_f(x^*) \succ 0$: local minimum.
  - $H_f(x^*) \prec 0$: local maximum.
  - $H_f(x^*)$ indefinite: saddle point.
  - $H_f(x^*)$ singular: degenerate behaviors possible.
- **Convexity:** $f((1-\alpha)x + \alpha y) \leq (1-\alpha)f(x) + \alpha f(y),\ \forall \alpha \in [0,1]$.
  - If f convex, any local minimum is global.

10. Unconstrained Optimization I

One-Dimensional Strategies

- **Newton's method:** $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$. Quadratic convergence near minima when $f''(x^*) > 0$.
- **Secant and parabolic interpolation:** Avoids computing $f''(x)$ by approximating derivatives from previous iterates.
  - Converges superlinearly.
- **Golden-section search:**
  - For unimodal f, recursively narrow search interval.
  - Evaluates at fixed fraction points determined by golden ratio: $\alpha = \frac{3-\sqrt{5}}{2}, \quad 1-\alpha = \frac{\sqrt{5}-1}{2}$.
  - Linear convergence, guaranteed for continuous f.