

Numerical Methods

28. November 2020

Teil I Computing with Matrices

1 Computational Effort

We use *Asymptotic Computation Complexity* (O-bounds) to judge our algorithms. But note that in many cases this approach is imprecise because on today's computers, *memory* bandwidth and latency is a key bottleneck.

Lem. (Matrix Product) The matrix product of an $m \times n$ and $n \times k$ matrix has complexity $O(mnk)$

Tricks to reduce complexity:

- Exploit associativity of operations
- Exploit hidden summations

2 Machine Arithmetic

Machine Numbers have a finite precision, hence any involved computation suffers from **Roundoff**.

Def. (Cancellation) Subtraction of almost equal numbers and accordingly an extreme *amplification of relative errors*.

Tricks to avoid cancellation:

- Trigonometric Identities
- Case Distinction
- Taylor Approximations
- Computing Diff. Quot. through Approx.

Def. (Stability) An Algorithm F is **numerically stable** if for all $x \in X$ its result $F(x)$ (possibly affected by roundoff) is the exact result for 'slightly perturbed' data.

Teil II Linear Systems

1 Theory

Def. The Condition Number of a matrix is

$$\text{cond}(\mathbf{A}) := \|\mathbf{A}\| \|\mathbf{A}^{-1}\|$$

and is a measure for stability (≈ 1 for stability). Note that it is infinite for singular matrices. Intuitively $\text{cond}(\mathbf{A})$ catches the maximum stretching factor divided by the minimum stretching factor.

Lem. If $m = n$ and \mathbf{A} is regular/invertible, then its 2-norm condition number is $\text{cond}_2(\mathbf{A}) = \sigma_1/\sigma_n$

2 LU Decomposition

There are two methods to choose pivots in the elimination process with a Permutation Matrix:

- Partial Pivoting: Max of column
- Full Pivoting: Max of remaining matrix

Complexities:

- Gauss elim. $\mathcal{O}(n^3)$, back substitution $\mathcal{O}(n^2)$
- LU: decomp. $\mathcal{O}(n^3)$, solve $\mathcal{O}(n^2)$
- Inverse: compute $\mathcal{O}(n^3)$, solve $\mathcal{O}(n^2)$

3 Exploiting Structure when Solving Linear Systems

Method: (Low Rank Modification) Use the *Sherman-Morrison-Woodbury* Formula to compute the inverse of slightly modified (rank-1-modification) matrix.

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^H)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^H\mathbf{A}^{-1}$$

4 Matrix Storage Formats

Dense Matrices are either stored in Column-Major Ordering or Row-Major Ordering. By default Eigen employs Column-Major Ordering.

Sparse Matrices can be stored in

- COO/Triplet: Vector of triplets of (row, col, value)
- CCS: Compressed Column Storage
- CRS: Compressed Row Storage

where CRS consists of the arrays:

- val: All non-zero values in Row-Major Order

- col_idx: Column index of each entry in val

- row_ptr: Pointers to first elements of row

Note that $\text{size}(\text{row_ptr}) = m + 1$ (Sentinel).

Teil III Linear Least Squares

Goal: Generalisation of solving $\mathbf{Ax} = \mathbf{b}$, find

$$\mathbf{x} \in \underset{\mathbf{y} \in \mathbb{R}^n}{\text{argmin}} \|\mathbf{Ay} - \mathbf{b}\|_2^2$$

Critical comparison of Methods:

- Normal Equations: $\mathcal{O}(n^2m + n^3)$
→ blows up instability
- Extended Normal Equations:
→ same conditioning as \mathbf{A}
→ sparsity preserved
- Householder QR: $\mathcal{O}(n^2m)$
→ always stable
→ loss of sparsity
- SVD:
→ no full rank requirement for \mathbf{A}

1 Normal Equation Methods

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$$

Warning: Normal Equations are vulnerable to instability since $\text{cond}_2(\mathbf{A}^T \mathbf{A}) = \text{cond}_2(\mathbf{A})^2$

2 Singular Value Decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^H, \quad \mathbf{A} \in \mathbb{K}^{m \times n},$$

$p := \min\{m, n\}$, $r := \text{rank}(\mathbf{A})$,
 $\mathbf{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_p)$ $\sigma_1 > \sigma_2 > \dots > \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0$

Full:

- $\mathbf{U} \in \mathbb{K}^{m \times m}$ $[\mathcal{R}(\mathbf{A}) | \mathcal{N}(\mathbf{A})]$ (unitary)
- $\mathbf{\Sigma} \in \mathbb{K}^{m \times n}$ (generalized diagonal)
- $\mathbf{V} \in \mathbb{K}^{n \times n}$ $[\mathcal{R}(\mathbf{A}^H) | \mathcal{N}(\mathbf{A}^H)]$ (unitary)

Economical:

- $\mathbf{U} \in \mathbb{K}^{m \times p}$ $[\mathcal{R}(\mathbf{A})]$ (orthogonal columns)
- $\mathbf{\Sigma} \in \mathbb{K}^{p \times p}$ (diagonal)
- $\mathbf{V} \in \mathbb{K}^{n \times p}$ $[\mathcal{R}(\mathbf{A}^H)]$ (orthogonal columns)

Numerical Rank $r := \max_{j \in \{1, \dots, p\}} \left(\frac{\sigma_j}{\sigma_1} \geq \text{TOL} \right)$
Cost of Eco SVD $\mathcal{O}(\min\{m, n\}^2 \max\{m, n\})$

Lem. (LSQ by SVD)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^n}{\text{argmin}} \|\mathbf{Ax} - \mathbf{b}\|_2 = \mathbf{V}_1 \mathbf{\Sigma}_r^{-1} \mathbf{U}_1^T \mathbf{b}$$

If the lsq has multiple solutions, \mathbf{x}^* has minimal norm.

Thm. (Low Rank Approximation) Let \mathbf{A}_k be the matrix resulting from only keeping the first k singular values of \mathbf{A} . \mathbf{A}_k is the best rank- k approximation of \mathbf{A} .

Method: (Principal Component Analysis)

Given a dataset in the form of a matrix $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$, such that every column represents a time series.

- Columns of \mathbf{U} capture dominant trends.
- Columns of \mathbf{V} capture how strong the trends are in a particular column of \mathbf{A}
- Diagonal Entries of $\mathbf{\Sigma}$ tell us how dominant the trend is overall.

Method: (Proper orthogonal decomp.)

Closely related is the POD problem, where we seek k -dimensional subspace U_k of \mathbb{R}^m , for which the sum of squared distances of the data points $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ to U_k is minimal.

Using the *Low Rank Approximation* Theorem we can prove that

$$U_k = \mathcal{R}(\mathbf{U}_{:,1:k})$$

3 QR Decomposition

$$\mathbf{A} = \mathbf{QR}, \quad \mathbf{A} \in \mathbb{K}^{m \times n} \quad \mathbf{Q} \in \mathbb{K}^{m \times m}$$

Idea: LLQ Problems are easier to solve for triangular matrices, hence we manage to decompose \mathbf{A} into \mathbf{QR} we have such a System since orthogonal matrices are norm-invariant.

Method: (Householder Reflections) The following Householder matrix performs a reflection

$$\mathbf{H}(\mathbf{v}) := \mathbf{I} - 2\mathbf{v}\mathbf{v}^T$$

at the hyperplane with normal unit vector \mathbf{v} (Intuitively $\mathbf{H}(\mathbf{v})$ subtracts the projection of \mathbf{x} on \mathbf{v} twice).

We can reduce a matrix \mathbf{A} to \mathbf{R} using n successive transformations

$$\mathbf{H}(\mathbf{v}_n) \dots \mathbf{H}(\mathbf{v}_1) \mathbf{A} = \mathbf{R}$$

, where $\mathbf{H}(\mathbf{v}_i)$ reflects the lower part of the i 'th column of the current \mathbf{A} on \mathbf{e}_i . E.g

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

Method: (Givens Rotations) We can selectively eliminate entries with Givens rotations. The following matrix rotates everything in the hyper-plane defined by $\mathbf{e}_1, \mathbf{e}_k$:

$$\mathbf{G}(1, k, \theta) = \begin{bmatrix} c & \cdots & s & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{bmatrix}$$

Note that $c = \cos(\theta), s = \sin(\theta)$. Two eliminate a_k we solve the equations.

$$c^2 + s^2 = 1 \quad \text{and} \quad -sa_1 + ca_k = 0$$

As with householder reflections, there are always two possibilities, one of whom is cancellation free.

Lem. (Modifications Techniques) Computing the QR decomposition of a slightly modified matrix (rank-1-modification, adding a row, adding a column) can be done efficiently in $\mathcal{O}(mn + n^2)$.

4 Constrained Least Squares

Problem: Find $x \in \mathbb{R}^n$ such that

$$\|\mathbf{Ax} - \mathbf{b}\| \rightarrow \min \quad \text{and} \quad \mathbf{Cx} = \mathbf{d}.$$

Method: (Lagrangian Multipliers) We introduce the multiplier $\mathbf{m} \in \mathbb{R}^p$ to solve

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \sup_{\mathbf{m} \in \mathbb{R}^p} \|\mathbf{Ax} - \mathbf{b}\|_2 + \mathbf{m}^T (\mathbf{Cx} - \mathbf{d})$$

and we notice that for any finite solution $(\mathbf{Cx} - \mathbf{d}) = 0$. By realising that that for the solution all partial derivatives must be zero we can obtain the *augmented normal equations*.

Method: (By SVD) We have

$$\mathbf{x} \in \mathbf{x}_0 + \mathcal{N}(\mathbf{C}) \quad \mathbf{x}_0 = \text{particular solution}$$

Hence since the SVD gives a basis of $\mathcal{N}(\mathbf{C})$ we write

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_2 \mathbf{y}$$

for some \mathbf{y} which leads to the standard lsq:

$$\|\mathbf{AV}_2 \mathbf{y} - (\mathbf{b} - \mathbf{Ax}_0)\| \rightarrow \min$$

Teil IV

Filtering Algorithms

1 Filters & Convolution

Def. (Filter) A function $F : l^\infty(\mathbb{Z}) \rightarrow l^\infty(\mathbb{Z})$ where $l^\infty(\mathbb{Z})$ is the space of bounded infinite sequences

$$l^\infty(\mathbb{Z}) = \left\{ (x_j)_{j \in \mathbb{Z}} : \sup |x_j| < \infty \right\}$$

Def. (LT-FIR) A Filter that is:

- Linear
- Time-Invariant: Shifting the input in time leads to the same output shifted in time.
- Finite: $\exists M \forall j : x_j = 0 \text{ if } |j| > M \implies \exists N \forall j : y_j = 0 \text{ if } |j| > N$
- Causal: The output does not start before the input.

Such a filter is uniquely characterized by its **Impulse response**:

$$F(\delta_{j,0}) = \dots, 0, h_0, h_1, \dots, h_{n-1}, 0, \dots$$

For inputs $\mathbf{x} \in \mathbb{R}^m$ we get outputs $\mathbf{y} \in \mathbb{R}^{m+n-1}$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0 \\ h_1 & h_0 & 0 & 0 \\ h_2 & h_1 & h_0 & 0 \\ h_3 & h_2 & h_1 & h_0 \\ 0 & h_3 & h_2 & h_1 \\ 0 & 0 & h_3 & h_2 \\ 0 & 0 & 0 & h_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_4 \end{bmatrix}$$

Def. (Discrete Convolution) A Filter where given $\mathbf{x} = [x_0, \dots, x_{m-1}]^T \in \mathbb{K}^m, \mathbf{h} = [h_0, \dots, h_{n-1}]^T \in \mathbb{K}^n$ their DCONV is the vector $\mathbf{y} \in \mathbb{K}^{m+n-1}$ with components.

$$y_k = \sum_{j=0}^{m-1} h_{k-j} x_j$$

Def. (Periodic Convolution) Given two n -periodic signals $(u_k)_{k \in \mathbb{Z}}, (x_k)_{k \in \mathbb{Z}}$ PCONV yields the n -periodic signal:

$$(y_k) := (u_k) *_{\mathbf{n}} (x_k), y_k := \sum_{j=0}^{n-1} u_{k-j} x_j$$

PCONV can be represented by a matrix

$$\mathbf{C} = [c_{ij}]_{i,j=1}^n, c_{ij} = u_{j-i}$$

. Such a matrix is called **circulant**. The vector \mathbf{u} s.t. $\mathbf{C} = \text{circ}(\mathbf{u})$ corresponds to $\mathbf{C}_{:,1}$.

2 Discrete Fourier Transform

The Fourier Matrix for inputs $\mathbf{y} \in \mathbb{C}^n$ is given by

$$\mathbf{F}_n = \left[\omega_n^{lj} \right]_{l,j=0}^{n-1} \in \mathbb{C}^{n,n} \quad \omega_n = \exp\left(\frac{-2\pi i}{n}\right)$$

and $\text{DFT}_n(\mathbf{y}) := \mathbf{F}_n \mathbf{y} = \left[\sum_{j=0}^{n-1} y_j \omega_n^{kj} \right]_{k=0}^{n-1}$.

Properties include

$$\mathbf{F}_n^{-1} = \frac{1}{n} \mathbf{F}_n^H = \frac{1}{n} \overline{\mathbf{F}_n}$$

Method: (Frequency Filtering) The columns of \mathbf{F}_n are trigonometric basis vectors, where

$$\mathbf{v}_k = \left[\cos\left(j \frac{2\pi}{n} \cdot k\right) + i \sin\left(\frac{2\pi j k}{n}\right) \right]_{j=0}^{n-1}$$

k is the 'frequency' and $j2\pi/n$ is the sample point. Hence the DFT_n is a basis transformation

$$B_E \leftrightarrow B_{\text{trig}}$$

Denosing and low/high filters can be implemented by manipulating the signal in the frequency domain.

Lem. (Diagonalizing circulant matrices)

For any circulant matrix $\mathbf{C} \in \mathbb{C}^{n,n}$ we have

$$\mathbf{C} = \mathbf{F}_n^{-1} \text{diag}(d_1, \dots, d_n) \mathbf{F}_n,$$

$$[d_0, \dots, d_{n-1}]^T = \mathbf{F}_n [u_0, \dots, u_{n-1}]^T$$

In other words, the columns of \mathbf{F}_n are eigenvectors of \mathbf{C} .

Thm. (Convolution Theorem) Periodic convolution in the time-domain equals to multiplication in the frequency-domain.

$$(\mathbf{u}) *_{\mathbf{n}} (\mathbf{x}) = \mathbf{F}_n^{-1} \left[(\mathbf{F}_n \mathbf{u})_j (\mathbf{F}_n \mathbf{x})_j \right]_{j=1}^n$$

Implementation Speedup: $\mathcal{O}(n^2) \rightarrow \mathcal{O}(n \log_2(n))$

Thm. (2D DFT) Is given by

$$\mathbf{C} = \mathbf{F}_m \left(\mathbf{F}_n \mathbf{Y}^T \right)^T = \mathbf{F}_m \mathbf{Y} \mathbf{F}_n$$

The real basis vectors can be visualized. Analogously there is a 2D-Convolution-Theorem:

$$\mathbf{X} *_{m,n} \mathbf{Y} = \text{DFT}_{m,n}^{-1} (\text{DFT}_{m,n}(\mathbf{X}) \odot \text{DFT}_{m,n}(\mathbf{Y}))$$



Method: (Deblurring an image) A blurred image can be modeled as the 2D-Convolution of the actual image with a small filter. In the

2D-frequency-domain deblurring equals component wise division.

3 Fast Fourier Transform

The DFT of a vector of length pq can be divided in p DFT's of vectors of length q . The asymptotic runtime for this special 'divide and conquer' is

$$\text{FFT}(n) \in \mathcal{O}(n \log n)$$

Special Case $p = 2$:

$$\text{DFT}(\mathbf{v}) = \text{DFT}(\mathbf{v}_{\text{even}}) + s \text{DFT}(\mathbf{v}_{\text{odd}})$$

Def. (Toeplitz Matrix) $\mathbf{T} = (t_{ij})_{i,j=1}^n \in \mathbb{K}^{mn}$ is a Toeplitz matrix if it has constant diagonals given by a vector $\mathbf{u} \in \mathbb{K}^{m+n-1}$. Note that every circulant matrix is also a Toeplitz matrix.

Lem. (Fast Arithmetic for Toeplitz)

Vector Multiplication in $\mathcal{O}((n+m) \log(n+m))$: Construct Circulant Matrix $\mathbf{C} \in \mathbb{K}^{m+n, m+n}$ s.t.

$$\begin{bmatrix} \mathbf{Tx} \\ \star \end{bmatrix} = \mathbf{C} \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix}$$

which can be solved in the frequency domain (Convolution Theorem).

LSE Solution in $\mathcal{O}(n^2)$: The Levinson Algorithm.

Teil V

Data Interpolation in 1D

Goal: Given data points (t_i, y_i) and a finite-dimensional vector-space of functions V , reconstruct $f \in V$ such that $\forall i \quad f(t_i) = y_i$.

Def. (Interpol. as linear mapping)

For data points $(t_i, y_i) \quad i = 0, \dots, n$ and V spanned by basis functions b_0, \dots, b_m we have

$$\mathbf{A} \mathbf{c} := \begin{bmatrix} b_0(t_0) & \dots & b_m(t_0) \\ \vdots & & \vdots \\ b_0(t_n) & \dots & b_m(t_n) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_m \end{bmatrix} = \mathbf{y}$$

→ The interpolant is uniquely solv. iff $m = n$

→ The basis is **cardinal** if $b_j(t_i) = \delta_{ij}$

1 Global Polynomial Interpol.

Def. (Monomial Basis)

→ Evaluation in $\mathcal{O}(n)$ by exploiting associativity.

Def. (Lagrange Basis)

$$L_i(t) := \prod_{\substack{j=0 \\ j \neq i}}^n \frac{t - t_j}{t_i - t_j} = I_{\mathcal{T}}(\mathbf{e}_{i+1})$$

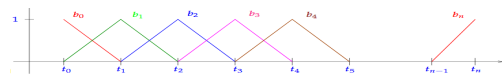
→ Cardinal Basis.

2 Shape-Preserving Interpol.

Method: (Piecewise linear Interpol.)

→ locally shape preserving

Tent functions make a cardinal basis:



3 Splines

4 Trigonometric Interpol.

5 Least Squares Fitting

- lagrange polynomial interpolation problem (scheme?)

- \hookrightarrow efficient interpolation at many points $\mathcal{O}(n^2 + N * n)$

- \hookrightarrow Aitken-Neville for single point evaluation in $\mathcal{O}(n^2)$, useful for simultaneous updates, update in $\mathcal{O}(k)$, eval in $\mathcal{O}(1)$.

- Extrapolation \hookrightarrow Algorithm to extrapolate to zero
Barycentric Algorithm: - Precomputation: $\mathcal{O}(n^2)$ - eval at single spot in $\mathcal{O}(n)$

Aitken-Neville (SINGLE POINT EVALUATION: - update in $\mathcal{O}(k)$).

Newton Basis: - Precomputation: add node in $\mathcal{O}(n)$

- eval at a single spot in $\mathcal{O}(n)$

- \hookrightarrow Divided Differences algorithm for newton basis (computes in situ, hence better than forward substitution)

sensitivity vs stability \hookrightarrow lebesgue?

Shape-preserving Interpolation \hookrightarrow Cubic Hermite Interpolation: \hookrightarrow Interpolate as piecewise polynomial, where ends match (derivative) Different methods to determine properties of piecewise borders: - Linear (Derivative is weighted mean of adjacent derivatives) \hookrightarrow No Preservation of monotonicity - Limited Slopes (Pchip): Since slope must be zero at boundary in some cases \hookrightarrow Not linear

Splines (see wooden splines!) \hookrightarrow Instead of fixing the boundary slopes, we add additional continuity conditions. We need to add exactly $n+1$ conditions. Cubic Splines \hookrightarrow $n-1$ conditions given by saying SE-COND derivate must be the same. The remaining two conditions can be: natural \hookrightarrow leftmost and rightmost zero periodic \hookrightarrow leftmost and rightmost must match full \hookrightarrow we dictate conditions on leftmost and rightmost border

Note that: - cubic splines are not shape preserving
- cubic splines are weakly local

See 5.4.2.1 for a comparison of methods.