Outline

1 Gaussian Elimination
   • Last Time...
   • Stability
   • Backward Stability? Practical Stability?

2 Cholesky Factorization
   • Hermitian Positive Definite Matrices
   • $R^*R$-factorization

3 Reference
Rewind: Last Time

We quickly reviewed a familiar algorithm — **Gaussian Elimination**.

If we save the multipliers generated by the elimination, we get the **LU-factorization** of $A$, *i.e.*, $A = LU$, where $L$ is lower triangular, and $U$ is upper triangular.

In this initial form, GE/LU is completely useless (unstable), we discussed a couple of fixes, some probably familiar, some new...
In **Partial Pivoting** we rearrange the rows of the matrix $A$ (on the fly) in order to move the largest element in the “active” column to the diagonal entry — this way we can guarantee that the multiplier is bounded by one

$$\tilde{l}_{ji} = a_{ji} \odot a_{ii} = \frac{a_{ji}}{a_{ii}}(1 + \epsilon), \quad |\epsilon| \leq \epsilon_{\text{mach}}, \quad |\delta_{\tilde{l}_{ji}}| \leq \epsilon_{\text{mach}} \ell_{ji}$$

We get $PA = LU$
Rewind: Last Time

**Partial Pivoting** is stable “most of the time.” We looked at enhancements taking scale into consideration: **Scaled Partial Pivoting**.

The overall work for GE/LU is $\sim \frac{2m^3}{3}$, and partial pivoting adds $\mathcal{O}(m^2)$ operations, which is a small cost.

Sometimes **Complete Pivoting** — rearrangement of both the rows and columns of $A$ is necessary to achieve high accuracy. The cost is significant since the additional work adds $\mathcal{O}(m^3)$ operations.

We get $PAQ = LU$
Now...

— We look at the stability of Gaussian elimination.

— Gaussian Elimination for *Hermitian Positive Definite Matrices*:

  — Cholesky Factorization — The Hermitian (Symmetric) version of LU-factorization.
“Gaussian Elimination with partial pivoting is explosively unstable for certain matrices, yet stable in practice. This apparent paradox has a statistical explanation.”

[Trefethen-&-Bau, p.163]

The stability analysis of Gaussian Elimination with Partial Pivoting (GE w/PP) is complicated, consider the example $A = LU$

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

The likely naively computed $\tilde{L}$ and $\tilde{U}$ are

$$
\begin{bmatrix}
1 & 0 \\
10^{20} & 1
\end{bmatrix}
\begin{bmatrix}
10^{-20} & 1 \\
0 & -10^{20}
\end{bmatrix}
= 
\begin{bmatrix}
10^{-20} & 1 \\
1 & 0
\end{bmatrix} \neq A
$$
This behavior is quite generic — instability in Gaussian Elimination (with or without pivoting) can arise if the factors $\tilde{L}$ or $\tilde{U}$ are large compared with $A$.

In the previous example we have

$$\|A\|_F = 1.7321, \quad \|\tilde{L}\|_F = 1.0000 \times 10^{20}, \quad \|\tilde{U}\|_F = 1.0000 \times 10^{20}$$

i.e. the computed factors are 20 orders of magnitude larger than the initial matrix — no wonder we run into problems!

The purpose of pivoting — from the point of view of stability/accuracy — is to make sure that $\tilde{L}$ and $\tilde{U}$ are not too large.
Formal Result

Theorem (LU-Factorization without (explicit) Pivoting)

Let the factorization \( A = LU \) of a non-singular matrix \( A \in \mathbb{C}^{m \times m} \) be computed by Gaussian Elimination without pivoting in a floating point environment satisfying the floating point axioms. If \( A \) has an LU-factorization, then for \( \varepsilon_{mach} \) small enough, the factorization completes successfully in floating point arithmetic (no zero pivots \( \tilde{a}_{ii} \) are encountered), and the computed matrices \( \tilde{L} \) and \( \tilde{U} \) satisfy

\[
\tilde{L}\tilde{U} = A + \delta A, \quad \frac{\|\delta A\|}{\|L\|\|U\|} = \mathcal{O}(\varepsilon_{mach})
\]

for some \( \delta A \in \mathbb{C}^{m \times m} \).

Note that we can make the theorem apply to GE w/Pivoting by applying it to the “pre-pivoted matrix:” \( A := PA[Q] \).
Formal Result: Comments

If we just flash by the previous slide, the result looks just like all the other backward stability results... **BUT!!** take a closer look... we have

$$\frac{\|\delta A\|}{\|L\| \|U\|} = O(\varepsilon_{\text{mach}}).$$

Usually, the results contain something like

$$\frac{\|\delta A\|}{\|A\|} = O(\varepsilon_{\text{mach}}).$$

There is a **critical difference** here. If $\|L\| \|U\| = O(\|A\|)$, then the theorem states that GE is backward stable. However (like in our previous example), if $\|L\| \|U\| \gg O(\|A\|)$, all bets are off!
Quantifying Stability

Without pivoting, both $\|L\|$ and $\|U\|$ can be unbounded, and GE w/o Pivoting is unstable by any standard.

Consider GE w/PP. By construction $|\ell_{ij}| \leq 1$, so that $\|L\| = O(1)$ in any norm (this is true for all the pivoting schemes we have discussed). We now focus our attention to $U$; essentially GE w/PP is backward stable provided $\|U\| = O(\|A\|)$.

The following quantity turns out to be very useful:

**Definition (Growth Factor)**

The **growth factor** of $A$ (and the algorithm) is defined as the ratio

$$\rho = \frac{\max_{i,j} |u_{ij}|}{\max_{i,j} |a_{ij}|}$$
The Growth Factor... and Stability

If $\rho \sim 1$, there is little growth, and the elimination process is stable. When $\rho$ is large, we expect loss of accuracy and/or instability of the algorithm... We make this precise: —

Theorem

Let the factorization $PA = LU$ of a non-singular matrix $A \in \mathbb{C}^{m \times m}$ be computed by GE w/PP in a floating point environment satisfying the floating point axioms. The computed matrices $\tilde{P}$, $\tilde{L}$, and $\tilde{U}$ satisfy

$$\tilde{L}\tilde{U} = \tilde{P}A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = O(\rho \varepsilon_{\text{mach}})$$

for some $\delta A \in \mathbb{C}^{m \times m}$, where $\rho$ is the growth factor of $A$. If $|\ell_{ij}| < 1$ for $i > j$, then $P = \tilde{P}$ for $\varepsilon_{\text{mach}}$ small enough.
If $\rho = O(1)$ uniformly for all matrices of a given dimension $m$, then GE w/PP is backward stable; otherwise it is not.

Let the mathematical hair-splitting begin!

Consider the worst-case scenario

\[
\begin{bmatrix}
1 \\
-1 & 1 \\
-1 & -1 & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & -1 & \ldots & -1 & 1 \\
-1 & -1 & \ldots & -1 & -1 & 1
\end{bmatrix}
= \begin{bmatrix}
1 \\
-1 & 1 \\
-1 & -1 & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & -1 & \ldots & -1 & 1 \\
-1 & -1 & \ldots & -1 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
-1 & 1 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & -1 & \ldots & -1 & 1 \\
1 & 2^{m-2}
\end{bmatrix}
\]

Here $\rho = 2^{m-1}$, which is the maximal value $\rho$ can take for GE w/PP.
A growth factor of $2^{m-1}$ corresponds to a loss of $\sim (m - 1)$ bits of information (Recall: we have at most 52 binary digits in IEEE-754-1985 double precision floating point computations).

According the worst-case estimate we cannot safely operate on matrices of dimension larger than $52 \times 52$, and in that case only have one bit of information! This is an intolerable state of affairs for practical computations!!!
On the other hand... We have a uniform bound \(2^{m-1}\) on the growth factor for \((m \times m)\)-matrices, thus according to our previous definitions of backward stability; GE w/PP is backward stable.

Clearly, for practical purposes, this is an absurd conclusion. In this context, let’s put the previous formal definition of backward stability aside; and say that the worst-case scenario indicates that GE w/PP can be unstable.
Practical Stability of Gaussian Elimination

Now... If GE w/PP is so unstable, why is it so famous and popular?!?

“Despite worst-case examples, GE w/PP is utterly stable in practice. Large factors $U$ like the one in the worst-case scenario never seem to appear in real applications. In 50 years of computing no matrix problems that excite explosive instability are known to have arisen under natural circumstances.”

[Trefethen-&-Bau (1997), p.166]

In “Matrix Computations” by Golub & Van-Loan, the upper bounds for the growth factors for partial and complete pivoting are given as

$$
\rho_{PP} \leq 2^{m-1}, \quad \rho_{CP} \leq 1.8m^{\left(\frac{\ln m}{4}\right)}.
$$
Curious...

The number of matrices with large growth factors is very small — if we select a random matrix in $\mathbb{C}^{m \times m}$ it turns out that a practical bound on $\rho_{PP}$ is given by $\sqrt{m}$. This is illustrated below.

**Figure:** The growth factors for GE w/PP for 500 random matrices ranging in size from $(2 \times 2)$ to $(1448 \times 1448)$. The blue line (left panel) corresponds to the practical bound $\sqrt{m}$; and the red line (right panel only) corresponds to the worst-case bound for complete pivoting, $\rho_{cp}$. 
Curious... Where is the $\rho_{pp}$ line?!

Figure: The corresponding values for $\rho_{pp}$ are $\geq \{2, 8, 16, 128, 10^3, 10^4, 10^6, 10^9, 10^{13}, 10^{18}, 10^{26}, 10^{38}, 10^{54}, 10^{76}, 10^{108}, 10^{153}, 10^{217}, 10^{307}, 10^{435}, 10^{616}, 10^{871}, 10^{1232}, 10^{1743}\}$, whereas in this ($m \in \{2, \ldots, 5792\}$) range, $\rho_{cp} < 2.6 \cdot 10^8$; and $\sqrt{m} \leq 77$. 
GE w/PP Bottom Line

The bottom line is that GE w/PP works well “almost always.”

It is almost impossible to prove any useful result in this context.

Vigorous hand-waving and numerical recovery of the probability density functions for the growth-factor vs. the matrix size can be used to get indications that the number of matrices with large growth factors is exponentially small in a probabilistic sense.

See e.g. Trefethen-&-Bau pp.166–170, for some discussion.
We now turn our attention to application of Gaussian Elimination / LU-Factorization to a special class of matrices —

Definition (Hermitian Positive Definite)

$A \in \mathbb{C}^{m \times m}$ is **Hermitian Positive Definite** if $A = A^*$, and

$$\vec{x}^* A \vec{x} > 0, \quad \forall \vec{x} \in \mathbb{C}^m - \{\vec{0}\}.$$ 

This type of matrices show up **many** applications — due to symmetry (reciprocity) in physical systems.

My favorite application is **optimization** [MATH 693A], where we constantly build second order models

$$m_k(\vec{p}) = f(\vec{x}_k) + \vec{p} \nabla f(\vec{x}_k) + \frac{1}{2} \vec{p}^* B_k \vec{p}$$

where the matrix $B_k \approx \nabla^2 f(\vec{x}_k)$ is symmetric (Hermitian) positive definite.
Hermitian Positive Definite (HPD) Matrices: Properties

Let $A \in \mathbb{C}^{m \times m}$ be HPD.

- $\lambda(A) \in \mathbb{R}^+$.  
- Eigenvectors that correspond to distinct eigenvalues of a Hermitian matrix are orthogonal (For general matrixes we only get linear independence).
- $\forall X \in \mathbb{C}^{m \times n}, m \geq n, \text{rank}(X) = n; X^*AX$ is also HPD.
- By selecting $X \in \mathbb{C}^{m \times n}$ to be a matrix with a 1 in each column, and zeros everywhere else, we can write any $(n \times n)$ principal sub-matrix of $A$ in the form $X^*AX$. It follows that every principal sub-matrix of $A$ must be HPD, and in particular $a_{ii} \in \mathbb{R}^+$. 
We now turn to the main task at hand — decomposing a HPD matrix into triangular factors, $R^* R$...

We assume that $A$ is an HPD matrix, and write it in the form

\[
\begin{bmatrix}
\alpha & \vec{w}^* \\
\vec{w} & B
\end{bmatrix} =
\begin{bmatrix}
\beta & 0^* \\
\vec{w}/\beta & I(n-1)
\end{bmatrix}
\begin{bmatrix}
1 & 0^* \\
0 & B - \vec{w}\vec{w}^*/\alpha
\end{bmatrix}
\begin{bmatrix}
\beta & \vec{w}^*/\beta \\
\vec{0} & I(n-1)
\end{bmatrix}
\]

Where

\[
\beta = \sqrt{\alpha}, \quad 0^* \text{ is the zero-vector,} \quad (B - \vec{w}\vec{w}^*/\alpha) \equiv (B - \vec{w}\vec{w}^*)/\alpha,
\]

$I(n-1)$ is the $(n-1) \times (n-1)$-identity matrix

Before moving forward, we check the matrix identity...
We have

\[
\begin{bmatrix}
\beta & 0^* \\
\vec{w} / \beta & I(n-1)
\end{bmatrix}
\begin{bmatrix}
1 & 0^* \\
0 & B - ww' / a
\end{bmatrix}
\begin{bmatrix}
\beta & \vec{w}^* / \beta \\
0 & I(n-1)
\end{bmatrix}
\]

Multiplying the first two matrices, and then third together gives

\[
\begin{bmatrix}
\beta & 0^* \\
\vec{w} / \beta & B - ww' / a
\end{bmatrix}
\begin{bmatrix}
\beta & \vec{w}^* / \beta \\
0 & I(n-1)
\end{bmatrix}
= \begin{bmatrix}
\alpha & \vec{w}^* \\
\vec{w} & B
\end{bmatrix}
\]

as desired.
It can be shown (see slides 31–32) that the sub-matrix \((B - \vec{w}\vec{w}^*/\alpha)\) is also HPD.

We can now define the Cholesky Factorization recursively:

\[
R^{(n)} = \begin{bmatrix}
\beta & \vec{w}^*/\beta \\
\vec{0} & R^{(n-1)}
\end{bmatrix}
\]

Where \(R^{(n-1)} = R^{(n-1)}\) is the Cholesky factor \(R\) associated with \((B - \vec{w}\vec{w}^*/\alpha)\), i.e. \([R^{(n-1)}]^*[R^{(n-1)}] = (B - \vec{w}\vec{w}^*/\alpha)\).

A note on the implementation (next slide): Since we only need to compute one of the triangular parts (it's Hermitian, remember?!?) of the factorization, the Cholesky factorization uses about 1/2 the operations of a general \(LU\)-factorization.
\% Cholesky Factorization of an m-by-m matrix A  
for i = 1:m  
\%  
\% compute $\vec{w}^*/\beta$  
\%  
A(i, i) = sqrt(A(i, i));  
A(i, (i+1):m) = A(i, (i+1):m) / A(i, i);  
\%  
\% compute the upper triangular part of $B - \vec{w}\vec{w}^*/\alpha$  
\%  
for j = (i+1):m  
    A(j, j:m) = A(j, j:m) - A(i, j:m) * A(i, j)';  
end  
\%  
\% We zero out the sub-diagonal elements, since  
\% the answer is an upper triangular matrix.  
\%  
A(((i+1):m, i) = zeros(m-i, 1);  
end
Cholesky Factorization: Existence, Uniqueness, and Work

**Theorem**

*Every HPD matrix* \( A \in \mathbb{C}^{m \times m} \) *has a unique Cholesky factorization.*

The existence follows from the argument on slides 31–32, and uniqueness from the algorithm. □

Compared with standard Gaussian elimination / LU-factorization we are saving about half the operations since we only form the upper triangular part \( R \)

<table>
<thead>
<tr>
<th>Method</th>
<th>Work (( m^3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky ( R^* R ) Factorization</td>
<td>( \frac{m^3}{3} )</td>
</tr>
<tr>
<td>LU-Factorization</td>
<td>( \frac{2m^3}{3} )</td>
</tr>
<tr>
<td>QR: Householder</td>
<td>( \frac{4m^3}{3} )</td>
</tr>
<tr>
<td>QR: Gram-Schmidt</td>
<td>( 2m^3 )</td>
</tr>
<tr>
<td>SVD</td>
<td>( 13m^3 )</td>
</tr>
</tbody>
</table>
Cholesky Factorization: Stability

Usually when we see this table

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholesky $R^*R$ Factorization</td>
<td>$\frac{m^3}{3}$</td>
</tr>
<tr>
<td>LU-Factorization</td>
<td>$\frac{2m^3}{3}$</td>
</tr>
<tr>
<td>QR: Householder</td>
<td>$\frac{4m^3}{3}$</td>
</tr>
<tr>
<td>QR: Gram-Schmidt</td>
<td>$2m^3$</td>
</tr>
<tr>
<td>SVD</td>
<td>$13m^3$</td>
</tr>
</tbody>
</table>

we note that with increased cost comes increased stability. The Cholesky factorization is the one pleasant exception!

All the subtle things that can go wrong in general LU-factorization (Gaussian elimination) are safe in the Cholesky factorization context!

**Cholesky factorization is always backward stable!**

*(For HPD matrices, that is.)*
Cholesky Factorization: Stability

In the 2-norm we have $\|R\| = \|R^*\| = \sqrt{\|A\|}$, thus the growth factor cannot be large. We also note that we can safely compute the Cholesky factorization without pivoting.

**Theorem**

Let $A \in \mathbb{C}^{m \times m}$ be HPD, and let $R^* R = A$ be computed using the Cholesky factorization algorithm in a floating point environment satisfying the floating point axioms. For sufficiently small $\varepsilon_{\text{mach}}$, this process is guaranteed to run to completion (no zero or negative entries $r_{kk}$ will arise), generating a computed factor $\tilde{R}$ that satisfies

$$\tilde{R}^* \tilde{R} = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{\text{mach}})$$

for some $\delta A \in \mathbb{C}^{m \times m}$. 
Solving $A\vec{x} = \vec{b}$ using Cholesky Factorization

If $A$ is HPD, the standard (best) way to solve $A\vec{x} = \vec{b}$ is by Cholesky decomposition.

Once we have $R^*R\vec{x} = \vec{b}$, we get the solution by solving $R^*\vec{y} = \vec{b}$ (by forward substitution), followed by $R\vec{x} = \vec{y}$ (by backward substitution). Each triangular solve requires $\sim m^2$ operations, so the total work is $\sim \frac{1}{3}m^3$. 
We have the following important result

**Theorem**

The solution of an HPD system $A\vec{x} = \vec{b}$ via Cholesky factorization is backward stable, generating a computed solution $\tilde{\vec{x}}$ that satisfies

$$(A + \Delta A)\tilde{\vec{x}} = \tilde{\vec{b}}, \quad \frac{||\Delta A||}{||A||} = O(\varepsilon_{mach})$$

for some $\Delta A \in \mathbb{C}^{m \times m}$. 
One More Comment

If we have a Hermitian matrix $A \in \mathbb{C}^{m \times m}$ the best way to check if it is also Positive Definite is to try to compute the Cholesky factorization.

If $A$ is not HPD, then the Cholesky factorization will break down in the sense that

$$\sqrt{r_{kk}} \quad \text{or, if you want} \quad \sqrt{A(i, i)}$$

will fail (if $r_{kk} < 0$) or the subsequent division by $\sqrt{r_{kk}}$ will fail (if $r_{kk} = 0$).

Usually, in applications (such as optimization) we require $A$ to be sufficiently HPD, meaning that we must have $r_{kk} \geq \delta > 0$ for some $\delta$. Quite possibly $\delta \in \{ \sqrt{\varepsilon_{\text{mach}}}, 3\sqrt{\varepsilon_{\text{mach}}} \}$. 
Use Gaussian Elimination with Partial Pivoting, create plots like TB-Figure-22.1, and TB-Figure-22.2

- For matrices with random, normally distributed $N(0, 1)$ entries:
  6.5.1 Growth factor $\rho$ for GE w/PP. (TB-Figure-22.1) — Use at least 1,024 matrices with varying sizes (up to at least $2,048 \times 2,048$ matrices)
  6.5.2 Probability density of $\rho$. (TB-Figure-22.2) — Use at least $1,048,576$ matrices of each $(m \times m)$ size, $m \in \{8, 16, 32, 64\}$.

- For matrices with random, uniformly distributed in $[0, 1]$ entries:
  6.5.3 Growth factor $\rho$ for GE w/PP. (variant of TB-Figure-22.1) — Use at least 1,024 matrices with varying size (up to at least $2,048 \times 2,048$ matrices)
  6.5.4 Probability density of $\rho$. (variant of TB-Figure-22.2) — Use at least $1,048,576$ matrices of each $(m \times m)$ size, $m \in \{8, 16, 32, 64\}$.
  6.5.5 Comment on similarities / differences of normally vs. uniformly distributed matrix entries.

**Hint:** For computational efficiency, use built-in/library $LU$-factorizations with partial pivoting — `lu()` or `scipy.linalg.lu()` — *read the fine documentation.*
Reference: Proof that $B - \vec{w}\vec{w}^*/\alpha$ is HPD

If $A$ is HPD, and $X$ is a non-singular matrix, then $B = X^*AX$ is also HPD: since $X$ is non-singular $\vec{x} \neq 0 \Rightarrow X\vec{x} \neq 0$, hence

$$\forall \vec{x} \neq 0, \quad \vec{x}^*B\vec{x} = \vec{x}^*X^*AX\vec{x} = (X\vec{x})^*A(X\vec{x}) > 0$$

Now, with the representation

$$A = \begin{bmatrix}
\beta^2 & \vec{w}^* \\
\vec{w} & B
\end{bmatrix}$$

We select

$$X = \begin{bmatrix}
1/\beta & -\vec{w}^*/\beta^2 \\
\vec{0} & I(n-1)
\end{bmatrix}, \quad X^* = \begin{bmatrix}
1/\beta & \vec{0}^* \\
-\vec{w}/\beta^2 & I(n-1)
\end{bmatrix}$$
Now,

\[
X^*AX = \begin{bmatrix}
1/\beta & \vec{0}^* \\
\vec{w}/\beta^2 & I(n-1)
\end{bmatrix}
\begin{bmatrix}
\beta^2 & \vec{w}^* \\
\vec{w} & B
\end{bmatrix}
\begin{bmatrix}
1/\beta & -\vec{w}^*/\beta^2 \\
\vec{0} & I(n-1)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\beta & \vec{w}^*/\beta \\
\vec{0} & B - \vec{w}\vec{w}^*/\alpha
\end{bmatrix}
\begin{bmatrix}
1/\beta & -\vec{w}^*/\beta^2 \\
\vec{0} & I(n-1)
\end{bmatrix}
= \begin{bmatrix}
1 & \vec{0} \\
\vec{0} & B - \vec{w}\vec{w}^*/\alpha
\end{bmatrix}
\]

It now follows from the definition (use \(\vec{x} \neq 0\) such that \(x_1 = 0\)) that \(B - \vec{w}\vec{w}^*/\beta^2\) is also HPD.