Chapter 5: Linear Systems: Direct Methods

Uri M. Ascher and Chen Greif
Department of Computer Science
The University of British Columbia
{ascher,greif}@cs.ubc.ca

Slides for the book
http://www.ec-securehost.com/SIAM/CS07.html
Goals of this chapter

- To learn practical methods to handle the most common problem in numerical computation;
- to get familiar (again) with the ancient method of Gaussian elimination in its modern form of LU decomposition, and develop pivoting methods for its stable computation;
- to consider LU decomposition in the very important special cases of symmetric positive definite and sparse matrices;
- to study the expected quality of the computed solution, introducing as we go the fundamental concept of a condition number.
Outline

- Gaussian elimination and backward substitution
- LU decomposition
- Pivoting strategies
- Efficient implementation
- Cholesky decomposition
- Sparse matrices
- Permutations and ordering strategies
- Estimating error and the condition number
In general

- Here and in Chapter 7 we consider the problem of finding $x$ which solves

$$Ax = b,$$

where $A$ is a given, real, nonsingular, $n \times n$ matrix, and $b$ is a given, real vector.

- **Such problems are ubiquitous in applications!**

- Two solution approaches:
  - **Direct methods**: yield exact solution in absence of roundoff error.
    - Variations of **Gaussian elimination**.
    - Considered in this chapter
  - **Iterative methods**: iterate in a similar fashion to what we do for nonlinear problems.
    - Use only when direct methods are ineffective.
    - Considered in Chapter 7
Backward substitution

- Special case: $A$ is an upper triangular matrix

\[ A = \begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  & a_{22} & \ddots & \vdots \\
  & & \ddots & \vdots \\
  & & & a_{nn}
\end{pmatrix}, \]

i.e., all elements below the main diagonal are zero: $a_{ij} = 0, \forall i > j$.

- The algorithm:

\[
\text{for } k = n : -1 : 1 \\
\quad x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}} \\
\text{end}
\]
Example

\[
x_1 - 4x_2 + 3x_3 = -2 \\
5x_2 - 3x_3 = 7 \\
-2x_3 = -2
\]

In matrix form:

\[
\begin{pmatrix}
1 & -4 & 3 \\
0 & 5 & -3 \\
0 & 0 & -2
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
-2 \\
7 \\
-2
\end{pmatrix}.
\]

Backward substitution: \( x_3 = \frac{-2}{-2} = 1 \), then \( x_2 = \frac{1}{5} (7 + 3 \cdot 1) = 2 \), then \( x_1 = -2 + 4 \cdot 2 - 3 \cdot 1 = 3 \).
Forward substitution

- Special case: \( A \) is a **lower triangular** matrix

\[
A = \begin{pmatrix}
a_{11} & a_{21} & a_{22} \\
a_{21} & a_{22} & & \ddots \\
& \ddots & \ddots & \ddots \\
& & a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix},
\]

where all elements above the main diagonal are zero: \( a_{ij} = 0, \forall i < j \).

- The algorithm:

\[
\text{for } k = 1 : n \\
x_k = \frac{b_k - \sum_{j=1}^{k-1} a_{kj} x_j}{a_{kk}}
\]

end
Gaussian elimination

- Can multiply a row of $Ax = b$ by a scalar and add to another row: elementary transformation.
- Use this to transform $A$ to upper triangular form:

$$MAx = Mb, \quad U = MA.$$ 

- Apply backward substitution to solve $Ux = Mb$. 

![Gaussian elimination diagrams](image-url)
Gaussian elimination (basic)

for $k = 1 : n - 1$
    for $i = k + 1 : n$
        $l_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1 : n$
            $a_{ij} = a_{ij} - l_{ik}a_{kj}$
        end
        $b_i = b_i - l_{ik}b_k$
    end
end

Then apply backward substitution.

Note: upper part of $A$ is overwritten by $U$, lower part no longer of interest.
Cost (flop count)

- For the elimination:

\[
\approx 2 \sum_{k=1}^{n-1} (n - k)^2 = 2((n - 1)^2 + (n - 2)^2 + \cdots + 1^2) = \frac{2}{3} n^3 + \mathcal{O}(n^2).
\]

- For the backward substitution:

\[
\approx 2 \sum_{k=1}^{n-1} (n - k) = 2 \frac{(n - 1)n}{2} \approx n^2.
\]
Example

- Solve $Ax = b$ for
  
  $A = \begin{pmatrix} 1 & -4 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}$, $b = \begin{pmatrix} -2 \\ 5 \\ 6 \end{pmatrix}$.

- Gaussian elimination: $(A \mid b) \Rightarrow$

  $\begin{pmatrix} 1 & -4 & 3 & -2 \\ 0 & 5 & -3 & 7 \\ 0 & 10 & -8 & 12 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & -4 & 3 & -2 \\ 0 & 5 & -3 & 7 \\ 0 & 0 & -2 & -2 \end{pmatrix}$.

- Backward substitution: $x_3 = \frac{-2}{-2} = 1$, then $x_2 = \frac{1}{5}(7 + 3 \cdot 1) = 2$, then $x_1 = -2 + 4 \cdot 2 - 3 \cdot 1 = 3$. 
What if we have many right hand side vectors, or we don’t know $b$ right away?

Note that determining transformation $M$ such that $MA = U$ does not depend on $b$.

$M = M^{(n-1)} \ldots M^{(2)} M^{(1)}$, where $M^{(k)}$ is the transformation of the $k$th outer loop step. These are elementary lower triangular matrices, e.g.,

\[
M^{(2)} = \begin{pmatrix}
1 & 1 \\
 & 1 \\
 & -l_{32} & \ddots \\
 & & \ddots & \ddots \\
 & & & -l_{n2} & 1
\end{pmatrix}.
\]
The matrix $M$ is unit lower triangular.

The matrix $L = M^{-1}$ is also unit lower triangular:

$$A = LU, \quad L = \begin{pmatrix}
1 & & & \\
l_{21} & 1 & & \\
l_{31} & l_{32} & 1 & \\
\vdots & \vdots & \ddots & \ddots \\
l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1
\end{pmatrix}.$$
So, Gaussian elimination is equivalent to:

1. decompose $A = LU$.
   
   *Now for a given $b$ we have to solve $L(Ux) = b$.*

2. use forward substitution to solve $Ly = b$;

3. use backward substitution to solve $Ux = y$. 
Example

\[ A = \begin{pmatrix} 1 & -4 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix}. \]

Obtain

1. \( l_{21} = \frac{1}{1} = 1, \ l_{31} = \frac{3}{1} = 3, \) so

\[
M^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -3 & 0 & 1 \end{pmatrix}, \quad A^{(1)} = M^{(1)}A = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 10 & -8 \end{pmatrix}.
\]

2. \( l_{32} = \frac{10}{5} = 2, \) so

\[
M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}, \quad A^{(2)} = M^{(2)}A^{(1)} = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}.
\]
Example (cont.)

- We thus obtain

\[ U = A^{(2)} = M^{(2)} A^{(1)} = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}, \]

and collect the multipliers \( l_{21}, l_{31} \) and \( l_{32} \) into the unit lower triangular matrix

\[ L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix}. \]

- Indeed, \( A = LU \):

\[ \begin{pmatrix} 1 & -4 & 3 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -4 & 3 \\ 0 & 5 & -3 \\ 0 & 0 & -2 \end{pmatrix}. \]
Examples where the LU decomposition is useful

- When we have multiple right-hand sides, form once the LU decomposition (which costs $O(n^3)$ flops); then for each right-hand side only apply forward/backward substitutions (which are computationally cheap at $O(n^2)$ flops each).

- Can compute $A^{-1}$ by decomposing $A = LU$ once, and then solving $LUx = e_k$ for each column $e_k$ of the unit matrix. These are $n$ right hand sides, so the cost is approximately $\frac{2}{3}n^3 + n \cdot 2n^2 = \frac{8}{3}n^3$ flops.
  (However, typically we try to avoid computing the inverse $A^{-1}$; the need to compute it explicitly is rare.)

- Compute determinant of $A$ by

  \[
  \det(A) = \det(L) \det(U) = \prod_{k=1}^{n} u_{kk}.
  \]
Example: need for pivoting

- First step of Gaussian elimination:

\[
\begin{pmatrix}
1 & 1 & 1 & | & 1 \\
1 & 1 & 2 & | & 2 \\
1 & 2 & 2 & | & 3
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 1 & 1 & | & 1 \\
0 & 0 & 1 & | & 1 \\
0 & 1 & 1 & | & 2
\end{pmatrix}.
\]

- Second step: Now $a_{22}^{(1)} = 0$ and we’re stuck.
- Simple remedy: exchange rows 2 and 3:

\[
\begin{pmatrix}
1 & 1 & 1 & | & 1 \\
1 & 2 & 2 & | & 3 \\
1 & 1 & 2 & | & 2
\end{pmatrix} \Rightarrow \begin{pmatrix}
1 & 1 & 1 & | & 1 \\
0 & 1 & 1 & | & 2 \\
0 & 0 & 1 & | & 1
\end{pmatrix}.
\]

Here the decomposition has been completed without difficulty.
Partial pivoting

- It is rare to hit precisely a zero pivot, but common to hit a very small one.
- Example:

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 + 10^{-12} & 2 \\
1 & 2 & 2
\end{pmatrix}
\Rightarrow
\begin{pmatrix}
1 & 1 & 1 \\
0 & 10^{-12} & 1 \\
0 & 1 & 1
\end{pmatrix}.
\]

- Now we get a multiplier \( l_{3,2} = 1/10^{-12} = 10^{12} \), so roundoff error in elimination step is magnified by this factor \( 10^{12} \).

- Employ Gaussian elimination with partial pivoting (GEPP) not just to avoid zero pivots but more generally to obtain a stable algorithm.
GEPP

- At each stage $k$ choose $q = q(k)$ as the smallest integer for which
  $$|a_{qk}^{(k-1)}| = \max_{k \leq i \leq n} |a_{ik}^{(k-1)}|,$$

  and interchange rows $k$ and $q$.
- This ensures that pivots are not too small (unless matrix is close to singular) and $|l_{i,k}| \leq 1$, all $i \geq k$.
- $PA = LU$ where $P$ is permutation matrix, e.g.,

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$
Simple GEPP algorithm

for $k = 1 : n - 1$
  for $i = k + 1 : n$
    $q = \arg \max_{k \leq i \leq n} |a_{ik}^{(k-1)}|$
    exchange rows $k$ and $q$
    $l_{ik} = \frac{a_{ik}}{a_{kk}}$
    for $j = k + 1 : n$
      $a_{ij} = a_{ij} - l_{ik} \cdot a_{kj}$
    end
    $b_i = b_i - l_{ik} \cdot b_k$
  end
end
Forming $PA = LU$

- It’s not so obvious, but it’s true, that with
  
  $$B = M^{(n-1)}P^{(n-1)} \cdots M^{(2)}P^{(2)}M^{(1)}P^{(1)}, \quad P = P^{(n-1)} \cdots P^{(2)}P^{(1)},$$

  we get $L$ lower triangular and
  
  $$B = L^{-1}P.$$

- The matrix $L$ is lower triangular, although not the same as it would be without pivoting. It is obtained by a similar sequence of steps as before, with the addition of permutation steps.

- The permutation matrix $P$ is orthogonal, so
  
  $$A = (P^T L)U.$$

$P^T L$ is “psychologically lower triangular”.

In practice, keep record of permutations in a 1D array.
Example revisited (1/3)

Same matrix we worked on a few slides ago, now with pivoting:

\[
A = \begin{pmatrix}
1 & -4 & 3 \\
1 & 1 & 0 \\
3 & -2 & 1
\end{pmatrix}.
\]

Go through first column and find pivot:

\[
P^{(1)} = \begin{pmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{pmatrix} ; \quad P^{(1)}A = \begin{pmatrix}
3 & -2 & 1 \\
1 & 1 & 0 \\
1 & -4 & 3
\end{pmatrix}.
\]

So, we have

\[
M^{(1)} = \begin{pmatrix}
1 & 0 & 0 \\
-\frac{1}{3} & 1 & 0 \\
-\frac{1}{3} & 0 & 1
\end{pmatrix}, \quad A^{(1)} = M^{(1)}P^{(1)}A = \begin{pmatrix}
3 & -2 & 1 \\
0 & \frac{5}{3} & -\frac{1}{3} \\
0 & -\frac{10}{3} & \frac{8}{3}
\end{pmatrix}.
\]

Now, work on $A^{(1)}$:

$$P^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}; \quad P^{(2)}A^{(1)} = \begin{pmatrix} 3 & -2 & 1 \\ 0 & -\frac{10}{3} & \frac{8}{3} \\ 0 & \frac{5}{3} & -\frac{1}{3} \end{pmatrix},$$

and we have

$$M^{(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix}, \quad A^{(2)} = M^{(2)}P^{(2)}M^{(1)}P^{(1)}A = \begin{pmatrix} 3 & -2 & 1 \\ 0 & -\frac{10}{3} & \frac{8}{3} \\ 0 & \frac{5}{3} & -\frac{1}{3} \end{pmatrix}.$$

So the upper triangular $U$ is $U = A^{(2)} = M^{(2)}P^{(2)}M^{(1)}P^{(1)}A$. 


Example revisited (3/3)

- Let us find $L$ and $P$. Write

$$U = M^{(2)} P^{(2)} M^{(1)} P^{(1)} A = \underbrace{M^{(2)}}_{\tilde{M}^{(2)}} \underbrace{P^{(2)} M^{(1)} P^{(2)^T}}_{\tilde{M}^{(1)}} \underbrace{P^{(2)} P^{(1)}}_{P} A.$$  

- Next, take the elements of $L$ below the diagonal to be those of the $\tilde{M}^{(k)}$ with flipped signs; the permutation matrix $P$ is just the product of the $P^{(k)}$:

$$L = \begin{pmatrix}
1 & 0 & 0 \\
\frac{1}{3} & 1 & 0 \\
\frac{1}{3} & -\frac{1}{2} & 1
\end{pmatrix};
\quad U = \begin{pmatrix}
3 & -2 & 1 \\
0 & -\frac{10}{3} & \frac{8}{3} \\
0 & 0 & 1
\end{pmatrix};
\quad P = \begin{pmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{pmatrix}.$$  

Exercise: confirm that indeed, $PA = LU$.

- In MATLAB obtain these matrices by the commands

```matlab
A=[1 -4 3; 1 1 0; 3 -2 1];
[L,U,P]=lu(A);
```

- For more on the general principle illustrated in this example, see pages 107–108 in the book, as well as Exercises 7 and 8 of Chapter 5.
GEPP stability

- Want to be assured that
  \[ g_n(A) = \max_{i,j,k} |a_{i,j}^{(k)}| \]
  does not grow exponentially in \( n \). However, this is easily said than done!
- Bad scaling of rows can fool the GEPP we saw, because multiplying a row of \((A \mid b)\) by an arbitrary nonzero constant can affect which \( q = k \) maximizes \( |a_{ik}^{(k-1)}| \).
- Can occasionally do better by scaled partial pivoting, where pivot dominance is relative to its original row norm.
- However, provably stable is only the more expensive complete pivoting. And yet, in practice partial pivoting is usually sufficient.
- There are special cases where no pivoting is required, including symmetric positive definite and diagonally dominant matrices.
GEPP vectorization

- Memory access and inter-processor communications can be as expensive as floating point operations.
- A simple way to improve efficiency in MATLAB is to avoid if- for- and while-loops where possible.
- Work with array operations rather than on individual elements.

```matlab
for k = 1:n-1
    % find pivot q ...
    % interchange rows k and q and record this in p
    A([k,q],:) = A([q,k],:); p([k,q]) = p([q,k]);
    % compute the corresponding column of L
    J = k+1:n; A(J,k) = A(J,k) / A(k,k);
    % update submatrix by outer product
    A(J,J) = A(J,J) - A(J,k) * A(k,J);
end
```
Linear systems: Direct Methods

Inner and outer products

Example:

\[ y = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, \quad z = \begin{pmatrix} 0 \\ 1 \\ 3 \end{pmatrix}. \]

- **Inner product**

\[ y^T z = z^T y = 3 \times 0 + 2 \times 1 + 1 \times 3 = 5. \]

- **Outer products**

\[ yz^T = \begin{pmatrix} 0 & 3 & 9 \\ 0 & 2 & 6 \\ 0 & 1 & 3 \end{pmatrix}, \quad zy^T = \begin{pmatrix} 0 & 0 & 0 \\ 3 & 2 & 1 \\ 9 & 6 & 3 \end{pmatrix}. \]

- Note that \( y \) and \( z \) do not need to have the same length for an outer product, although they do for an inner product.
Example

Same matrix as before, now with vectorized GEPP:

\[
A = \begin{pmatrix}
1 & -4 & 3 \\
1 & 1 & 0 \\
3 & -2 & 1
\end{pmatrix}.
\]

Obtain for the first column \( k = 1 \) without pivoting

1. \( J = [2, 3], \ A([2, 3], 1) = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \ A(1, [2, 3]) = (-4 \ 3) \), so the update is

2. 

\[
A([2, 3], [2, 3]) = \begin{pmatrix} 1 & 0 \\ -2 & 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 3 \end{pmatrix} \ast \begin{pmatrix} -4 & 3 \end{pmatrix} \\
= \begin{pmatrix} 5 & -3 \\ 10 & -8 \end{pmatrix}.
\]
Fast memory access and BLAS

- Computer memories are built as *hierarchies*: from faster, smaller and more expensive to slower, larger and cheaper.
  - registers
  - cache
  - memory
  - disk, cloud

- Standardize basic matrix operations into BLAS:
  - BLAS1: $a \times \mathbf{x} + \mathbf{y}$ (SAXPY)
  - BLAS2: matrix-vector operations
  - BLAS3: matrix-matrix operations
Relative error in the solution

• Still consider

\[ Ax = b \]

but now assess quality of approximate solution obtained somehow.

• Denote exact solution \( x \), computed (or given) approximate solution \( \hat{x} \). Want to estimate

\[
\frac{\| x - \hat{x} \|}{\| x \|}.
\]

• Can compute the residual \( \hat{r} = b - A\hat{x} \) and so also \( \frac{\| \hat{r} \|}{\| b \|} \).

Does a small relative residual imply small relative error in solution?
Example

For the problem

\[ A = \begin{pmatrix} 1.2969 & .8648 \\ .2161 & .1441 \end{pmatrix}, \quad b = \begin{pmatrix} .8642 \\ .1440 \end{pmatrix}, \]

consider the approximate solution

\[ \hat{x} = \begin{pmatrix} .9911 \\ -.4870 \end{pmatrix}. \]

Then

\[ \hat{r} = b - A\hat{x} = \begin{pmatrix} -10^{-8} \\ 10^{-8} \end{pmatrix}, \]

so \( \|\hat{r}\|_\infty = 10^{-8}. \)

However, the exact solution is

\[ x = \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \quad \text{so} \quad \|x - \hat{x}\|_\infty = 1.513. \]
Conditioning of problem

- Since $\hat{r} = Ax - A\hat{x} = A(x - \hat{x})$, get

  $$\|x - \hat{x}\| = \|A^{-1}\hat{r}\| \leq \|A^{-1}\|\|\hat{r}\|.$$ 

- Since $Ax = b$, get

  $$\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}.$$ 

- Hence

  $$\frac{\|x - \hat{x}\|}{\|x\|} \leq \kappa(A) \frac{\|\hat{r}\|}{\|b\|},$$

with

$$\kappa(A) = \|A\|\|A^{-1}\|.$$ 

The scalar $\kappa(A)$ is the **condition number** of $A$. 
Quality of solution

- **Backward error analysis**: associate result of numerical algorithm (GEPP) with the exact solution of a perturbed problem

\[(A + \delta A)\hat{x} = b + \delta b.\]

- The job of GEPP is to make $\delta A$ and $\delta b$ small.
- Obtain good quality solution (only) if in addition, $\kappa(A)$ is not too large.
- In our $2 \times 2$ example, in fact, $\kappa(A) \approx 10^8$, and indeed we saw $\|x - \hat{x}\| \sim \kappa(A)\|\hat{r}\|$. 
The condition number

- Always $\kappa(A) \geq 1$.
- For orthogonal matrices, $\kappa_2(Q) = 1$: ideally conditioned!
- $\kappa(A)$ indicates how close $A$ is to being singular, which $\det(A)$ does not.
- If $A$ is symmetric positive definite with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n > 0$ then
  \[ \kappa_2(A) = \frac{\lambda_1}{\lambda_n}. \]
- If $A$ is noningular with singular values $\sigma_1 \geq \cdots \geq \sigma_n > 0$ then
  \[ \kappa_2(A) = \frac{\sigma_1}{\sigma_n}. \]