Lecture 13
Stability of LU Factorization; Cholesky Factorization

Songting Luo

Department of Mathematics
Iowa State University

MATH 562 Numerical Analysis II

Outline

1. Stability of LU Factorization

2. Cholesky Factorization

3. Linear Algebra Software

Outline

1. Stability of LU Factorization
2. Cholesky Factorization
3. Linear Algebra Software

Stability of LU without Pivoting

- For $\mathbf{A} = \mathbf{LU}$ computed without pivoting

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{L}\|\|\mathbf{U}\|} = O(\epsilon_{machine})$$

- This is close to backward stability, except that we have $\|\mathbf{L}\|\|\mathbf{U}\|$ instead of $\|\mathbf{A}\|$ in the denominator
- Instability of Gaussian elimination can happen only if one or both of the factors $\mathbf{L}$ and $\mathbf{U}$ is large relative to size of $\mathbf{A}$
- Unfortunately, $\|\mathbf{L}\|$ and $\|\mathbf{U}\|$ can be arbitrarily large (even for well-conditioned $\mathbf{A}$), e.g.,

$$\mathbf{A} = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix} \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{-20} \end{bmatrix}$$

- Therefore, the algorithm is unstable

Stability of LU with Partial Pivoting

- With pivoting, all entries of **L** are in $[-1,\ 1]$, so $\|\mathbf{L}\| = O(1)$.
- To measure growth in **U**, we introduce the growth factor $\rho = \frac{\max_{ij}|u_{ij}|}{\max_{ij}|a_{ij}|}$, and hence $\|\mathbf{U}\| = O(\rho(\mathbf{A}))$
- We then have **PA** = **LU**

$$\tilde{\mathbf{L}}\tilde{\mathbf{U}} = \tilde{\mathbf{P}}\mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\rho\epsilon_{machine})$$

- If $|l_{ij}| < 1$ for each $i > j$, (i.e., there is no tie for the pivoting), then $\tilde{\mathbf{P}} = \mathbf{P}$ for sufficiently small $\epsilon_{machine}$
- If $\rho = O(1)$, then the algorithm is backward stable
- In fact, $\rho \leqslant 2^{m-1}$, so by definition ρ is a constant but can be very large

The Growth Factor

- ρ can indeed be as large as 2^{m-1}. Consider matrix
$$\begin{bmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ -1 & -1 & 1 & & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & & 0 \\ -1 & 1 & & & 0 \\ -1 & -1 & 1 & & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & 1 & 8 \\ & & & & 16 \end{bmatrix}$$
where growth factor $\rho = 16 = 2^{m-1}$.
- $\rho = 2^{m-1}$ is as large as ρ can get. It can be catastrophic in practice
- Theoretically, Gaussian elimination with partial pivoting is backward stable according to formal definition
- However, in the worst case, Gaussian elimination with partial pivoting may be unstable for practical values of m

The Growth Factor in Practice

- Good news: Large ρ occurs only for very skewed matrices. Experimentally, one rarely see very large ρ
- Probability of large ρ decreases exponentially in ρ.
- "If you pick a billion matrices at random, you will almost certainly not find one for which Gaussian elimination is unstable"
- In practice, ρ is no larger than $O(\sqrt{m})$. However, this behavior is not fully understood yet
- In conclusion,
    - Gaussian elimination with partial pivoting is backward stable
    - In theory, its error may grow exponentially in m
    - In practice, it is stable for matrices of practical interests

Outline

1. Stability of LU Factorization
2. Cholesky Factorization
3. Linear Algebra Software

Hermitian Positive-Definite Matrices

- Symmetric matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$ is symmetric positive definite (SPD) if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for $\mathbf{x} \in \mathbb{R}^m \backslash \{\mathbf{0}\}$
- Hermitian matrix $\mathbf{A} \in \mathbb{C}^{m\times m}$ is Hermitian positive definite (HPD) if $\mathbf{x}^*\mathbf{A}\mathbf{x} > 0$ for $\mathbf{x} \in \mathbb{C}^m \backslash \{\mathbf{0}\}$
- If $\mathbf{A}$ is $m\times$ HPD and $\mathbf{X} \in \mathbb{C}^{m\times m}$ has full column rank, then $\mathbf{X}^*\mathbf{A}\mathbf{X}$ is HPD
- Any principal submatrix (picking some rows and corresponding columns) of $\mathbf{A}$ is HPD and $a_{ii} > 0$.
- HPD matrices have positive real eigenvalues and orthogonal eigenvectors
- Note: A positive-definite matrix does not need to be symmetric or Hermitian! A real matrix $\mathbf{A}$ is positive definite iff $\mathbf{A} + \mathbf{A}^T$ is SPD.

Cholesky Factorization

- Key idea: take advantage and preserve the properties of symmetry and positive-definiteness in factorization
- Eliminate below diagonal and to the right of diagonal

$$\mathbf{A} = \begin{bmatrix} a_{11} & \mathbf{w}^* \\ \mathbf{w} & \mathbf{K} \end{bmatrix} = \begin{bmatrix} \alpha & \mathbf{0} \\ \mathbf{w}/\alpha & \mathbf{I} \end{bmatrix} \begin{bmatrix} \alpha & \mathbf{w}^*/\alpha \\ \mathbf{0} & \mathbf{K} - \mathbf{w}\mathbf{w}^*/a_{11} \end{bmatrix}$$

$$= \begin{bmatrix} \alpha & \mathbf{0} \\ \mathbf{w}/\alpha & \mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K} - \mathbf{w}\mathbf{w}^*/a_{11} \end{bmatrix} \begin{bmatrix} \alpha & \mathbf{w}^*/\alpha \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \mathbf{R}_1^* \mathbf{A}_1 \mathbf{R}_1$$

where $\alpha = \sqrt{a_{11}}$, $a_{11} > 0$

- $\mathbf{K} - \mathbf{w}\mathbf{w}^*/a_{11}$ is principal submatrix of HPD $\mathbf{A}_1 = \mathbf{R}_1^{-*}\mathbf{A}\mathbf{R}_1^{-1}$ and therefore is HPD, with positive diagonal entries

Cholesky Factorization

- Apply recursively to obtain

$$\mathbf{A} = (\mathbf{R}_1^*\mathbf{R}_2^* \cdots \mathbf{R}_m^*)(\mathbf{R}_m \cdots \mathbf{R}_2\mathbf{R}_1) = \mathbf{R}^*\mathbf{R}, \quad r_{jj} > 0$$

  which is known as Cholesky factorization
- Question: Is $\mathbf{R}$ simply ?union? of kth rows of $\mathbf{R}_k$ (or $\mathbf{R}^*$ "union" of kth columns of $\mathbf{R}_k^*$)? Yes. Hint: Write $\mathbf{R}_k^*$ in a form similar to $\mathbf{L}_k = \mathbf{I} + \mathbf{l}_k\mathbf{e}_k^*$ in LU.
- Existence and uniqueness: every HPD matrix has a unique Cholesky factorization
  - Exists because algorithm for Cholesky factorization always works for HPD matrices
  - Is unique since once $\alpha = \sqrt{a_1 1}$ is determined at each step, entire column $\mathbf{w}/\alpha$ is determined
  - Question: How to check whether a Hermitian matrix is positive definite? Answer: Run Cholesky factorization and it would succeeds iff the matrix is positive definite.

Algorithm of Cholesky Factorization

- Factorize HPD matrix $\mathbf{A}$

Algorithm: Cholesky factorization

$\mathbf{R} = \mathbf{A}$
for $k = 1$ to m
  for $j = k+1$ to m
    $\mathbf{r}_{j,j:m} \leftarrow \mathbf{r}_{j,j:m} - \mathbf{r}_{k,j:m}\bar{r}_{kj}/r_{kk}$
  $\mathbf{r}_{k,k:m} \leftarrow \mathbf{r}_{k,k:m}/\sqrt{r_{kk}}$

- Operation count

$$\sum_{k=1}^{m}\sum_{j=k+1}^{m} 2(m-j) \sim 2\sum_{k=1}^{m}\sum_{j=1}^{k} j \sim \sum_{k=1}^{m} k^2 \sim m^3/3$$

Stability

Theorem

The computed Cholesky factor $\tilde{\mathbf{R}}$ satisfies

$$\tilde{\mathbf{R}}^*\tilde{\mathbf{R}} = \mathbf{A} + \delta\mathbf{A}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\epsilon_{machine}),$$

i.e., Cholesky factorization is backward stable

- Forward errors in $\tilde{\mathbf{R}}$ is $\|\tilde{\mathbf{R}} - \mathbf{R}\|/\|\mathbf{R}\| = O(\kappa(\mathbf{A})\epsilon_{machine})$, which may be large for ill-conditioned $\mathbf{A}$.
- Solve $\mathbf{Ax} = \mathbf{b}$ for positive definite $\mathbf{A}$
  - Factorize $\mathbf{A} = \mathbf{R}^*\mathbf{R}$, solve $\mathbf{R}^*\mathbf{y} = \mathbf{b}$, solve $\mathbf{Rx} = \mathbf{y}$
  - Operation count is $\sim m^3/3$
  - Algorithm is backward stable:

$$(\mathbf{A} + \Delta\mathbf{A})\tilde{\mathbf{x}} = \mathbf{b}, \quad \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} = O(\epsilon_{machine})$$

$\mathbf{LDL}^*$ Factorization

- Cholesky factorization is sometimes given by $\mathbf{A} = \mathbf{LDL}^*$ where $\mathbf{D}$ is diagonal matrix and $\mathbf{L}$ is unit lower triangular matrix
- It avoids computing square roots
- Analogously, LU factorization can also be written as $\mathbf{LDU}$, where $\mathbf{U}$ is unit upper triangular
- Question: How is $\mathbf{R}$ in $\mathbf{A} = \mathbf{R}^*\mathbf{R}$ related to the $\mathbf{L}$ and $\mathbf{U}$ factors of $\mathbf{A} = \mathbf{LU}$?
  - $\mathbf{U} = \mathbf{DL}^* = \sqrt{\mathbf{D}}\mathbf{R}$, where $\sqrt{\mathbf{D}} = diag(\sqrt{d_{11}}, \cdots, \sqrt{d_{mm}})$
- Hermitian indefinite systems can be factorized with $\mathbf{PAP}^T = \mathbf{LDL}^*$, but $\mathbf{D}$ is block diagonal with 1×1 and 2×2 blocks. Its cost is similar to Cholesky factorization and is about 50% of Gaussian elimination.

Outline

1. Stability of LU Factorization

2. Cholesky Factorization

3. Linear Algebra Software

Software for Linear Algebra

- **LAPACK**: **L**inear **A**lgebra **PACK**age (`www.netlib.org`)
    - Standard library for solving linear systems and eigenvalue problems
    - Depends on **BLAS** (Basic Linear Algebra Subprograms)
    - Note: Uses Fortran conventions for matrix arrangements
    - C-version: **CLAPACK**
- **MATLAB**
    - e.g., Factorize **A**: $lu(A)$ and $chol(A)$
    - Solve **Ax** $=$ **b**: $x = A\backslash b$
        - Uses back/forward substitution for triangular matrices
        - Uses Cholesky factorization for positive-definite matrices
        - Uses LU factorization with column pivoting for nonsymmetric matrices
        - Uses Householder QR for least squares problems
    - Uses LAPACK and other packages internally
- Solvers for sparse matrices (e.g., SuperLU, TAUCS)

Some Examples

Example BLAS routines: Matrix-vector multip.: dgemv; Matrix-matrix multip: dgemm

	LU Factorization		Solve linear system		Est. cond
	General	Symmetric	General	Symmetric	
LAPACK	dgetrf	dpotrf/dsytrf	dgesv	dposv/dposvx	dgecon
LINPACK	dgefa	dpofa/dsifa	dgesl	dposl/dsisl	dgeco
MATLAB	lu	chol			rcond

	Linear least squares			Eigenvalue/vector		SVD
	QR	Solve	Rank-deficient	General	Sym.	
LAPACK	dgeqrf	dgesl	dgelsy/s/d	dgeev	dsyev	dgesvd
LINPACK	dqrdc	dqrsl	dqrst			dsvdc
MATLAB	qr			eig	eig	svd

For BLAS, LINPACK, and LAPACK, first letter **s** stands for single-precision real, **d** for double-precision real, **c** for single-precision complex, and **z** for double-precision complex.

Using LAPACK Routines in C Programs

- LAPACK was written in Fortran 77. Special attention is required when calling a Fortran subroutine from C.
- Key differences between C and Fortran
  - Storage of matrices: column major (Fortran) versus row major (C/C++)
  - Argument passing for subroutines in C and Fortran: pass by reference (Fortran) and pass by value (C/C++)
- To find a function name, refer to LAPACK Users? Guide. or search `netlib.org`
- To find out arguments for a given function, search on `netlib.org`
- Note the difference on precision