Chapter 2, Linear Systems

OUTLINE

- *Geometry of Linear Systems*
- *Existence, Uniqueness and Conditioning*
- *Solving Linear Systems*
- *Special Types of Linear Systems*
- *Software for Linear Systems*

Linear Systems

- We now consider solution of linear systems of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ system matrix of the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix},$$

  while $\mathbf{x}$ and $\mathbf{b}$ are n-vectors.

- We will study cases in which these systems are singular or ill-conditioned (i.e., *nearly* singular) and cases where the systems are well-conditioned.

- We start with a brief review of conditions for singularity and of geometric interpretations of linear systems.

Existence and Uniqueness

- An $n \times n$ matrix $\mathbf{A}$ is said to be *nonsingular* if it satisfies any one of the following **equivalent** conditions:

  1. $\mathbf{A}$ has an inverse: $\mathbf{A}^{-1}$ such that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, the identity matrix.
  2. $\det(\mathbf{A}) \neq 0$ (i.e., $\mathbf{A}$ has a nonzero determinant)
  3. rank $(\mathbf{A}) = n$ (the *rank* of a matrix = maximum number of linearly independent rows or columns it has)
  4. For any vector $\mathbf{z} \neq 0$, $\mathbf{Az} \neq 0$

The Geometry of Linear Equations[1]

- Example, 2×2 system:

$$\left.\begin{array}{rcl} 2x - y &=& 1 \\ x + y &=& 5 \end{array}\right\} \iff \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Can look at this system by *rows* or *columns*.
- We will do both.

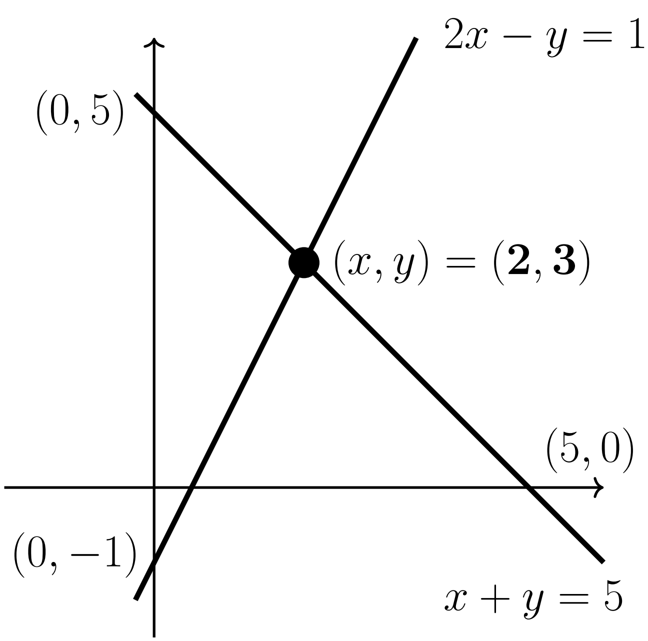
[1]Gilbert Strang: *Linear Algebra and Its Applications*

Row Form

- In the 2×2 system, each equation represents a line:

$$\begin{aligned} 2x - y &= 1 \qquad \text{line 1} \\ x + y &= 5 \qquad \text{line 2} \end{aligned}$$

- The intersection of the two lines gives the unique point $(x, y) = (2, 3)$, which is the solution.



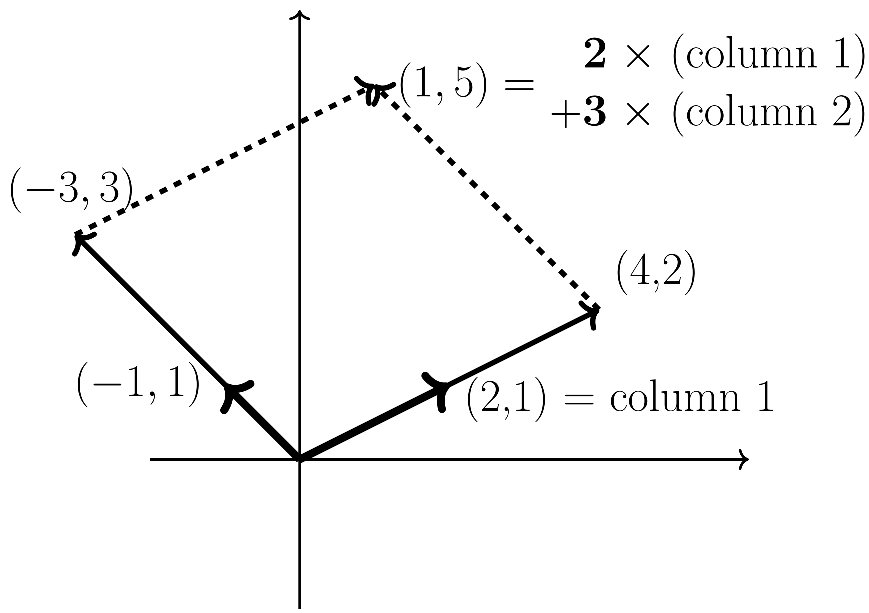
- We remark that the system is relatively *ill-conditioned* if the lines are close to being parallel, that is, if the smallest subtended angle is close to 0.

Column Form

- The second (and more important) geometry is column based.
- Here, we view the system of equations as *one vector equation*:

$$\textbf{Column form} \qquad x\begin{bmatrix} 2 \\ 1 \end{bmatrix} + y\begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}.$$

- The problem is to find coefficients, x and y, such that the combination of vectors on the left equals the vector on the right.



- In this case, the system is *ill-conditioned* if the column vectors are nearly parallel. If these vectors are separated by an angle θ, it's relatively easy to show that the condition number scales as $\kappa \sim \frac{2}{\theta}$ as $\theta \longrightarrow 0$.

Row Form: A Case with n=3.

Three planes:

$$\begin{aligned} 2u + v + w &= 5 \\ 4u - 6v &= -2 \\ -2u + 7v + 2w &= 9 \end{aligned}$$

- Each equation (*row*) defines a plane in $\mathbb{R}^3$.
- The first plane is $2u + v + w = 5$ and it contains points ($\frac{5}{2}$,0,0) and (0,5,0) and (0,0,5).
- It is determined by three points, provided they do not lie on a line.
- Changing 5 to 10 would shift the plane to be parallel this one, with points (5,0,0) and (0,10,0) and (0,0,10).

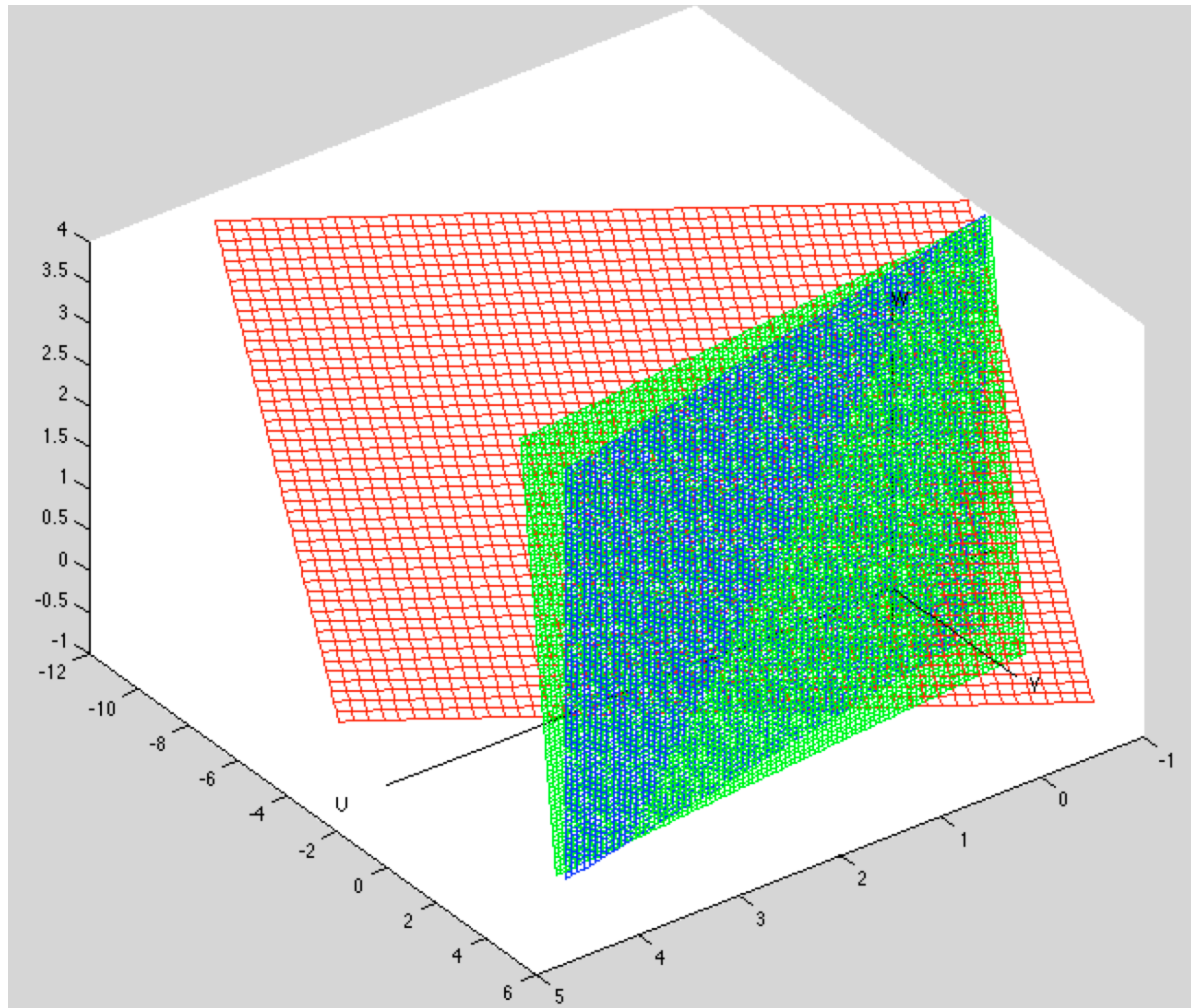
Row Form: A Case with n=3, cont'd.

- The second plane is $4u - 6v = -2$.
- It is vertical because it can take on any w value.
- The intersection of this plane with the first is a *line.*
- The third plane, $-2u + 7v + 2w = 9$ intersects this line at a point, $(u, v, w) = (1, 1, 2)$, which is the solution.
- In n dimensions, the solution is the intersection point of n hyperplanes, each of dimension $n - 1$.   A bit confusing.

Row Form: A Case with n=3, cont'd.

- The **green** and **blue** planes (Eqs. 2 and 3) intersect in a line.
- The **red** plane (Eq. 1) intersects this line.

$$\begin{aligned} 2u + v + w &= 5 \\ 4u - 6v &= -2 \\ -2u + 7v + 2w &= 9 \end{aligned}$$



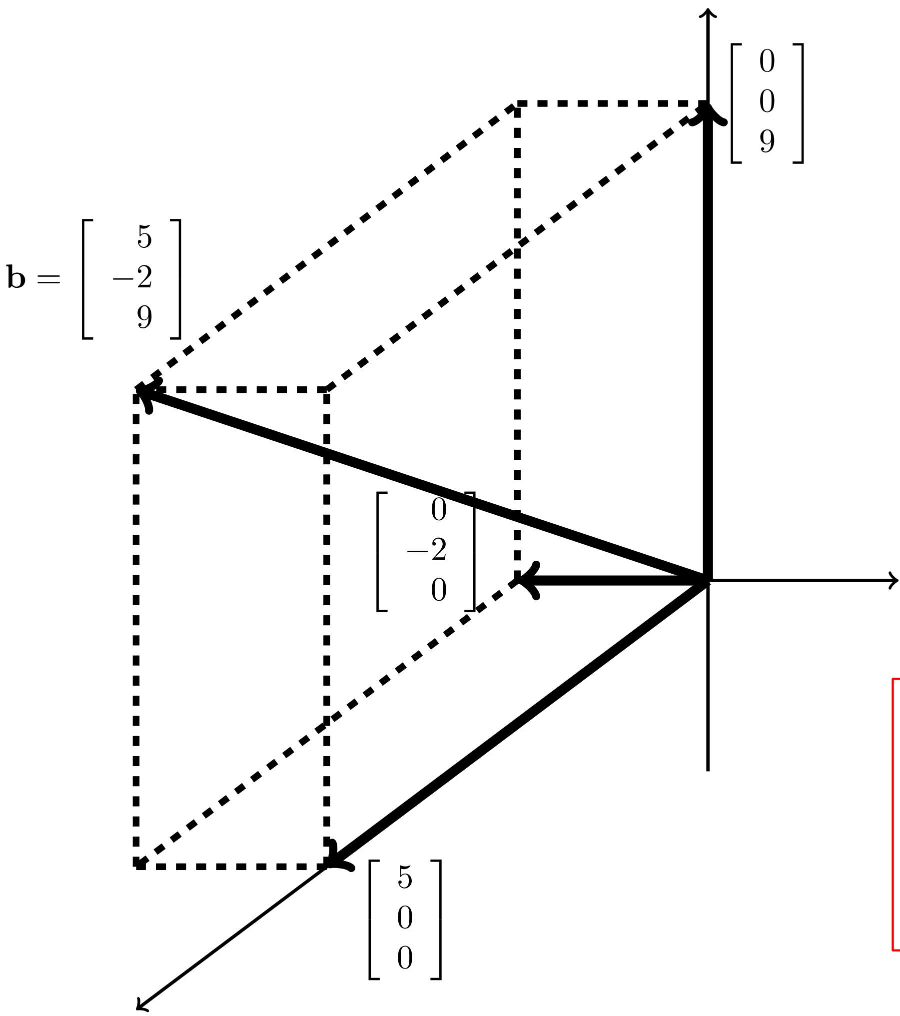
Column Vectors and Linear Combinations

- The preceding system in $\mathbb{R}^3$ can be viewed as the vector equation

$$u \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} + v \begin{bmatrix} 1 \\ -6 \\ 7 \end{bmatrix} + w \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix} = \mathbf{b}.$$

- Our task is to find the multipliers, u, v, and w.
- The vector $\mathbf{b}$ is identified with the point (5,-2,9).
- We can view $\mathbf{b}$ as a list of numbers, a point, or an arrow.
- For $n > 3$, it's probably best to view it as a list of numbers.

Vector Addition Example

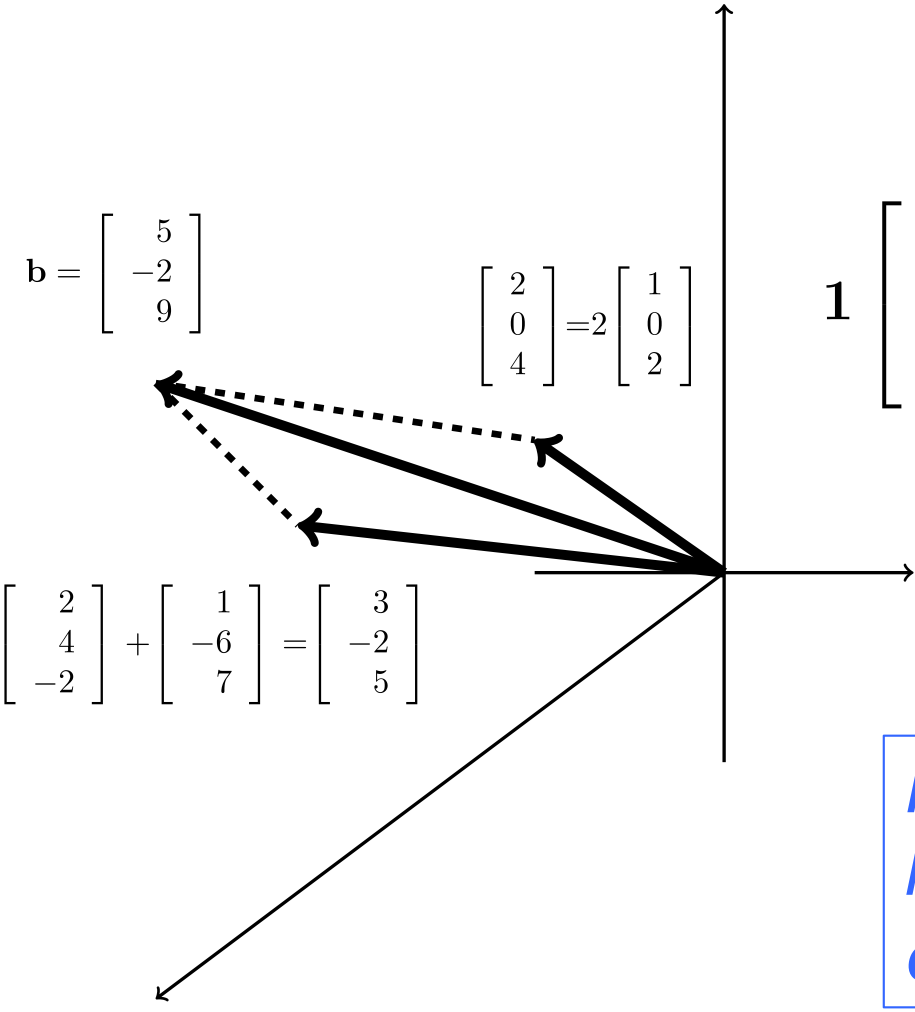


$$\begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix}$$

Here, we write the rhs as a linear combination of the ***orthogonal unit basis vectors***

$$5 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - 2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 9 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

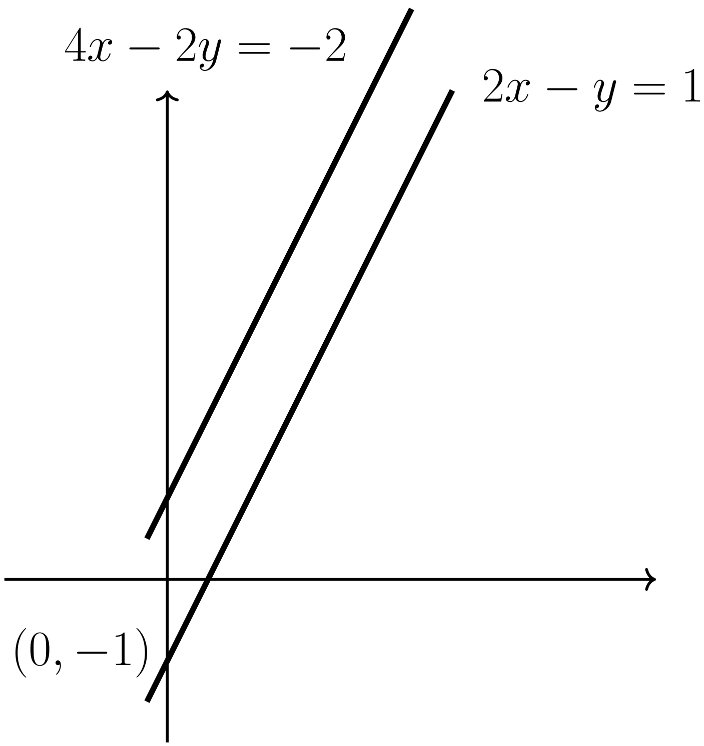
Linear Combination



$$\mathbf{1} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} + \mathbf{1} \begin{bmatrix} 1 \\ -6 \\ 7 \end{bmatrix} + \mathbf{2} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix}$$

Here, we write the rhs as a linear combination of the ***columns of the system matrix.***

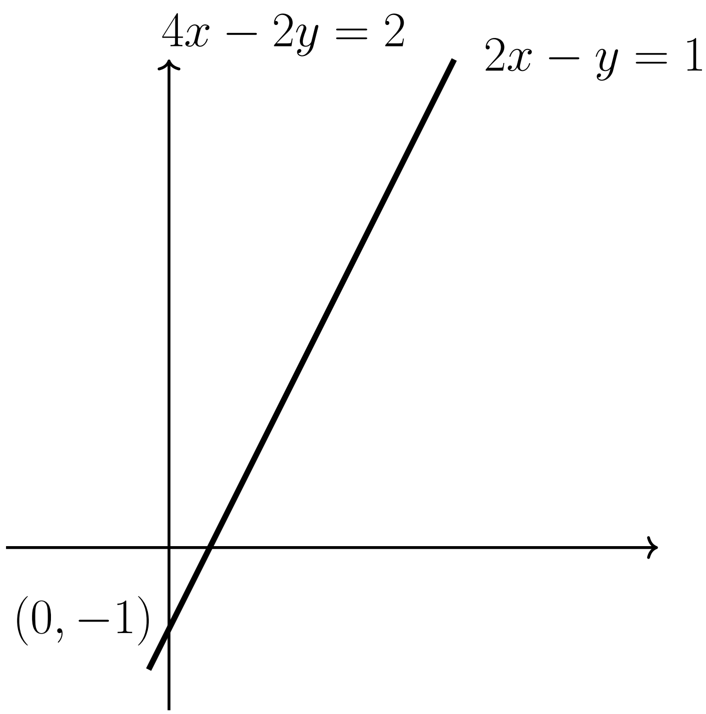
Singular Case: Row Picture



$$
\begin{aligned}
2x - y &= 1 \\
4x - 2y &= -2
\end{aligned}
$$

- No solution.

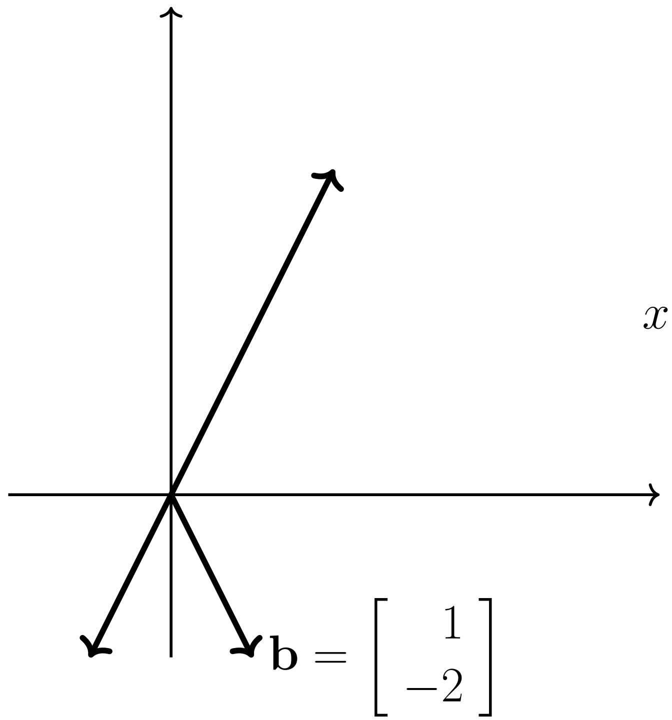
Singular Case: Row Picture



$$\begin{aligned} 2x - y &= 1 \\ 4x - 2y &= 2 \end{aligned}$$

- Infinite number of solutions.

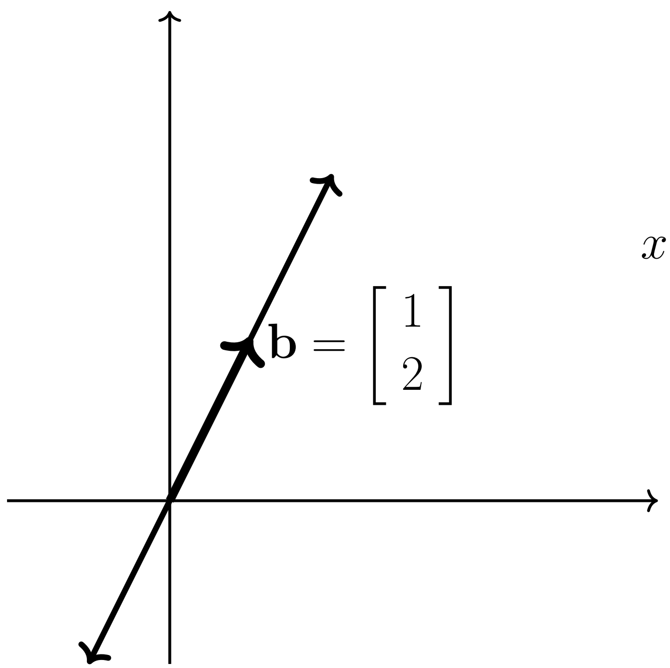
Singular Case: Column Picture



$$x \begin{bmatrix} 2 \\ 4 \end{bmatrix} + y \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

- No solution.

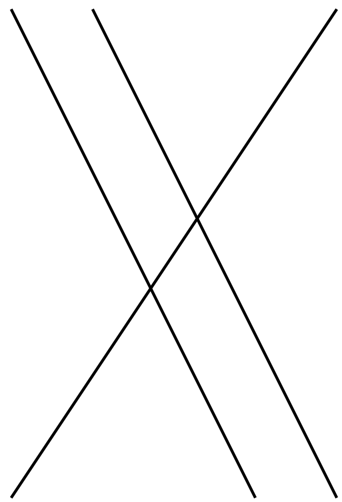
Singular Case: Column Picture



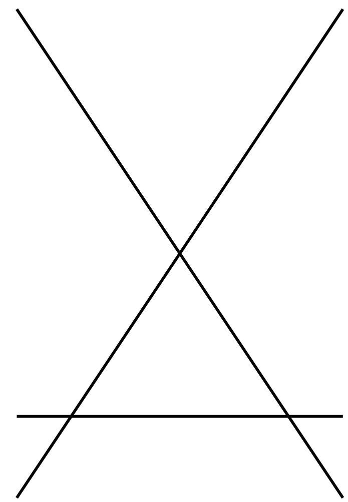
$$x \begin{bmatrix} 2 \\ 4 \end{bmatrix} + y \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

- Infinite number of solutions. ($\mathbf{b}$ coincident with $\mathbf{a}_1$ and $\mathbf{a}_2$.)

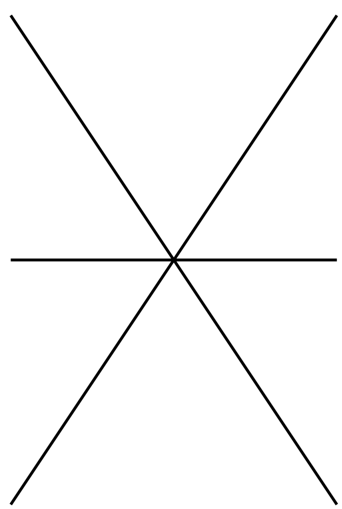
Singular Case: Row Picture with n=3



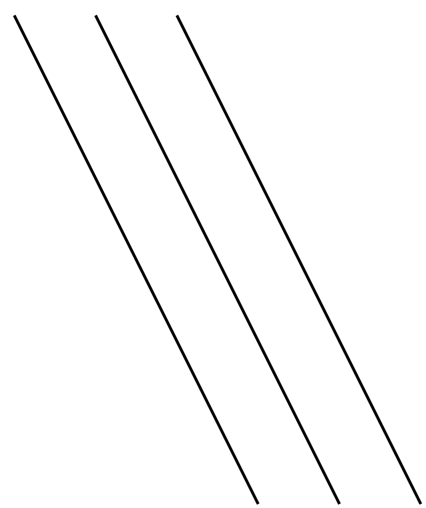
(a) two parallel planes



(b) no intersection



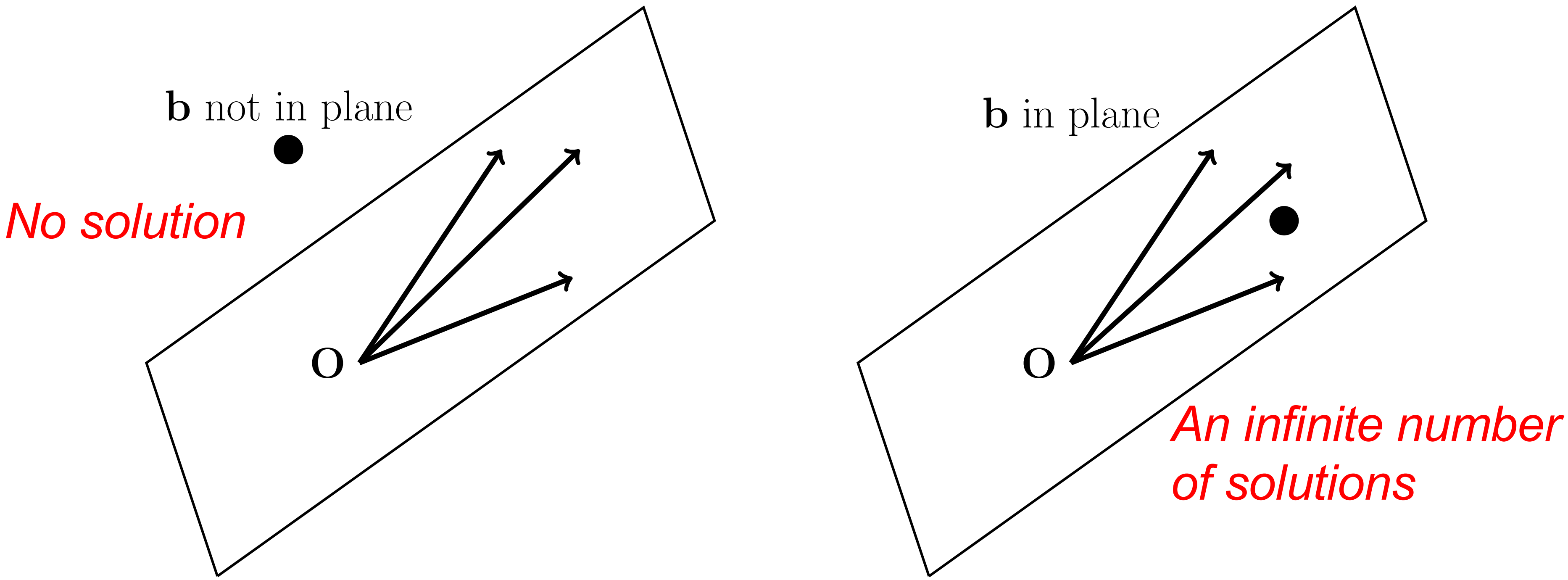
(c) line of intersection



(d) all planes parallel

End-on view of 3 planes.

Singular Case: Column Picture with $n=3$



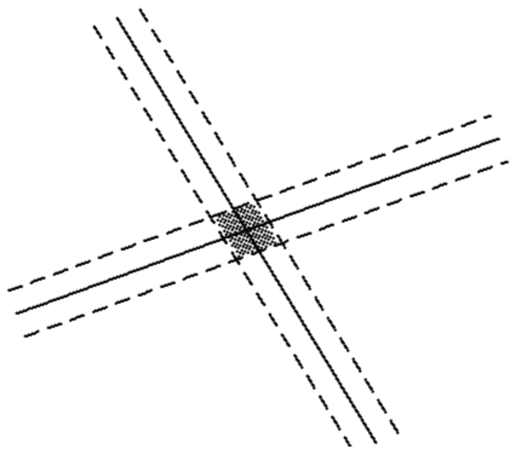
- In this case, the three columns of the system matrix lie in the same plane.

$$\text{Example:} \quad u\begin{bmatrix}1\\2\\3\end{bmatrix} + v\begin{bmatrix}4\\5\\6\end{bmatrix} + w\begin{bmatrix}7\\8\\9\end{bmatrix} = \mathbf{b}.$$

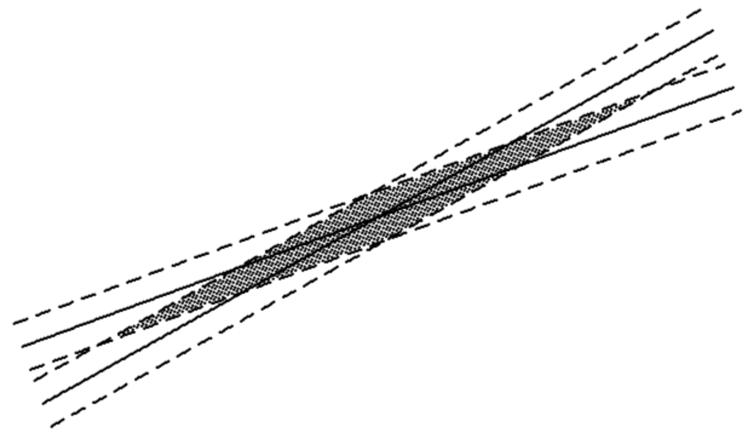
- Our system is *solvable* (we can get to any point in $\mathbb{R}^3$) if the three columns are *linearly independent*.

Nearly Singular Matrices – Row Perspective

- In two dimensions, uncertainty in intersection point of two lines depends on whether lines are nearly parallel



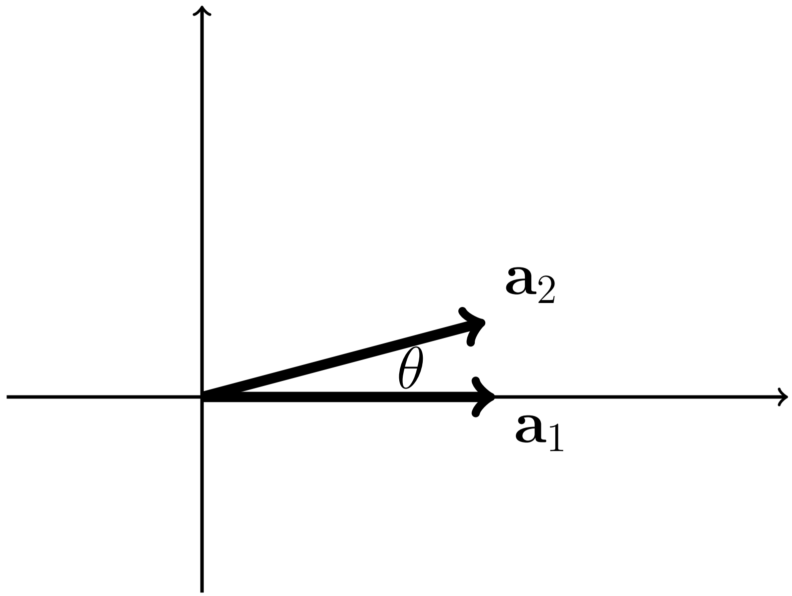
Well-Conditioned



Ill-Conditioned (nearly singular)

[An interesting question: For the 2x2 case, can you relate the angle to the condition number ?]

Nearly Singular Matrices – Column Perspective



$$A = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} 1 & c \\ 0 & s \end{bmatrix}$$

$$c = \cos\theta, \qquad s = \sin\theta.$$

- Clearly, as $\theta \longrightarrow 0$ the matrix becomes singular.
- Can show that

$$\text{cond} = \sqrt{\frac{1+|c|}{1-|c|}} \approx \frac{2}{\theta}$$

for small θ (by Taylor series!)

Matrix Form and Matrix-Vector Products.

- We start with the familiar (row) form

$$\begin{aligned} 2u + v + w &= 5 \\ 4u - 6v &= -2 \\ -2u + 7v + 2w &= 9 \end{aligned}$$

- In matrix form, this is

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix}, \quad \text{or } A\mathbf{u} = \mathbf{b}.$$

- Of course, this must equal our column form,

$$u \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} + v \begin{bmatrix} 1 \\ -6 \\ 7 \end{bmatrix} + w \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix} = \mathbf{b}.$$

Matrix Form and Matrix-Vector Products, 2.

- So, if A is the matrix with columns $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$,

$$A := \begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix} =: \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix}, \qquad \text{and } \mathbf{u} := \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

- Then

$$A\mathbf{u} = u\,\mathbf{a}_1 + v\,\mathbf{a}_2 + w\,\mathbf{a}_3$$

Matrix Form and Matrix-Vector Products, 3.

- In general, if $\mathbf{x}$ is the n-vector

$$\mathbf{x} := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

and A is an $m \times n$ matrix, then

$$\begin{aligned} A\mathbf{x} &= x_1\,\mathbf{a}_1 + x_2\,\mathbf{a}_2 + \cdots + x_n\,\mathbf{a}_n \\ &= \textbf{linear combination of the columns of } A\textbf{.} \end{aligned}$$

- **Always.**

Matrix-Vector Products, Example.

$$\text{If} \qquad \hat{\mathbf{x}} := V \left(V^T A V\right)^{-1} V^T \mathbf{b}$$

$$= V\mathbf{y}.$$

$$\text{Then } \hat{\mathbf{x}} = \textbf{linear combination of the columns of } V\textbf{.}$$

- $\hat{\mathbf{x}}$ lies in the *column space* of V.
- $\hat{\mathbf{x}}$ lies in the *range* of V.
- $\hat{\mathbf{x}} \in \text{span}(V)$

Column Picture Example

- What linear combination of $(1 \;\; 2 \;\; 3)$ and $(1 \;\; 1 \;\; 1)$ will produce the vector $(0 \;\; 2 \;\; 4)$?

$$x_1 \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + x_2 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 4 \end{pmatrix}.$$

- Is it unique?

Sigma Notation

- Let A be an $m \times n$ matrix,

$$A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_j & \cdots & \mathbf{a}_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & & \vdots & & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}.$$

- Then

$$\mathbf{w} = A\mathbf{x} = \sum_{j=1}^{n} x_j \, \mathbf{a}_j = \sum_{j=1}^{n} \mathbf{a}_j \, x_j$$

$$w_i = (A\mathbf{x})_i = \sum_{j=1}^{n} a_{ij} \, x_j$$

Matrix Multiplication

$$\text{If} \quad B = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix},$$

$$\text{Then} \quad C = AB = \begin{bmatrix} A\mathbf{b}_1 & A\mathbf{b}_2 \end{bmatrix}.$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik}\, b_{kj}$$

Q: (Important.) Suppose A and B are $n \times n$ matrices.

- How many floating point operations (flops) are required to compute $C = AB$?
- What is the number of memory accesses?

$$for\ i = 1, \ldots, n,$$
$$j = 1, \ldots, n,$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$= a_{i1} * b_{1j} + a_{i2} * b_{2j} + a_{i3} * b_{3j} + \cdots + a_{in} * b_{nj}.$$

ANSWER:

- *~2n ops, "+" and "*", for each of n^2 results.*
- *→ $2n^3$ operations total.*

Some Special Matrix-Vector Products, 1/2.

- Suppose $V = \mathbf{v}$ and $W = \mathbf{w}$ are $n \times 1$ matrices (i.e., vectors).
- Then

$$C \;=\; V^T W \;=\; \mathbf{v}^T \mathbf{w} \;=\; \sum_{j=1}^{n} v_j w_j \;=\; c$$

  is a 1×1 matrix (i.e., a scalar).
- We refer to $\mathbf{v}^T \mathbf{w}$ as the "dot" or *inner* product of $\mathbf{v}$ and $\mathbf{w}$.

Some Special Matrix-Vector Products, 2/2.

- Suppose $V = \mathbf{v}$ and $W = \mathbf{w}$ are $n \times 1$ matrices (i.e., vectors).
- Then

$$C = VW^T = \mathbf{v}\mathbf{w}^T = \mathbf{v}\left[\, w_1 \; w_2 \; \cdots \; w_n \,\right]$$

$$= \begin{bmatrix} \mathbf{v}w_1 & \mathbf{v}w_2 & \cdots & \mathbf{v}w_n \end{bmatrix}$$

is an $n \times n$ matrix, with each column a multiple of $\mathbf{v}$.

- We refer to $\mathbf{v}\mathbf{w}^T$ as the *outer* product of $\mathbf{v}$ and $\mathbf{w}$.
- It is a matrix of rank 1 and not invertible (unless $n = 1$).
  - *every column is a multiple of* $\mathbf{v}$
  - *every row is a multiple of* $\mathbf{w}^T$

Start here, Lecture 4

Solving a Linear System

Given

- $m \times n$ matrix, $\mathbf{A}$
- m vector $\mathbf{b}$

What are we looking for and when are we allowed to ask the question?

Want: n-vector $\mathbf{x}$ so that $\mathbf{Ax} = \mathbf{b}$

- Linear combination of columns of $\mathbf{A}$ to yield $\mathbf{b}$
- Consider ***square*** case $(m = n)$ for now
- Even then, solution may not exist or may not be unique
- Unique solution exists *iff* $\mathbf{A}$ is nonsingular

Next: Look at *conditioning* of this operation. Need matrix *norms*.

Matrix Norms

- Since we are considering $\mathbf{Ax}$, we need a measure of how $\mathbf{A}$ can influence $\mathbf{x}$.

- Note that $\mathbf{y} = \mathbf{Ax}$ is just a *vector*.

- We have already introduced the p-norms for vectors.

- We can introduce an associated (or *induced*) matrix norm as the scalar $\|\mathbf{A}\|$ that satisfies

  $$\|\mathbf{Ax}\| \le \|\mathbf{A}\| \, \|\mathbf{x}\|$$

  for all $\mathbf{x} \in \mathbb{R}^n$, which simply defines $\|\mathbf{A}\|$ in terms of two vector norms, which we know how to compute.

- $\|\mathbf{A}\|$ is the *maximum stretching* realizable when multiplying $\mathbf{x}$ by $\mathbf{A}$.

  Of course, can have $\|\mathbf{A}\| < 1$

Matrix Norms, continued

- This idea leads to two equivalent definitions

$$\|\mathbf{A}\| \;:=\; \max_{\mathbf{x}\in\mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}$$

$$:= \max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\|$$

- For each *vector* norm, $\|\mathbf{x}\|$, we get a different *matrix norm* $\|\mathbf{A}\|$
- For example, for the vector norm $\|\mathbf{x}\|_2$ we have an associated matrix norm $\|\mathbf{A}\|_2$
- Note that these norms are well defined even if $\mathbf{A}$ is not square.

Identifying Matrix Norms

- What is $\|\mathbf{A}\|_1$? $\|\mathbf{A}\|_\infty$?

- If $\mathbf{A} = [a_{ij}]$,

$$\|\mathbf{A}\|_1 \; = \; \max_{col\ j} \sum_{i=1}^{m} |\, a_{ij} \,| \; = \; \text{maximum column sum of } \mathbf{A}$$

$$\|\mathbf{A}\|_\infty \; = \; \max_{row\ i} \sum_{j=1}^{n} |\, a_{ij} \,| \; = \; \text{maximum row sum of } \mathbf{A}$$

- **Q**: What is $\|I\|$ for the $n \times n$ identity matrix?

Hint: Consider $\mathbf{x} = [\pm 1 \;\; \pm 1 \;\; \cdots \pm 1]^T$ *so that* $\mathbf{Ax}$ *yields a sum on row* i.

Matrix Norm Examples

- What is the 1-norm of the matrix A?
- What is the ∞-norm?

$$A = \begin{bmatrix} 1 & -7 & 1 \\ 1 & 0 & 4 \\ 0 & 1 & 5 \end{bmatrix}$$

- *Hint:*

  - For the ∞-norm, set $\mathbf{x} = [\pm 1 \;\pm 1 \;\ldots\; \pm 1]^T$ with signs chosen to maximize output. $||\mathbf{x}||_\infty = 1$.

  - For the 1-norm, set $\mathbf{x} = [0\; 0 \;\ldots 1 \ldots\; 0]^T$ with row chosen to maximize output. $||\mathbf{x}||_1 = 1$.

Identifying Matrix Norms, continued

- What is $\|\mathbf{A}\|_2$?

- In general, $\|\mathbf{A}\|_2 \;=\; \sigma_1$, the largest *singular value* of $\mathbf{A}$ (more on this later)

- If $\mathbf{A}$ is real, square and symmetric, $\mathbf{A} = \mathbf{A}^T \iff a_{ij} = a_{ji}$, then

$$\|\mathbf{A}\|_2 \;=\; \max_j |\lambda_j| =: \rho(\mathbf{A}),$$

  the *spectral radius* of $\mathbf{A}$, corresponding the eigenvalue of maximum absolute value.

- The eigenvalues are the set of scalars $\lambda_j \in \mathbb{C}$, $j = 1, \ldots, n$, satisfying $\mathbf{A}\mathbf{z}_j = \lambda_j \mathbf{z}_j$ for given *eigenvectors*, $\mathbf{z}_j$.

- If $\mathbf{A}$ symmetric then the λ_js are *real*

Identifying Matrix Norms

- How do matrix and vector norms relate for $n \times 1$ matrices?
- They are the same. WHY?
  - If $\mathbf{A} \in \mathbb{R}^{m\times 1}$, then $\mathbf{x} \in \mathbb{R}^1$ is a *scalar*
  - If $\|\mathbf{x}\| = 1$, then $x = 1$ (or -1), so $\|\mathbf{A}1\| = \|\mathbf{A}\| \cdot 1$
- **Q**: What is 1-norm of an $m \times 1$ matrix?

Properties of Matrix Norms

Matrix norms inherit the vector norm properties:

- $\|\mathbf{A}\| > 0 \iff \mathbf{A} \neq 0$
- $\|\gamma\, \mathbf{A}\| = |\gamma|\, \|\mathbf{A}\|$ for all scalars γ
- $\|\mathbf{A} + B\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$, *triangle inequality*

There are also two *submultiplicativity* properties that result from the induced norm definition,

- $\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\, \|\mathbf{x}\|$
- $\|\mathbf{A}\mathbf{B}\| \leq \|\mathbf{A}\|\, \|\mathbf{B}\|$

In general we will write $\|\cdot\|$ for matrix norms without subscript if the statement is true for any induced norm.

Matrix Norm Examples

Consider

$$\mathbf{A} = \begin{bmatrix} .2 & .7 & 0 \\ .1 & .6 & 0 \\ 0 & 0 & .3 \end{bmatrix}$$

Hint: Consider $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

- What is $\|\mathbf{A}\|_\infty$?
- What is $\|\mathbf{A}\|_1$?
- What is $\lim_{k \longrightarrow \infty} \|\mathbf{x}_k\|_*$ for $\mathbf{x}_k := A^k \mathbf{x}$?
  - For the $* = 1$ case?
  - For the $* = \infty$ case?

- **A:** $\|\mathbf{x}_k\|_* = \|A^k \mathbf{x}\|_* \leq \|A\|_*^k \|\mathbf{x}\|_*$

Conditioning

What is the condition number when solving $\mathbf{Ax} = \mathbf{b}$?

- ***Input***: $\mathbf{b}$ with error $\Delta\mathbf{b}$
- ***Output***: $\mathbf{x}$ with error $\Delta\mathbf{x}$
- Observe $\mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = (\mathbf{b} + \Delta\mathbf{b})$, so $\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$

$$
\begin{aligned}
\frac{\text{rel err in output}}{\text{rel err in input}} &= \frac{\|\Delta\mathbf{x}\|/\|\mathbf{x}\|}{\|\Delta\mathbf{b}\|/\|\mathbf{b}\|} = \frac{\|\Delta\mathbf{x}\|}{\|\Delta\mathbf{b}\|} \cdot \frac{\|\mathbf{b}\|}{\|\mathbf{x}\|} \\
&= \frac{\|\mathbf{A}^{-1}\Delta\mathbf{b}\|}{\|\Delta\mathbf{b}\|} \cdot \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \\
&\leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|
\end{aligned}
$$

Condition Number

- We denote the *condition number* of $\mathbf{A}$ as

$$\kappa(\mathbf{A}) \;=\; \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\|$$

- **Q**: What is the condition number of $\mathbf{A}^{-1}$?

- $\kappa(\mathbf{A})$ is also the condition number associated with matrix-vector multiplication, $\mathbf{y} = \mathbf{A}\mathbf{x}$.

- Notice that $\kappa(\mathbf{A})$ depends on the associated matrix norm, $\|\mathbf{A}\|$.

- If $\mathbf{A}$ is *singular* we define $\kappa = \infty$

Condition Number, continued

- **Example**: Suppose $\kappa(\mathbf{A}) = 100$. What is $\kappa(10\,\mathbf{A})$?

- Consider $\mathbf{B} := 10\,\mathbf{A}$ with $\|\mathbf{A}\| = 5$ and $\|\mathbf{A}^{-1}\| = 20$

- What is $\|\mathbf{B}\|$?

- What is $\|\mathbf{B}^{-1}\|$?

  - $\mathbf{B} = 10\mathbf{A} \;\longleftrightarrow\; \mathbf{B}^{-1} = \mathbf{A}^{-1}10^{-1} = 0.1\mathbf{A}^{-1}$

  - $\kappa(\mathbf{B}) \;=\; \|\mathbf{B}\| \cdot \|\mathbf{B}^{-1}\| \;=\; 10\|\mathbf{A}\| \cdot \left(0.1\,\|\mathbf{A}^{-1}\|\right) \;=\; \kappa(\mathbf{A})$

Properties of Condition Number

- For any matrix $\mathbf{A}$, $\kappa(\mathbf{A}) \geq 1$
- For identity matrix, $\kappa(\mathbf{I}) = 1$
- For any matrix $\mathbf{A}$ and scalar γ, $\kappa(\gamma\mathbf{A}) = \kappa(\mathbf{A})$
- For any diagonal matrix $\mathbf{D} = \text{diag}(d_i)$, $\kappa(\mathbf{D}) = \frac{\max|d_i|}{\min|d_i|}$
- If $\mathbf{A}$ is symmetric positive definite (SPD), $\kappa_2(\mathbf{A}) = \frac{\lambda_{\max}}{\lambda_{\min}}$

- Condition number:

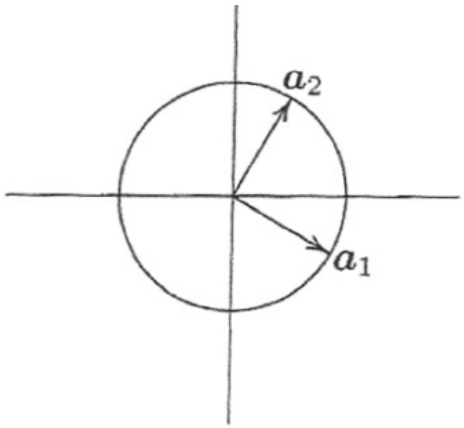
$$\kappa(A) \;:=\; \|A\| \cdot \|A^{-1}\| \;=\; \frac{\max\limits_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|}{\min\limits_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|}.$$

  – To see this, note that $\mathbf{y} = A^{-1}\mathbf{x} \iff \mathbf{x} = A\mathbf{y}$, and

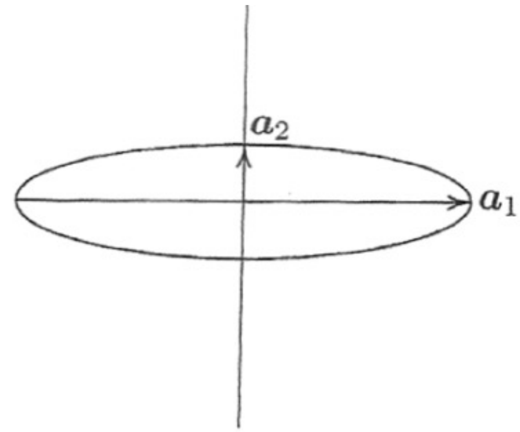
$$\begin{aligned} \|A^{-1}\| \;&=\; \max_{\mathbf{x}\neq 0} \frac{\|A^{-1}\mathbf{x}\|}{\|\mathbf{x}\|}. \;=\; \max_{\mathbf{y}\neq 0} \frac{\|\mathbf{y}\|}{\|A\mathbf{y}\|} \\ &=\; \max_{\|\mathbf{y}\|=1} \frac{1}{\|A\mathbf{y}\|} \\ &=\; \frac{1}{\min_{\|\mathbf{y}\|=1} \|A\mathbf{y}\|}. \end{aligned}$$

- So, condition number is the ratio of max-to-min stretching of A acting on a vector.

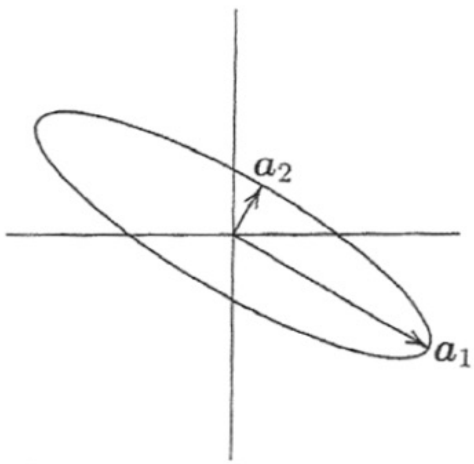
Condition Number Examples
Apply *A* to unit-vector *x* at different angles



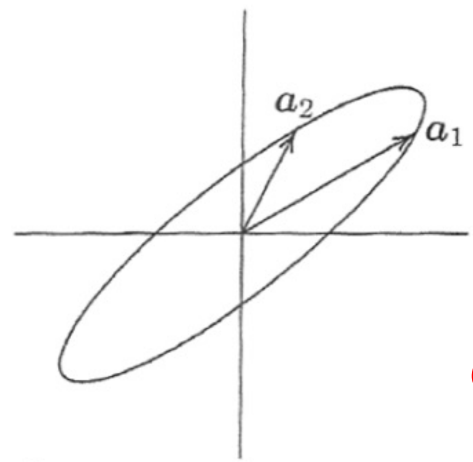
$$A_1 = \begin{bmatrix} 0.87 & 0.5 \\ -0.5 & 0.87 \end{bmatrix}, \quad \text{cond}_2(A_1) = 1$$



$$A_2 = \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad \text{cond}_2(A_2) = 4$$



$$A_3 = \begin{bmatrix} 1.73 & 0.25 \\ -1 & 0.43 \end{bmatrix}, \quad \text{cond}_2(A_3) = 4$$



$$A_4 = \begin{bmatrix} 1.52 & 0.91 \\ 0.47 & 0.94 \end{bmatrix}, \quad \text{cond}_2(A_4) = 4$$

condc.m

condc.m

```
hdr

A=[ 1.52 0.91 ;
    0.47 0.94 ];

theta = 2*pi*[0:1000]/1000;

x=cos(theta);
y=sin(theta);

X=[x ; y];

AX=A*X;

plot(x,y,'k-',lw,2,AX(1,:),AX(2,:),'r-',lw,2);
axis equal
legend('locus of {\bf x}','locus of {\bf Ax}',...
       'location','southeast')

cond_A = cond(A)


"condc.m" 37L, 292B written
```

Residual Vector

- What is the **residual vector** when solving $\mathbf{A}\mathbf{x} = \mathbf{b}$?

- **Answer**: It is the "remainder" that results from an inaccurate solution.
- Suppose the answer produced by our code is $\hat{\mathbf{x}}$.
- Then the residual vector is

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}} = -\mathbf{A}\,\Delta\mathbf{x}$$

Relationship between Residual and Error

- How does the norm of the residual vector $\mathbf{r}$ relate to the norm of the error $\Delta\mathbf{x}$?

- Consider

$$\|\Delta\mathbf{x}\| \;=\; \|\mathbf{x}-\hat{\mathbf{x}}\| \;=\; \|\mathbf{A}^{-1}(\mathbf{b}-\mathbf{A}\hat{\mathbf{x}})\| \;=\; \|\mathbf{A}^{-1}\mathbf{r}\|$$

- Divide both sides by $\|\mathbf{x}\|$:

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \;=\; \frac{\|\mathbf{A}^{-1}\mathbf{r}\|}{\|\mathbf{x}\|} \;\le\; \frac{\|\mathbf{A}^{-1}\|\,\|\mathbf{r}\|}{\|\mathbf{x}\|} \;=\; \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{A}\|\,\|\mathbf{x}\|} \;\le\; \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

- (relative error) $\le \kappa(\mathbf{A})$ (relative residual)

- Given small relative residual $\|\mathbf{r}\|/\|\mathbf{b}\|$, relative error is only (guaranteed to be) small if the condition number is also small.

Perturbations in the Matrix

- Matrix entries are also FP numbers and thus subject to round-off.
- How do changes in $\mathbf{A}$ influence the output, $\hat{\mathbf{x}}$?

$$\mathbf{A}\mathbf{x} \;=\; \mathbf{b} \longrightarrow \hat{\mathbf{A}}\,\hat{\mathbf{x}} \;=\; \mathbf{b}$$

- Consider

$$\Delta\mathbf{x} \;=\; \hat{\mathbf{x}} - \mathbf{x} \;=\; \mathbf{A}^{-1}\left(\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\right) \;=\; \mathbf{A}^{-1}\left(\mathbf{A}\hat{\mathbf{x}} - \hat{\mathbf{A}}\,\hat{\mathbf{x}}\right) \;=\; -\mathbf{A}^{-1}\,\Delta\mathbf{A}\,\hat{\mathbf{x}}$$

- Thus

$$\|\Delta\mathbf{x}\| \;\leq\; \|\mathbf{A}^{-1}\|\,\|\Delta\mathbf{A}\|\,\|\hat{\mathbf{x}}\|$$

and

$$\frac{\|\Delta\mathbf{x}\|}{\|\hat{\mathbf{x}}\|} \;\leq\; \kappa(A)\,\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}$$

Changing Condition Numbers

It is often possible to mitigate large condition numbers by *preconditioning*.

- Left preconditioning: $\mathbf{M}\,\mathbf{A}\mathbf{x} = \mathbf{M}\mathbf{b}$
- Right preconditioning: $\mathbf{A}\,\mathbf{M}\mathbf{y} = \mathbf{b}$, $\mathbf{x} = \mathbf{M}\mathbf{y}$

For example, can use a diagonal matrix $\mathbf{D}$ as a preconditioner

- Row-wise scaling: $\mathbf{D}\mathbf{A}\mathbf{x} = \mathbf{D}\mathbf{b}$
- Column-wise scaling: $\mathbf{A}\,\mathbf{D}\mathbf{y} = \mathbf{b}$, $\mathbf{x} = \mathbf{D}\mathbf{y}$

Orthogonal Matrices

What is an orthogonal (= orthonormal) matrix?

- An orthonormal matrix is a square matrix that satisfies $\mathbf{Q}^T\mathbf{Q} = I$ and $\mathbf{Q}\mathbf{Q}^T = I$
- Recall, if $\mathbf{Q} = [\mathbf{q}_1\ \mathbf{q}_2\ \cdots\ \mathbf{q}_n]$, then $\mathbf{Q}^T\mathbf{Q} = [\mathbf{q}_i^T\mathbf{q}_j] = \delta_{ij}$ (the Kronecker delta, $\delta_{ij} = 1$ if $i = j$, 0 otherwise)
- That is, *the columns of an orthonormal matrix* $\mathbf{Q}$ *are mutually orthogonal.*
- If $\mathbf{Q}$ is an orthogonal matrix, then $\mathbf{Q}^T$ is also orthogonal, so the *rows of an orthonormal matrix* $\mathbf{Q}$ *are also mutually orthogonal.*

Orthogonal Matrices and the 2-Norm

How do orthogonal matrices interact with the 2-norm?

$$\|\mathbf{Q}\mathbf{v}\|_2^2 \;=\; (\mathbf{Q}\mathbf{v})^T(\mathbf{Q}\mathbf{v}) \;=\; \mathbf{v}^T\mathbf{Q}^T\mathbf{Q}\mathbf{v} \;=\; \mathbf{v}^T\mathbf{v} = \|\mathbf{v}\|_2^2$$

Singular Value Decomposition (SVD)

The ***SVD*** of an $m \times n$ matrix $\mathbf{A}$ is given by the factorization

$$\mathbf{A} \;=\; \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

where

- $\mathbf{U}$ is $m \times m$ and orthogonal
  Columns $\mathbf{u}_j$ are called the *left singular vectors*

- $\boldsymbol{\Sigma}$ =diag(σ_i) is $m \times n$ and non-negative
  Typically $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s \geq 0$, with $s = \min(m, n)$.
  Diagonal entries σ_j are called the *singular values*

- $\mathbf{V}$ is $n \times n$ and orthogonal Columns $\mathbf{v}_j$ are called the *right singular vectors*

We'll discuss existance and computation later.

Computing the 2-Norm

Use the SVD of $\mathbf{A}$ to compute the 2-norm
$\mathbf{A} = \mathbf{U\Sigma V}^T$ with $\mathbf{U}$, $\mathbf{V}$ orthogonal

- 2-norm satisfies $\|\mathbf{QB}\|_2 = \|\mathbf{B}\|_2 = ||\mathbf{BQ}\|_2$ for any $\mathbf{B}$ and orthogonal $\mathbf{Q}$
- So $\|\mathbf{A}\|_2 = \|\mathbf{\Sigma}\|_2 = \sigma_{\max}$

We can express the matrix condition number, $\kappa_2(\mathbf{A})$ in terms of the SVD of $\mathbf{A}$

- $\mathbf{A}^{-1}$ has singular values $1/\sigma_j$
- $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 = \sigma_{\max}/\sigma_{\min}$

Frobenius Norm

- The 2-norm is costly to compute; *is there something cheaper?*

- The ***Frobenius norm***

$$\|\mathbf{A}\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

- $\|\mathbf{A}\|_F$ is ***not*** and induced norm.
- It does, however, satisfy the standar matrix-norm properties:
  - definiteness
  - scaling
  - triangle inequality
  - submultiplicativity (via Cauchy-Schwarz)

Frobenius Norm Properties

- Is the Frobenius norm induced by any vector norm?

  Not possible. What is $\|I\|_F$?

- How does the Frobenius norm relate to the SVD?

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n} \sigma_i^2}$$

Solving Systems: Simple Cases

- Solve $\mathbf{Dx} = \mathbf{b}$ if $\mathbf{D}$ is diagonal.

  $x_i = b_i/d_{ii}$ with cost $O(n)$

- Solve $\mathbf{Qx} = \mathbf{b}$ if $\mathbf{Q}$ is orthogonal

  $\mathbf{x} = \mathbf{Q}^T\mathbf{b}$ with cost $O(n^2)$

- Given SVD, $\mathbf{U\Sigma V}^T = \mathbf{A}$, solve $\mathbf{Ax} = \mathbf{b}$

  $\mathbf{z} = \mathbf{U}^T\mathbf{b}$

  $\mathbf{y} = \mathbf{\Sigma}^{-1}\mathbf{z}$

  $\mathbf{x} = \mathbf{Vy}$

  Cost: $O(n^2)$ to solve, $O(n^3)$ to compute SVD

Note on Row Scaling / Permutation

$$
\begin{aligned}
D\mathbf{v} &= \text{scale rows of } \mathbf{v} \\
P\mathbf{v} &= \text{permute rows of } \mathbf{v} \\
DA &= [\, D\mathbf{a}_1 \, D\mathbf{a}_2 \, \cdots \, D\mathbf{a}_n \,] = \text{scale rows of } A \\
PA &= [\, P\mathbf{a}_1 \, P\mathbf{a}_2 \, \cdots \, P\mathbf{a}_n \,] = \text{permute rows of } A
\end{aligned}
$$

$$
\begin{bmatrix} 2 & & \\ & 3 & \\ & & 4 \end{bmatrix}
\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}
=
\begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}
$$

$$
\begin{bmatrix} & & 1 \\ 1 & & \\ & 1 & \end{bmatrix}
\begin{bmatrix} 2 & 2 & 2 \\ 3 & 3 & 3 \\ 4 & 4 & 4 \end{bmatrix}
=
\begin{bmatrix} 4 & 4 & 4 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}
$$

Note on Column Scaling / Permutation

$$AD = [\, d_1\mathbf{a}_1 \, d_2\mathbf{a}_2 \, \cdots \, d_n\mathbf{a}_n \,] = \text{ scale columns of } A$$

$$AP = [\, \mathbf{a}_{p_1} \, \mathbf{a}_{p_2} \, \cdots \, \mathbf{a}_{p_n} \,] = \text{ permute columns of } A$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & & \\ & 3 & \\ & & 4 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \\ 2 & 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \\ 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} & 1 & \\ & & 1 \\ 1 & & \end{bmatrix} = \begin{bmatrix} 4 & 2 & 3 \\ 4 & 2 & 3 \\ 4 & 2 & 3 \end{bmatrix}$$

System Modification by Permutations

$$P\,A\,\mathbf{x} \;=\; P\,\mathbf{b} \qquad \text{Row Permutation}$$

$$\longrightarrow A'\,\mathbf{x} \;=\; \mathbf{b}'$$

$$A\,P\,P^T\mathbf{x} \;=\; \mathbf{b} \qquad \text{Column Permutation}$$

$$\longrightarrow A'\,\mathbf{x}' \;=\; \mathbf{b}$$

Solution of Lower Triangular Systems

$$\begin{bmatrix} l_{11} & & & & & \\ l_{21} & l_{22} & & & & \\ l_{31} & l_{32} & l_{33} & & & \\ \vdots & & & \ddots & & \\ \vdots & & & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} x_1 &= \frac{1}{l_{11}} \cdot b_1 \\ x_2 &= \frac{1}{l_{22}} \cdot [b_2 - l_{21}\, x_1] \\ x_3 &= \frac{1}{l_{33}} \cdot [b_3 - l_{31}\, x_1 - l_{32}\, x_2] \\ \vdots \quad & \quad \vdots \\ x_n &= \frac{1}{l_{nn}} \cdot [b_n - l_{n1} x_1 - \cdots - l_{n,n-1}\, x_{n-1}] \end{aligned}$$

Q: How could this go wrong?

Solution of Lower Triangular Systems

$$\begin{bmatrix} l_{11} & & & & & \\ l_{21} & l_{22} & & & & \\ l_{31} & l_{32} & l_{33} & & & \\ \vdots & & & \ddots & & \\ \vdots & & & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\text{for } i = 1, 2, \ldots, n: \quad x_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij}\, x_j \right).$$

for $i = 1 : n$

$\quad x_i = b_i$

$\quad$ for $j = 1 : i - 1$

$\quad\quad x_i = x_i - l_{ij}\, x_j$

$\quad$ end

$\quad x_i = x_i / l_{ii}$

end

Solution of Upper Triangular Systems

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & \cdots & u_{2n} \\ & & u_{33} & & & u_{33} \\ & & & \ddots & & \vdots \\ & & & & \ddots & \vdots \\ & & & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\begin{aligned} x_n &= \frac{1}{u_{n,n}} \cdot b_n \\ x_{n-1} &= \frac{1}{u_{n-1,n-1}} \cdot [b_{n-1} - u_{n-1,n}\, x_n] \\ x_{n-2} &= \frac{1}{u_{n-2,n-2}} \cdot [b_{n-1} - u_{n-2,n}\, x_n - u_{n-2,n-1}\, x_{n-1}] \\ \vdots \quad & \quad \vdots \\ x_1 &= \frac{1}{u_{1,1}} \cdot [b_1 - u_{1,n}\, x_n - \cdots - u_{1,2}\, x_2]\,. \end{aligned}$$

Solution of Upper Triangular Systems

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & \cdots & u_{2n} \\ & & u_{33} & & & u_{33} \\ & & & \ddots & & \vdots \\ & & & & \ddots & \vdots \\ & & & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\text{for } i = n, n-1, \ldots, 1 : \quad x_i = \frac{1}{u_{ii}} \left(b_i - \sum_{j=i+1}^{n} u_{ij}\, x_j \right).$$

for $i = n : 1$

  $x_i = b_i$

  for $j = i + 1 : n$

    $x_i = x_i - u_{ij}\, x_j$

  end

  $x_i = x_i / u_{ii}$

end

What is the cost ??

Solution of Upper Banded Systems

Suppose U is a *banded matrix*: $u_{ij} = 0,\ j > i + \beta$.

For example, $\beta = 2$:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & & & & \\ & u_{22} & u_{23} & u_{14} & & & \\ & & u_{33} & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \ddots & \ddots & u_{n-2,n} \\ & & & & & \ddots & u_{n-1,n} \\ & & & & & & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\text{for } i = n, n-1, \ldots, 1: \quad x_i = \frac{1}{u_{ii}} \left(b_i - \sum_{j=i+1}^{\min(i+\beta,n)} u_{ij}\, x_j \right).$$

What is the cost ??

Solution of Upper Banded Systems

$$\text{for } i = n, n-1, \ldots, 1: \quad x_i \;=\; \frac{1}{u_{ii}} \left(b_i \;-\; \sum_{j=i+1}^{\min(i+\beta,n)} u_{ij}\, x_j \right).$$

for $i = n : 1$

  $x_i \;=\; b_i, \;\; j_{\max} := \min(j+\beta, n)$

  for $j = i+1 : j_{\max}$

    $x_i \;=\; x_i \;-\; u_{ij}\, x_j$

  end

  $x_i \;=\; x_i / u_{ii}$

end

- In this case, there are $\sim 2\beta n$ operations and $\sim \beta n$ memory references (one for each u_{ij}).
- Often $\beta \ll n$, which means that the upper-banded system is *much* faster to solve than the full upper triangular system.
- The same savings applies to the lower-banded case.

START HERE, Lec 5

Generating Triangular Systems: LU Factorization

$$A = LU$$

Elimination

- To transform general linear system into upper triangular form, need to replace selected nonzero entries of matrix by zeros

- This can be accomplished by subtracting a multiple of "pivot row" from rows where zeros are desired

- Consider 2-vector $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$

- If $a_1 \neq 0$, then

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -a_2/a_1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ 0 \end{bmatrix}$$

Elimination

- Suppose we have a 3-vector $\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$

- If $a_1 \neq 0$, then

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -a_2/a_1 & 1 & 0 \\ -a_3/a_1 & 0 & 1 \end{bmatrix}}_{\mathbf{M}_1} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ 0 \\ 0 \end{bmatrix}$$

- We refer to $\mathbf{M}_1$ as an *elementary elimination matrix*

- It removes entries below row 1 in the prescribed vector

Elimination

- More generally, to eliminate all entries below kth row, $a_{k+1} \cdots a_n$, we would use a matrix of the form

$$\mathbf{M}_k = \mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -m_{k+1} & 1 & & \\ & & \vdots & & \ddots & \\ & & -m_n & & & 1 \end{bmatrix}$$

- Here, $\mathbf{e}_k = k$th column of the $n \times n$ identity matrix and

$$\mathbf{m}_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ m_{k+1} \\ \vdots \\ m_n \end{bmatrix},$$

with entries $m_i := a_i / a_k$, $i = k+1, \ldots, n$.

Elimination

- $\mathbf{M}_k$ is unit lower triangular and nonsingular

- $\mathbf{M}_k^{-1} = \mathbf{I} + \mathbf{m}_k \mathbf{e}_k^T$, which means $\mathbf{L}_k := \mathbf{M}_k^{-1}$ is same as $\mathbf{M}_k$ except signs of multipliers are reversed.

- If $j > k$, then

$$\begin{aligned} \mathbf{M}_k \mathbf{M}_j &= (\mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T)(\mathbf{I} - \mathbf{m}_j \mathbf{e}_j^T) \\ &= \mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T - \mathbf{m}_j \mathbf{e}_j^T + \mathbf{m}_k \mathbf{e}_k^T \mathbf{m}_j \mathbf{e}_j^T \\ &= \mathbf{I} - \mathbf{m}_k \mathbf{e}_k^T - \mathbf{m}_j \mathbf{e}_j^T \end{aligned}$$

  because $\mathbf{e}_k$ is orthogonal to $\mathbf{m}_j$ (the order, $j > k$, matters).

- The product, $\mathbf{M}_k \mathbf{M}_j$ is thus essentially the "union" of the entries, and similarly for the inverses, $\mathbf{L}_k \mathbf{L}_j$.

Example: Elementary Elimination Matrices

- For $\mathbf{a} = \begin{bmatrix} 2 \\ 4 \\ -6 \end{bmatrix}$

$$\mathbf{M}_1 \mathbf{a} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -6 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

and

$$\mathbf{M}_2 \mathbf{a} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 6/4 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -6 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}$$

Example, continued

- Note that

$$\mathbf{L}_1 := \mathbf{M}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}, \qquad \mathbf{L}_2 := \mathbf{M}_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3/2 & 1 \end{bmatrix}$$

and

$$\mathbf{M}_1\mathbf{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & 3/2 & 1 \end{bmatrix}, \qquad \mathbf{L}_1\mathbf{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -3/2 & 1 \end{bmatrix}$$

Gaussian Elimination as LU Factorization

- Consider the sequence of transformations

$$\begin{aligned}
\mathbf{A}_1 &:= \mathbf{M}_1\,\mathbf{A} && \textit{eliminate column 1 of } \mathbf{A} \\
\mathbf{A}_2 &:= \mathbf{M}_2\,\mathbf{A}_1 && \textit{eliminate column 2 of } \mathbf{A}_1 \\
&\vdots \\
\mathbf{A}_{n-1} &:= \mathbf{M}_{n-1}\,\mathbf{A}_{n-2} && \textit{eliminate column } n-1 \textit{ of } \mathbf{A}_{n-2} \\
&= \mathbf{M}_{n-1}\cdots\mathbf{M}_1\,\mathbf{A} \\
&= \mathbf{U} \quad \textit{upper triangular}
\end{aligned}$$

- Consequently,

$$\begin{aligned}
\mathbf{A} &= \mathbf{M}_1^{-1}\cdots\mathbf{M}_{n-2}^{-1}\,\mathbf{M}_{n-1}^{-1}\,\mathbf{U} \\
&= \underbrace{\mathbf{L}_1\cdots\mathbf{L}_{n-2}\,\mathbf{L}_{n-1}}_{L}\,\mathbf{U} \;=\; \mathbf{L}\,\mathbf{U}
\end{aligned}$$

- Our sequence of elementary elimination steps amounts to factoring $\mathbf{A}$ into a (nonsingular) unit lower triangular matrix $\mathbf{L}$ and a (possibly singular) upper triangular matrix $\mathbf{U}$

- Once we have the factorization $\mathbf{A} = \mathbf{LU}$, solve $\mathbf{Ax} = \mathbf{b}$ as $\mathbf{LUx} = \mathbf{b}$ by defining $\mathbf{y} = \mathbf{Ux}$ and
  - solving lower triangular system $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$ using forward substitution
  - solving upper triangular system $\mathbf{Ux} = \mathbf{y}$ using backward substitution

- An important concern when computing the $\mathbf{LU}$ factorization is if any pivot is 0 or *small*

- We will address this issue by swapping rows to find the largest (in absolute value) pivot in column k during the kth step of Gaussian elimination.

- Let's turn to some examples of how we implement $\mathbf{LU}$ factorization in practice

Gaussian Elimination - Main Steps

- The transformation of a general matrix to upper triangular form is known as *Gaussian Elimination* and it is equivalent to what is known as LU factorization.

- Equivalence-preserving operations used in Gaussian elimination include
  - row interchanges
  - column interchanges (relatively rare; used only for "full pivoting")
  - addition of a multiple of another row to a given row

  Notice that we do not include "multiplication of a row by a constant" because, while valid for any nonzero constant, it is generally not needed for Gaussian elimination.

- We have already seen how row/column interchanges can transform a system from lower-triangular form to upper-triangular form and can understand that reversing that procedure would take us back to our targeted upper-triangular form.

- Let's now look at the row-addition process for a more general example.

Generating Upper Triangular Systems: LU Factorization

- Example:

$$\begin{bmatrix} 1 & 2 & 3 & & \\ & 4 & 4 & 6 & 1 \\ & 8 & 8 & 9 & 2 \\ & 6 & 1 & 3 & 3 \\ & 4 & 2 & 8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

- First column is already in upper triangular form.
- Eliminate second column:

$$\begin{aligned} \text{row}_3 &\longleftarrow \text{row}_3 - \frac{8}{4} \times \text{row}_2 \\ \text{row}_4 &\longleftarrow \text{row}_4 - \frac{6}{4} \times \text{row}_2 \\ \text{row}_5 &\longleftarrow \text{row}_5 - \frac{4}{4} \times \text{row}_2 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & 3 & & \\ & 4 & 4 & 6 & 1 \\ & & 0 & -3 & 0 \\ & & -5 & -6 & \frac{3}{2} \\ & & -2 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ -4 \\ -2 \\ 0 \end{bmatrix}$$

- $a_{22} = 4$ is the *pivot*
- row_2 is the *pivot row*
- $l_{32} = \frac{8}{4}$, $l_{42} = \frac{6}{4}$, $l_{52} = \frac{4}{4}$, is the *multiplier column.* $= \dfrac{a_{ik}}{a_{kk}}, \quad i = k+1 \dots n$

Generating Upper Triangular Systems: LU Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:, n+1)$:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & 8 & 8 & 9 & 2 & 4 \\ & 6 & 1 & 3 & 3 & 4 \\ & 4 & 2 & 8 & 4 & 4 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

This Case.		***General Case.***
pivot	$= 4$	$= a_{kk}$ when zeroing the kth column.
pivot row	$= [\ 4\ \ 6\ \ 1 \mid 4\ ]$	$= \mathbf{r}_k^T = a_{kj},\ j = k+1, \dots, n\,[+b_k]$
multiplier column	$= \dfrac{1}{4}\begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$	$= \mathbf{c}_k = \dfrac{a_{ik}}{a_{kk}},\ i = k+1, \dots, n$
	$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$	

Generating Upper Triangular Systems: *LU* Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:, n+1)$:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & 8 & 8 & 9 & 2 & 4 \\ & 6 & 1 & 3 & 3 & 4 \\ & 4 & 2 & 8 & 4 & 4 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

	This Case.	***General Case.***
pivot	$= 4$	$= a_{kk}$ when zeroing the kth column.
pivot row	$= [\; 4 \;\; 6 \;\; 1 \mid 4 \;]$	$= \mathbf{r}_k^T = a_{kj},\; j = k+1, \dots, n\, [+ b_k]$
multiplier column	$= \dfrac{1}{4} \begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$	$= \mathbf{c}_k = \dfrac{a_{ik}}{a_{kk}},\; i = k+1, \dots, n$
	$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$	

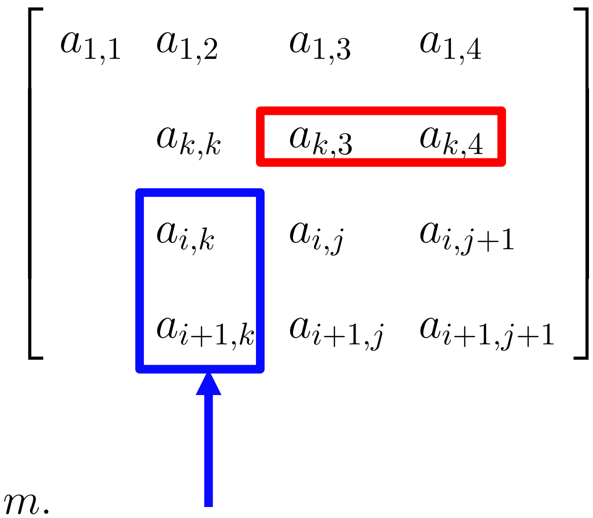
Generating Upper Triangular Systems: LU Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:, n+1)$:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & 8 & 8 & 9 & 2 & 4 \\ & 6 & 1 & 3 & 3 & 4 \\ & 4 & 2 & 8 & 4 & 4 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

This Case.		***General Case.***
pivot	$= 4$	$= a_{kk}$ when zeroing the kth column.
pivot row	$= [\ 4\ \ 6\ \ 1 \mid 4\ ]$	$= \mathbf{r}_k^T = a_{kj},\ j = k+1, \ldots, n\,[+b_k]$
multiplier column	$= \frac{1}{4}\begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$	$= \mathbf{c}_k = \dfrac{a_{ik}}{a_{kk}},\ i = k+1, \ldots, n$
	$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$	$\mathbf{c}_k \longrightarrow \mathbf{l}_k$, store as column k of L.

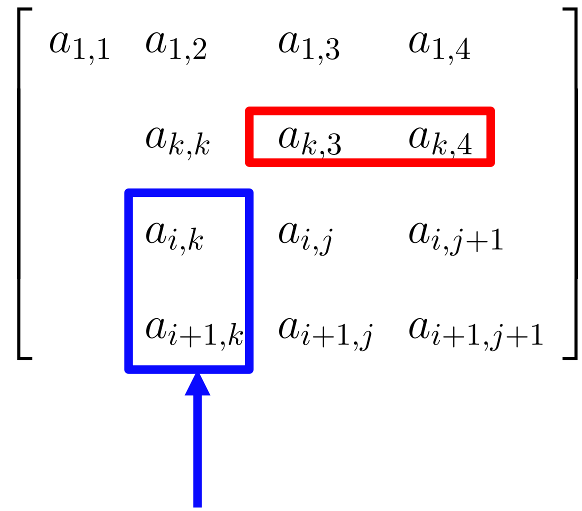
kth Update Step



- Look more closely at the kth update step for Gaussian elimination.
- Assume A is $m \times n$, which covers the case where A is augmented with the right-hand side vector.
- **Row k remains unchanged.**
- For each row i, with $i > k$, we want to generate a zero in place of a_{ik}.
- We do this by subtracting a multiple of row k from row i, $i = k + 1, \ldots, m$.

- This operation can be expressed in several equivalent ways:

$$
\begin{aligned}
\text{row}_i &= \text{row}_i - \frac{a_{ik}}{a_{kk}} \times \text{row}_k \\
a_{ij} &= a_{ij} - a_{ik}\, a_{kk}^{-1}\, a_{kj} \qquad j = k+1, \dots, n \\
&= a_{ij} - (\mathbf{c}_k)_i \left(\mathbf{r}_k^T\right)_j \qquad j = k+1, \dots, n \\
A^{(k+1)} &= A^{(k)} - \mathbf{c}_k\, \mathbf{r}_k^T,
\end{aligned}
$$



- Here, $\mathbf{c}_k$ is the column vector with entries $(\mathbf{c}_k)_i = a_{ik}/a_{kk}$, and $\mathbf{r}_k^T$ is the row vector with entries $\left(\mathbf{r}_k^T\right)_j = a_{kj}$.
- Formally, we think of $(\mathbf{c}_k)_i = 0$, $i \leq k$ and $\left(\mathbf{r}_k^T\right)_j = 0$, $j \leq k$, though we would implement as an update only to the active submatrix.
- The $m \times n$ matrix $\mathbf{c}_k\, \mathbf{r}_k^T$ is of rank 1. All columns are multiples of the only linearly independent column, $\mathbf{c}_k$.
- We typically save the entries of the multiplier column as the kth column of a lower triangular matrix: $l_{ik} := (\mathbf{c}_k)_i$.
- In fact, since the entries below a_{kk} in $A^{(k+1)}$ are zero, we can store the values of the multiplier column l_{ik} there.

Matlab: lu_demo_1.m

```
% Demo of outer-product-based LU factorization
format compact

U= [ 1 2 3 4 ;
     0 5 6 7 ;
     0 0 1 2 ;
     0 0 0 3 ]

L= [ 1 0 0 0 ;
     1 1 0 0 ;
     2 4 1 0 ;
     3 5 6 1 ]

A = L*U; [m,n]=size(A);

A, pause

v=[ ' | ';  ' | ';  ' | ';  ' | ' ];
for k=1:n-1; kp=k+1;
    r = A(k,k:m)';                        % Pivot Row
    c = A(kp:m,k)/A(k,k);                 % Multiplier Column
    A(kp:m,k:n) = A(kp:m,k:n) - c*r';     % Rank-1 Update
    disp([ num2str(A) v num2str(U) ]), pause
end;

%
%   COMPACT FORM
%

display('Compact form, with L U overwriting A')

A = L*U;
for k=1:n-1; kp=k+1;
    A(kp:m,k)    = A(kp:m,k)/A(k,k);                   %% Store l_k
    A(kp:m,kp:n) = A(kp:m,kp:n) - A(kp:m,k)*A(k,kp:m);
    disp([ num2str(A) v num2str(L) v num2str(U) ]), pause
end;

display('Compact form, with L U overwriting A')
A
```

Note: This demo does not use pivoting.

- ***For stability, we would invariably use partial pivoting because the computational overhead (cost, in terms of operations) is only O(n^2), where as the total factor cost is ~ 2/3 n^3***

Using LU Factorization in Practice

- Give $LU = A$, we can solve $A\mathbf{x} = \mathbf{b}$ as follows:

$$
\begin{aligned}
\text{Given: } A\mathbf{x} &= L\,U\,\mathbf{x} = \mathbf{b} \\
L\,(U\,\mathbf{x}) &= L\mathbf{y} = \mathbf{b}
\end{aligned}
$$

$$
\begin{aligned}
\text{Solve: } L\mathbf{y} &= \mathbf{b} \\
U\mathbf{x} &= \mathbf{y}
\end{aligned}
$$

- We have seen already that the total solve cost (for L and U solves) is $2 \times n^2$.
- What about the factor cost, $A \longrightarrow LU$?

LU Factorization Costs (Important)

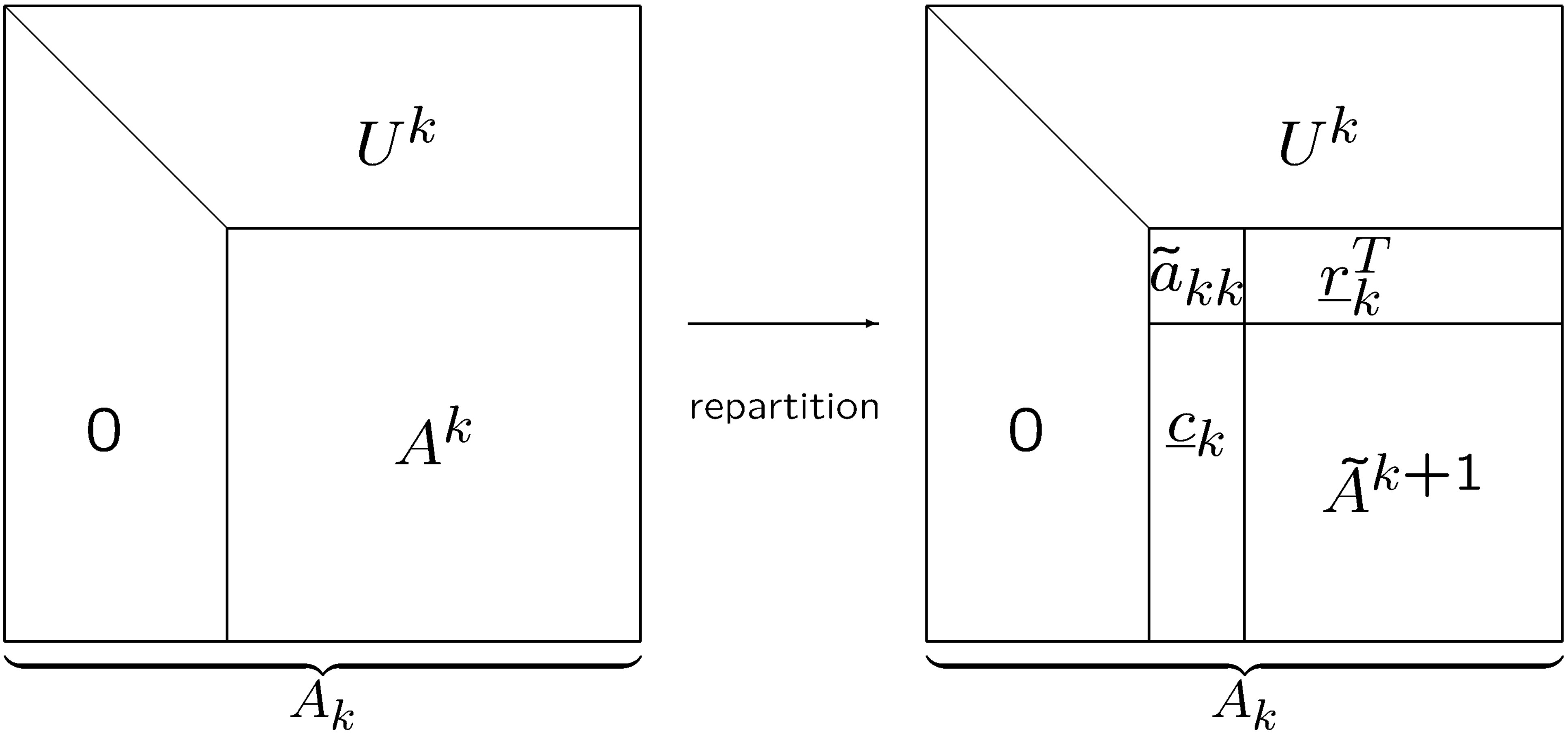
- In general, the cost for $A \longrightarrow LU$ is $O(n^3)$.
- It is large (i.e., it is not optimal, which would be $O(n)$), and therefore important.
- The dominant cost comes from the essential update step:

$$A^{(k+1)} \quad = \quad A^{(k)} \; - \; \mathbf{c}_k \, \mathbf{r}_k^T ,$$

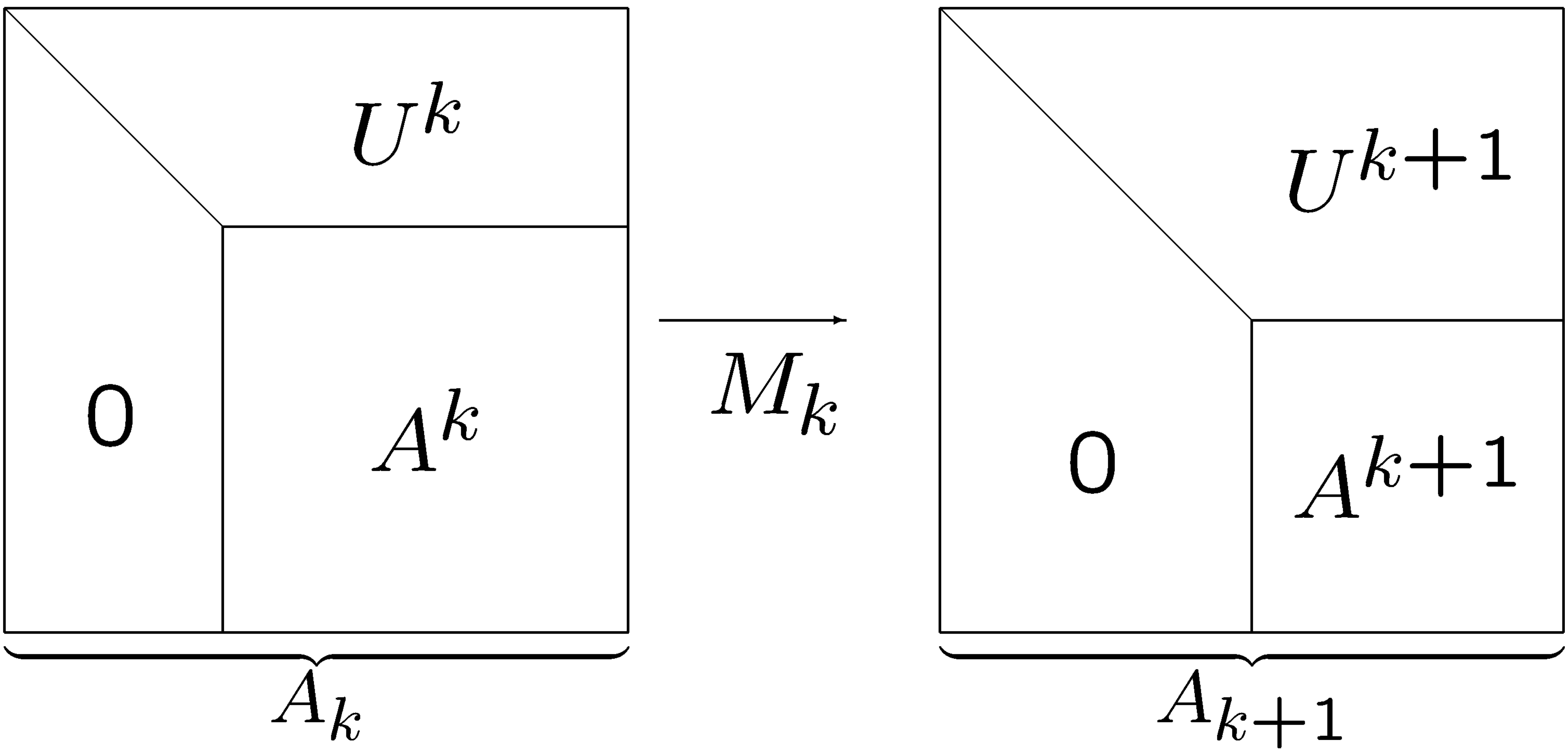
  which is effected for $k = 1, \ldots, n-1$ steps.

- If A is square ($n \times n$), then $\mathbf{c}_k \, \mathbf{r}_k^T$ is a square matrix with $(n-k)^2$ nonzeros.
- Each entry requires one "*" and its subtraction from $A^{(k)}$ requires one "-".
- Total cost is $2 \times \left[\, (n-1)^2 \; + \; (n-2)^2 \; + \; \ldots (1)^2 \right] \; \sim \; 2n^3/3$ operations.
- **Example:** $n = 10^3 \; \longrightarrow n^3 = 10^9$. Cost is about 0.6 billion operations.
  With a 3 GHz clock and 2 floating point ops / clock, expect about 0.1 seconds (very fast).
- **Example:** $n = 10^4 \; \longrightarrow n^3 = 10^{12}$. Cost is about 600 billion operations.
  With a 3 GHz clock and 2 floating point ops / clock, expect about 10.0 seconds.

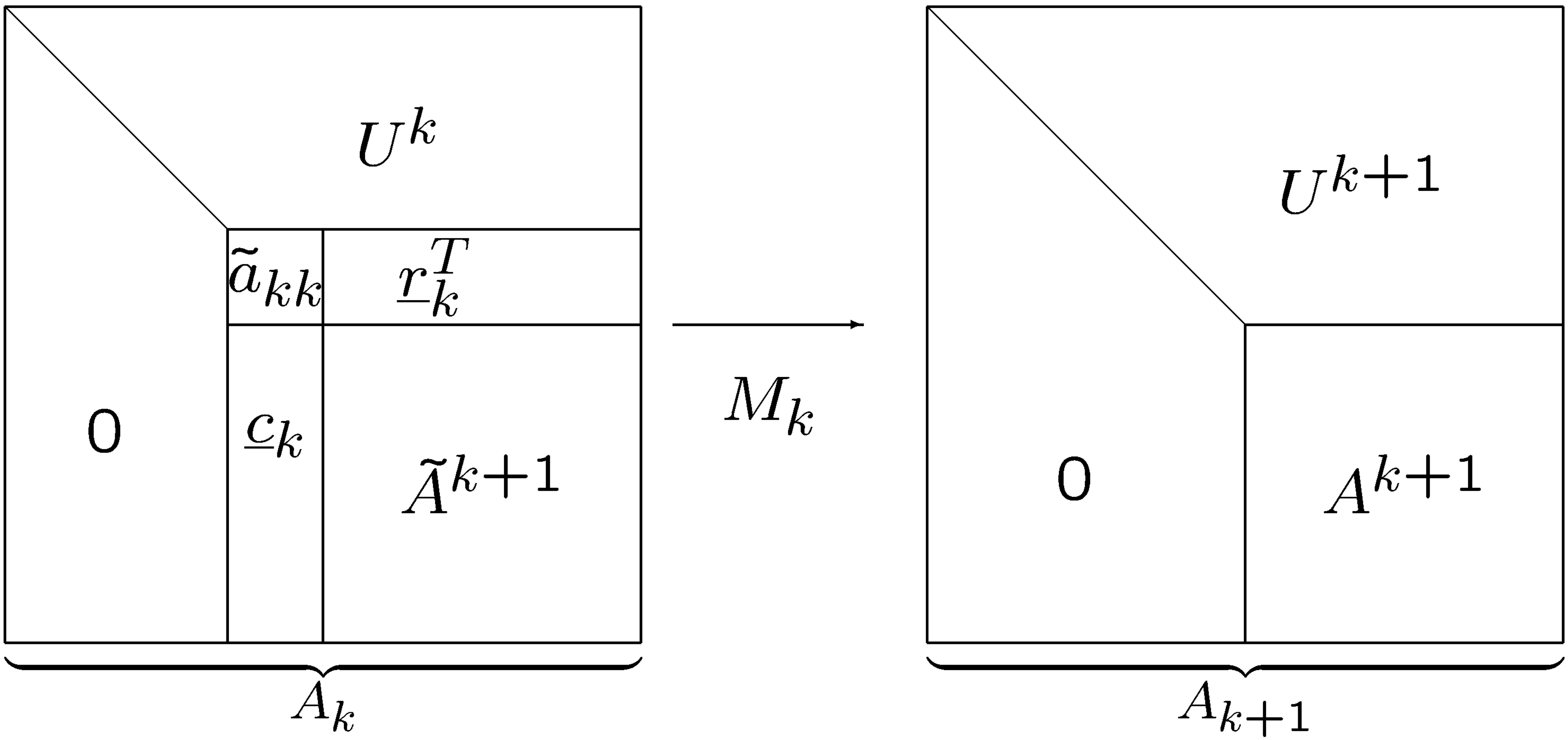
First Step: Define sub-block



Single Gaussian Elimination Step



Second Step: Annihilate $\underline{c}_k$



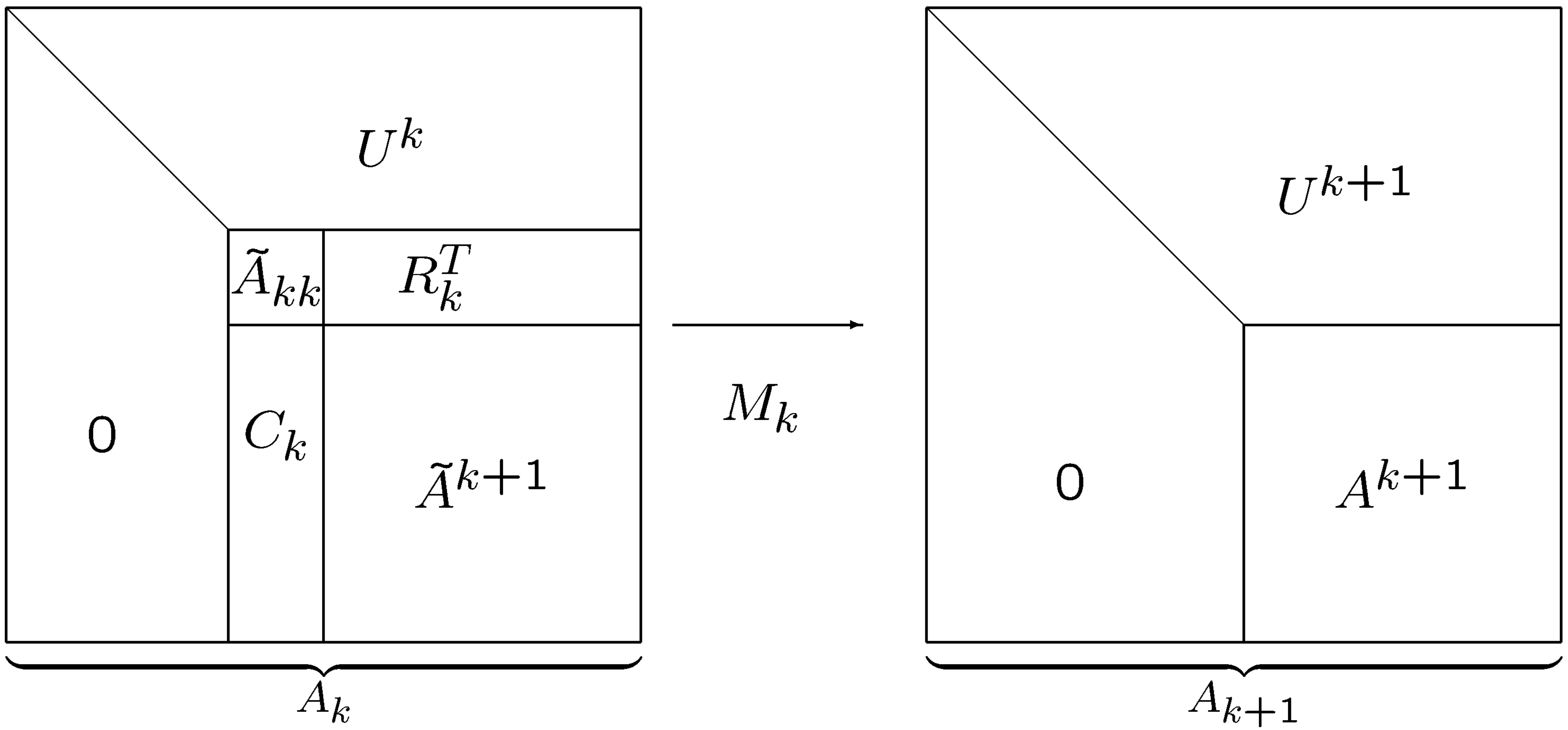
- Update step is:

$$A^{k+1} = \tilde{A}^{k+1} - \underline{c}_k \tilde{a}_{kk}^{-1} \underline{r}_k^T$$

which is a rank one update to A_k:

$$A_{k+1} = A_k - \underline{m}_k \underline{e}_k^T A_k$$

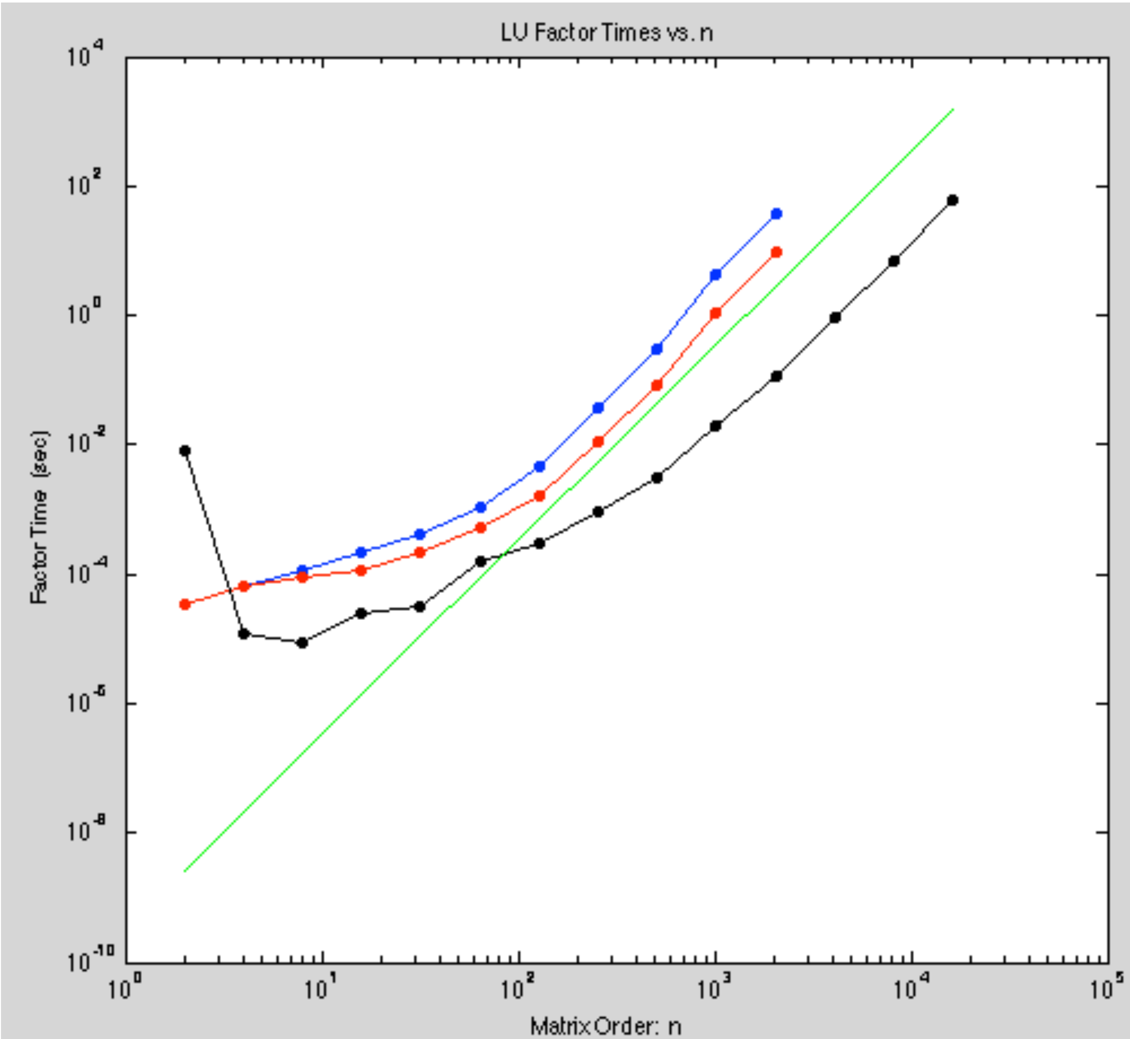
Can also be Implemented in ***Block Form***



$$A^{k+1} = \tilde{A}^{k+1} - C_k \tilde{A}_{kk}^{-1} R_k^T$$

- Advantage is that, if A_{kk} is a b x b block, you revisit the A^k sub-block only n/b times, and thus need fewer memory accesses. ***An order-of-magnitude faster.*** (LAPACK vs. LINPACK)

Matlab demo, gauss2.m



- Blue curve is rank-1 update
- Red curve is rank-4 update
- Black curve is matlab lu() function
  - It uses a 4 CPUs on the Mac and achieves an impressive 50 Gflops, which is very near peak

- Note that the black curve represents a ***~100X*** speed up over a naïve rank-1 update approach. (Why?)

Next Topics

- Pivoting / zeros & stability
  - Approach
  - Permutation Matrices
  - Stability
  - Cost
- Sherman Morrison
- Computing matrix 2-norm
- SPD / Cholesky Factorization
- Banded Factorization
  - Approach
  - Cost

Recall our earlier example:

$$\begin{bmatrix} 1 & 2 & 3 & & \\ & 4 & 4 & 6 & 1 \\ & 8 & 8 & 9 & 2 \\ & 6 & 1 & 3 & 3 \\ & 4 & 2 & 8 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

- First column is already in upper triangular form.
- Eliminate second column:

$$\begin{aligned} \text{row}_3 &\longleftarrow \text{row}_3 - \frac{8}{4} \times \text{row}_2 \\ \text{row}_4 &\longleftarrow \text{row}_4 - \frac{6}{4} \times \text{row}_2 \\ \text{row}_5 &\longleftarrow \text{row}_5 - \frac{4}{4} \times \text{row}_2 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & 3 & & \\ & 4 & 4 & 6 & 1 \\ & & 0 & -3 & 0 \\ & & -5 & -6 & \frac{3}{2} \\ & & -2 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ -4 \\ -2 \\ 0 \end{bmatrix}$$

- $a_{22} = 4$ is the *pivot*
- row_2 is the *pivot row*
- $l_{32} = \frac{8}{4}$, $l_{42} = \frac{6}{4}$, $l_{52} = \frac{4}{4}$, is the *multiplier column.*

Generating Upper Triangular Systems: LU Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:,n+1)$:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & 8 & 8 & 9 & 2 & 4 \\ & 6 & 1 & 3 & 3 & 4 \\ & 4 & 2 & 8 & 4 & 4 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

This Case.		***General Case.***
pivot	$= 4$	$= a_{kk}$ when zeroing the kth column.
pivot row	$= [\ 4\ \ 6\ \ 1 \mid 4\ ]$	$= \mathbf{r}_k^T = a_{kj},\ j = k+1,\ldots,n\,[+b_k\,]$
multiplier column	$= \frac{1}{4}\begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$	$= \mathbf{c}_k = \dfrac{a_{ik}}{a_{kk}},\ i = k+1,\ldots,n$
	$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$	

Generating Upper Triangular Systems: LU Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:, n+1)$:

$$
\left[\begin{array}{ccccc|c}
1 & 2 & 3 & & & 0 \\
 & 4 & 4 & 6 & 1 & 4 \\
 & 8 & 8 & 9 & 2 & 4 \\
 & 6 & 1 & 3 & 3 & 4 \\
 & 4 & 2 & 8 & 4 & 4
\end{array}\right]
\longrightarrow
\left[\begin{array}{ccccc|c}
1 & 2 & 3 & & & 0 \\
 & 4 & 4 & 6 & 1 & 4 \\
 & & 0 & -3 & 0 & -4 \\
 & & -5 & -6 & \frac{3}{2} & -2 \\
 & & -2 & 2 & 3 & 0
\end{array}\right]
$$

This Case.		***General Case.***
pivot	$= 4$	$= a_{kk}$ when zeroing the kth column.
pivot row	$= [\; 4 \;\; 6 \;\; 1 \mid 4 \;]$	$= \mathbf{r}_k^T = a_{kj},\ j = k+1, \ldots, n\,[+b_k]$
multiplier column	$= \frac{1}{4}\begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix}$	$= \mathbf{c}_k = \frac{a_{ik}}{a_{kk}},\ i = k+1, \ldots, n$
	$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$	

Generating Upper Triangular Systems: LU Factorization

- Augmented form. Store $\mathbf{b}$ in $A(:, n+1)$:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & 8 & 8 & 9 & 2 & 4 \\ & 6 & 1 & 3 & 3 & 4 \\ & 4 & 2 & 8 & 4 & 4 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

This Case. ***General Case.***

$$\text{pivot} = 4 \qquad = a_{kk} \text{ when zeroing the } k\text{th column.}$$

$$\text{pivot row} = [\ 4\ \ 6\ \ 1 \mid 4\ ] \qquad = \mathbf{r}_k^T = a_{kj},\ j = k+1, \dots, n\,[+b_k]$$

$$\text{multiplier column} = \frac{1}{4}\begin{bmatrix} 8 \\ 6 \\ 4 \end{bmatrix} \qquad = \mathbf{c}_k = \frac{a_{ik}}{a_{kk}},\ i = k+1, \dots, n$$

$$= \begin{bmatrix} 2 \\ \frac{3}{2} \\ 1 \end{bmatrix}$$

$\mathbf{c}_k \longrightarrow \mathbf{l}_k$, store as column k of L.

Pivoting

- We return to our original 5×5 example. The next step would be:

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & 0 & -3 & 0 & -4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & -2 & 2 & 3 & 0 \end{array}\right]$$

- Here, we have diffiulty because the nominal pivot, a_{33} is zero.
- The remedy is to exchange rows with one of the remaining two, since the order of the equations is immaterial.
- For numerical stability, we choose the row that maximizes $|a_{ik}|$.
- This choice ensures that all entries in the multiplier column are less than one in modulus.

Next Step: $k = k + 1$

- After switching rows, we have

$$\left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & 0 & -3 & 0 & -4 \\ & & -2 & 2 & 3 & 0 \end{array}\right] \longrightarrow \left[\begin{array}{ccccc|c} 1 & 2 & 3 & & & 0 \\ & 4 & 4 & 6 & 1 & 4 \\ & & -5 & -6 & \frac{3}{2} & -2 \\ & & 0 & -3 & 0 & -4 \\ & & 0 & 4\frac{2}{5} & 2\frac{2}{5} & \frac{4}{5} \end{array}\right]$$

$$\begin{aligned} \text{pivot} &= -5 \\ \text{pivot row} &= \left[\begin{array}{cc|c} -6 & \dfrac{3}{2} & -2 \end{array}\right] \\ \text{multiplier column} &= \frac{1}{-5}\left[\begin{array}{c} 0 \\ -2 \end{array}\right] \end{aligned}$$

Pivoting:

- Moving small pivots down moves us closer to upper triangular form, with ***no round-off.***

$$PA = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$$

- A general principle in numerical computing regarding round-off:
- ***Small corrections are preferred to large ones.***

- Failure to exchange a small pivot on the diagonal can result in all subsequent rows looking like multiples of the current pivot row → ***singular submatrix.***

Failure to pivot can result in all subsequent rows looking like multiples of the kth row:

- Consider

$$A = \begin{pmatrix} \epsilon & \text{---}\,\underline{r}_1^T\,\text{---} \\ a_{21} & \text{---}\,\underline{r}_2^T\,\text{---} \\ a_{31} & \text{---}\,\underline{r}_3^T\,\text{---} \\ \vdots & \text{---}\;\vdots\;\text{---} \end{pmatrix}$$

Gaussian elimination leads to

$$\underline{r}_i \;\longleftarrow\; \underline{r}_i - \frac{a_{i1}}{\epsilon}\underline{r}_1 \approx -\frac{a_{i1}}{\epsilon}\underline{r}_1 .$$

- Matlab example “pivot_off.m” , etc.

pivot_partial.m

1.0e-18	1.0000	2.0000	3.0000	4.0000
1.0000	4.0000	4.0000	6.0000	1.0000
2.0000	8.0000	7.0000	9.0000	2.0000
3.0000	6.0000	1.0000	3.0000	3.0000
4.0000	4.0000	2.0000	8.0000	4.0000

Failure to Pivot, Noncatastrophic Case

- In cases where the nominal pivot is small but > ε_M, we are effectively reducing the number of significant digits that represent the remainder of the matrix A.

- In essence, we are driving the rows (or columns) to be ***similar***, which is equivalent to saying that we have nearly parallel columns.

- We saw already a 2 x 2 example where the condition number of the matrix with 2 unit-norm vectors scales like 2 / μ , where μ is the (small) angle between the column vectors.

LU Factorization with Patial Pivoting

- With partial pivoting, each $\mathbf{M}_k$ is preceded by a permutation, $\mathbf{P}_k$ to interchange rows to bring entry with of largest magnitude into diagonal pivot position.

- Still obtain $\mathbf{MA} = \mathbf{U}$ with $\mathbf{U}$ upper triangular, but now,

$$\mathbf{M} \;=\; \mathbf{M}_{n-1}\,\mathbf{P}_{n-1}\,\cdots\,\mathbf{M}_1\,\mathbf{P}_1$$

- $\mathbf{L} = \mathbf{M}^{-1}$ is still triangular in a general sense, but not necessarily *lower triangular*

- Alternatively, can write

$$\mathbf{P}\,\mathbf{A} \;=\; \mathbf{L}\,\mathbf{U}$$

  where $\mathbf{P} \;=\; \mathbf{P}_{n-1}\,\cdots\,\mathbf{P}_1$ permutes rows of $\mathbf{A}$ into order determined by partial pivoting and now $\mathbf{L}$ is lower triangular

- "tlu.m" demo

Partial Pivoting: Costs

Procedure:

- For each k, pick k' such that $|a_{k'k}| \geq |a_{ik}|$, $i \geq k$.
- Swap rows k and k'.
- Proceed with central update step: $A^{(k+1)} \;=\; A^{(k)} \;-\; \mathbf{c}_k\,\mathbf{r}_k^T$

Costs:

- For each step, search is $O(n-k)$, total cost is $\approx n^2/2$.
- For each step, row swap is $O(n-k)$, total cost is $\approx n^2/2$.
- Total cost for partial pivoting is $O(n^2) \ll 2n^3/3$.
- If we use *full pivoting*, total search cost such that $|a_{k'k''}| \geq |a_{ij}|$, $i, j \geq k$, is $O(n^3)$.
- Row and column exchange costs still total only $O(n^2)$.

Notes:

- Partial (row) pivoting ensures that multiplier column entries have modulus ≤ 1. (Good.)
- For *banded matrices* full pivoting also destroys band structure, whereas partial pivoting leaves some band structure intact. (Matrix bandwith increases by at most $2\times$.)

Partial Pivoting: LU=PA

- Note: If we swap rows of A, we are swapping equations. $\longrightarrow$ Must swap rows of $\mathbf{b}$.
- LU routines normally return the pivot index vector to effect this exchange.
- Nominally, it looks like a permutation matrix P, which is simply the identity matrix with rows interchanged.
- If we swap equations, we must also swap rows of L
- If we are consistent, we can swap rows at any time (i.e., A, or L) and get the same final factorization: $LU = PA$.

- Swapping rows of $A^{(k+1)}$ helps with speed (vectorization) of $A^{(k+1)} = A^{(k)} - \mathbf{c}_k \mathbf{r}_k^T$.
- In parallel computing, one would *not* swap the pivot row between processors.
  Just pass the pointer to the processor holding the new pivot row, where the swap would take place locally.

Remaining Topics

- Condition Number Example
- Special Matrices
- SPD / Cholesky Factorization
- Sherman Morrison

Condition Number and Relative Error: $A\mathbf{x} = \mathbf{b}$

- Want to solve $A\mathbf{x} = \mathbf{b}$, but computed rhs is:

$$\mathbf{b}' \;=\; \mathbf{b} \,+\, \Delta\mathbf{b},$$

  where we anticipate

$$\frac{||\Delta\mathbf{b}||}{||\mathbf{b}||} \;\lesssim\; \epsilon_M.$$

- Net result is we end up solving $A\mathbf{x}' = \mathbf{b}'$ and want to know how large is the relative error in $\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}$,

$$\frac{||\Delta\mathbf{x}||}{||\mathbf{x}||}?$$

- Since $A\mathbf{x}' = \mathbf{b}'$ and (by definition) $A\mathbf{x} = \mathbf{b}$, we have $A\Delta\mathbf{x} = \Delta\mathbf{b}$ and thus,

$$\begin{aligned}
||\Delta\mathbf{x}|| &\leq ||A^{-1}||\; ||\Delta\mathbf{b}|| \\
||\mathbf{b}|| &\leq ||A||\; ||\mathbf{x}|| \\
\frac{1}{||\mathbf{x}||} &\leq ||A||\; \frac{1}{||\mathbf{b}||} \\
\frac{\Delta\mathbf{x}}{||\mathbf{x}||} &\leq ||A||\; \frac{\Delta\mathbf{x}}{||\mathbf{b}||} \\
&\leq ||A||\; ||A^{-1}||\; \frac{\Delta\mathbf{b}}{||\mathbf{b}||} \;=\; \mathrm{cond}(A)\; \frac{\Delta\mathbf{b}}{||\mathbf{b}||}.
\end{aligned}$$

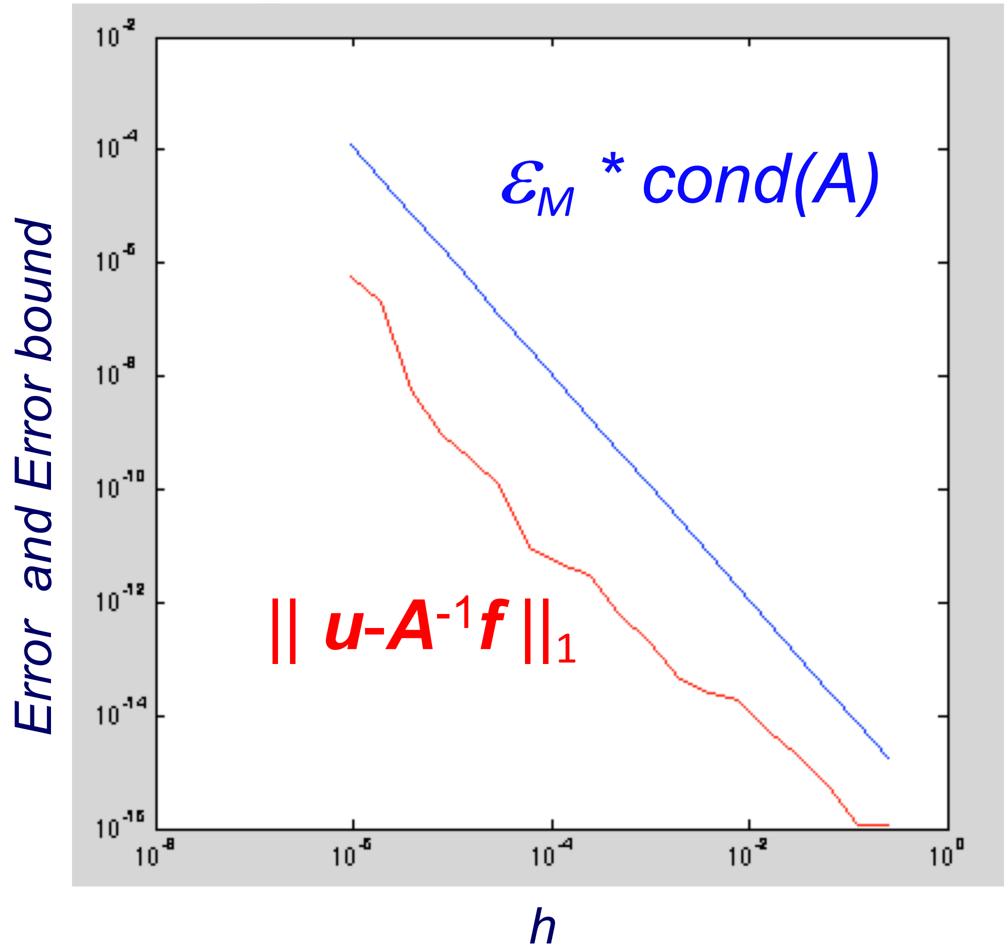
- Key point: If cond(A)=10^k, then expected relative error is $\approx 10^k \epsilon_M$, meaning that you will lose k digits (of 16, if $\epsilon_M \approx 10^{-16}$.

- A similar analysis and result holds when the entries *of* A are perturbed.

Illustration of Impact of cond(A)

```
%% Check the error in solving Au=f vs eps*cond(A).

%% Test problem is finite difference solution to -u" = f
%% on [0,1] with u(0)=u(1)=0.

for k=2:20; n = (2^k)-1; h=1/(n+1);

  e = ones(n,1);
  A = spdiags([-e 2*e -e],-1:1, n,n)/(h*h);
  x=1:n; x=h*x';
  ue=1+sin(pi*(8*x.*x));

  f=A*ue;
  u=A\f;

  hk(k)=h; ck(k)=cond(A);
  ek(k)=max(abs(u-ue))/max(ue);
end;
loglog(hk,ek,'r-',hk,eps*ck,'b-');
axis square
```

*Here, we see that ε_M * cond(A) bounds the error in the solution to Au=f, as expected.*



- If A is symmetric-positive definite (SPD), $\text{cond}(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$

- There are many matrices where we have good estimates for the condition number.

- For example, the tridiagonal matrix below arises in many boundary-value problems and has a condition number $\text{cond}(A) \sim \frac{4n^2}{\pi^2}$.

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}.$$

- The condition number can also be estimated at low cost when solving a linear system $A\mathbf{x} = \mathbf{b}$ using Gaussian elimination.

Some Special Matrices

- Diagonally dominant
- Symmetric Positive Definite (SPD)
- Banded ($a_{ij} = 0$ for $|i - j| > b$)
- Sparse (number of nonzeros per row bounded, independent of n)

Matrices that do not Require Pivoting

- ***Diagonally dominant***:

$$\sum_{i \neq j} |a_{ij}| \;\leq\; |a_{jj}|,\; j = 1, \ldots, n$$

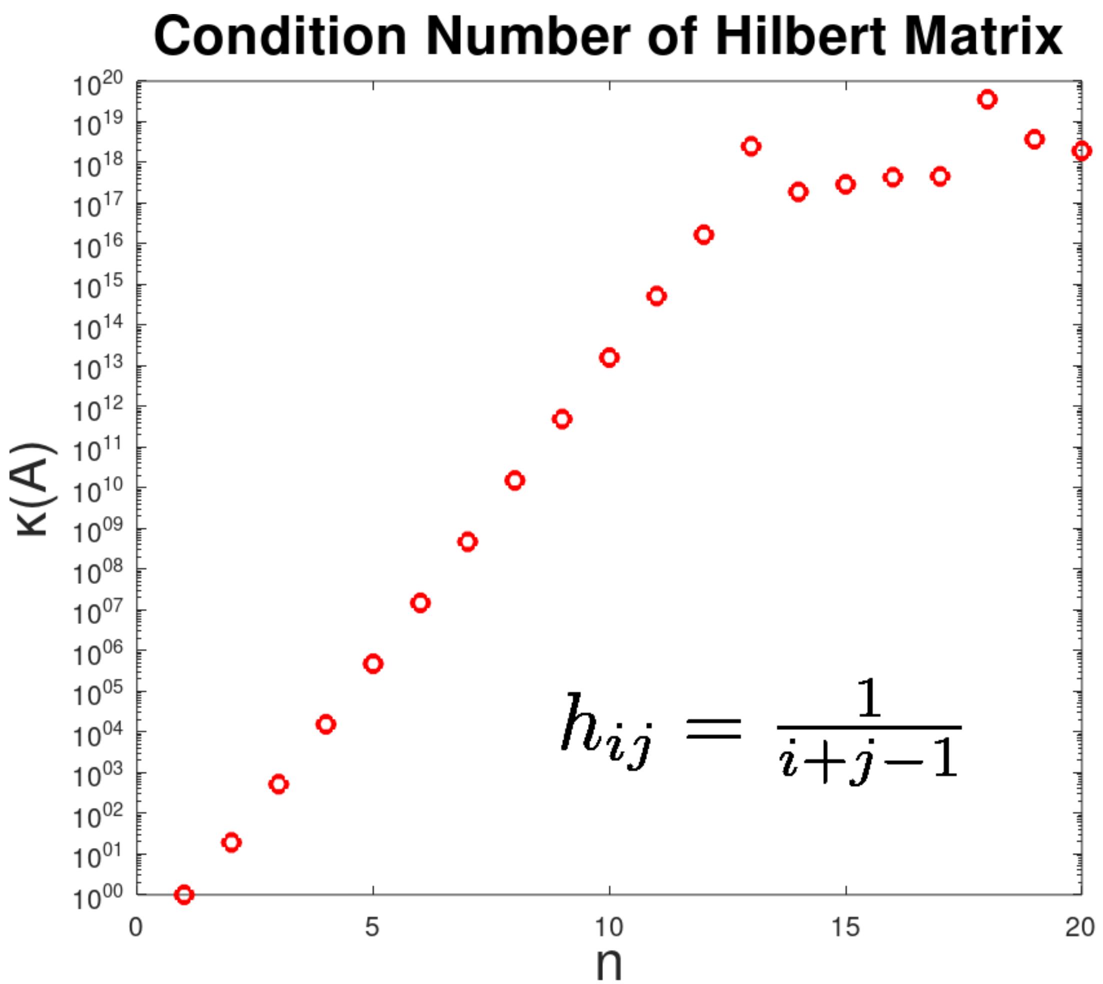
- ***Symmetric positive definite (SPD)***:

$$\mathbf{A} \;=\; \mathbf{A}^T \;\text{ and }\; \mathbf{x}^T\mathbf{A}\mathbf{x} \;>\; 0 \text{ for all } \mathbf{x} \neq 0$$

- Some consequences of $\mathbf{A}$ being SPD:
  - Diagonal entries, $a_{ii} > 0$, $i = 1, \ldots, n$
  - Eigenvalues, $\lambda_i > 0$, $i = 1, \ldots, n$
  - Linear systems can be solved with *Cholesky factorization* ("direct" method) or, in the case of a sparse SPD system, *conjugate gradients* ("iterative" method)
  - Being SPD does ***not***, however, imply that $\mathbf{A}$ is well-conditioned. (hilbert.m demo)

Condition Number of Hilbert Matrix

- The Hilbert matrix, $\mathbf{H} = h_{ij} = \frac{1}{i+j-1}$ is SPD
- It is notoriously *ill-conditioned*, however, with $\kappa(\mathbf{H})$ growing exponentially with n




```
hdr;  % Define fs=fontsize, etc.
hold off;

for n=1:20
   A=eye(n);

   for j=1:n;
   for i=1:n;
       A(i,j) = 1./(i+j-1);
   end;
   end;
   c=cond(A,2)

   semilogy(n,c,'ro',lw,2); hold on;
   xlabel('n',fs,20); ylabel('\kappa(A)',fs,20);
   title('Condition Number of Hilbert Matrix',fs,20);
   text(9,10000,'$h_{ij} = \frac{1}{i+j-1}$',intp,ltx,fs,30);

end;
```

Example of SPD Matrix

- If $\mathbf{B}$ is invertible, then $\mathbf{A} = \mathbf{B}^T\mathbf{B}$ is SPD.

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \;=\; \mathbf{x}^T\mathbf{B}^T\mathbf{B}\mathbf{x} \;=\; (\mathbf{B}\mathbf{x})^T\mathbf{B}\mathbf{x} \;=\; \mathbf{y}^T\mathbf{y} \;= \|\mathbf{y}\|_2^2 \;>\; 0$$

- The expression $\mathbf{y} = \mathbf{B}\mathbf{y}$ can only be singular for nonzero $\mathbf{x}$ if $\mathbf{B}$ is singular.

Cholesky Factorization

- If $\mathbf{A}$ is SPD then LU factorization can be arranged so that $U = L^T$ (for $\mathbf{L}$ not *unit* lower triangular)

- This gives the ***Cholesky factorization***

$$\mathbf{A} \;=\; \mathbf{L}\mathbf{L}^T$$

  where $\mathbf{L}$ is lower triangular with posivite diagonal entries

- Algorithm for computing it can be derived by equating corresponding entries of $\mathbf{A}$ and $\mathbf{L}\mathbf{L}^T$

- In 2×2 case, for example,

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

  implies

$$l_{11} = \sqrt{a_{11}} \qquad l_{21} = a_{21}/l_{11} \qquad l_{22} = \sqrt{a_{22} - l_{21}^2}$$

Cholesky Factorization

- One way to write the algorithm, with Cholesky factor $\mathbf{L}$ overwriting lower triangle of $\mathbf{A}$, is

$$
\begin{array}{ll}
\textit{for} \quad k = 1 \ \textit{to} \ n & (\textit{loop over columns}) \\
\qquad a_{kk} \ = \ \sqrt{a_{kk}} & \\
\qquad \textit{for} \ i = k+1 \ \textit{to} \ n & \\
\qquad\qquad a_{ik} \ = \ a_{ik}/a_{kk} & (\textit{scale current column}) \\
\qquad \textit{end} & \\
\qquad \textit{for} \ j = k+1 \ \textit{to} \ n & \\
\qquad \textit{for} \ i = j \ \textit{to} \ n & \\
\qquad\qquad a_{ij} \ = \ a_{ij} - a_{ik} \cdot a_{jk} & (\textit{rank-1 update}) \\
\qquad \textit{end} & \\
\qquad \textit{end} & \\
\textit{end} &
\end{array}
$$

Cholesky Factorization, continued

- Features of Cholesky factorization
  - Requires that $\mathbf{A}$ be SPD
  - All n square roots are positive $\longrightarrow$ algorithm is well defined
  - No pivoting required to maintain numerical stability
  - Only lower triangular part of $\mathbf{A}$ is accessed, so only 1/2 the storage is required
  - Only $n^3/6$ multiplications and additions required, so 1/2 the work
- Cholesky requires about half the work and half the storage of LU and avoids the need for pivoting.

Band Matrices

- $a_{ij} = 0$ for $|j - i| > b$

- Gaussian elimination for band matrices differs little from general case–only loop ranges change

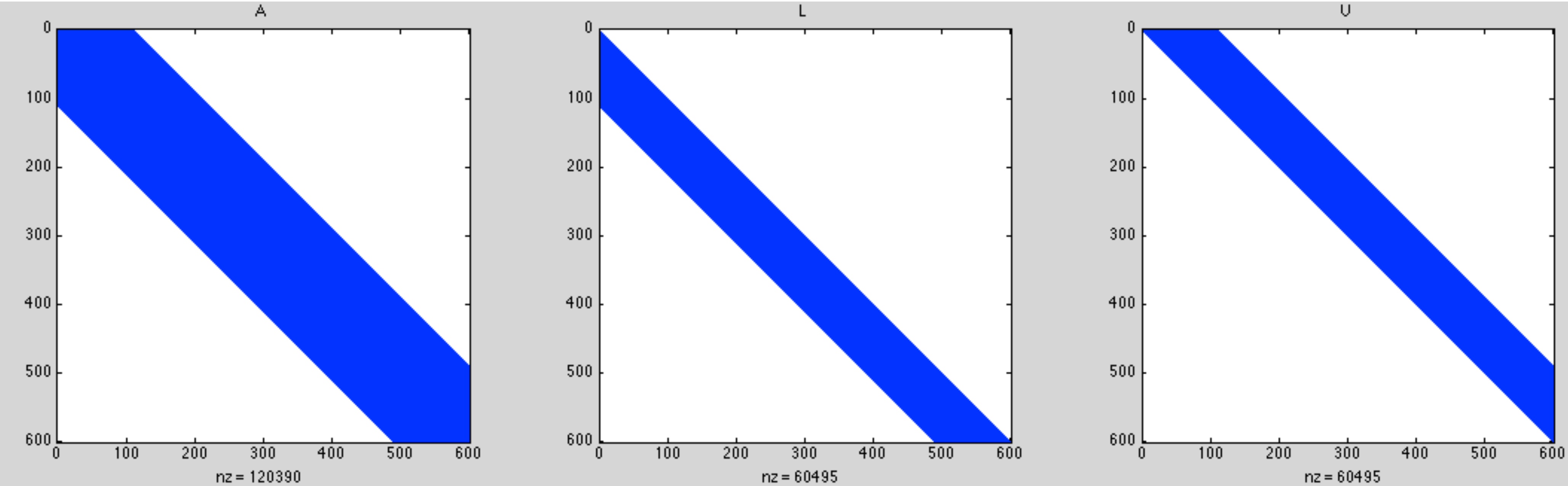
- Typically matrix is stored in array by diagonals to avoid storing zero entries

- If pivoting is required for numerical stability, bandwidth can grow (but no more than double)

- General purpose solver for arbitrary bandwidth is similar to code for Gaussian elimination for general matrices

- For fixed small bandwidth, band solver can be extremely simple, especially if pivoting is not required for stability

Band Matrices



- Significant savings in storage and work if A is banded → $a_{ij} = 0$ if $|i-j| > b$

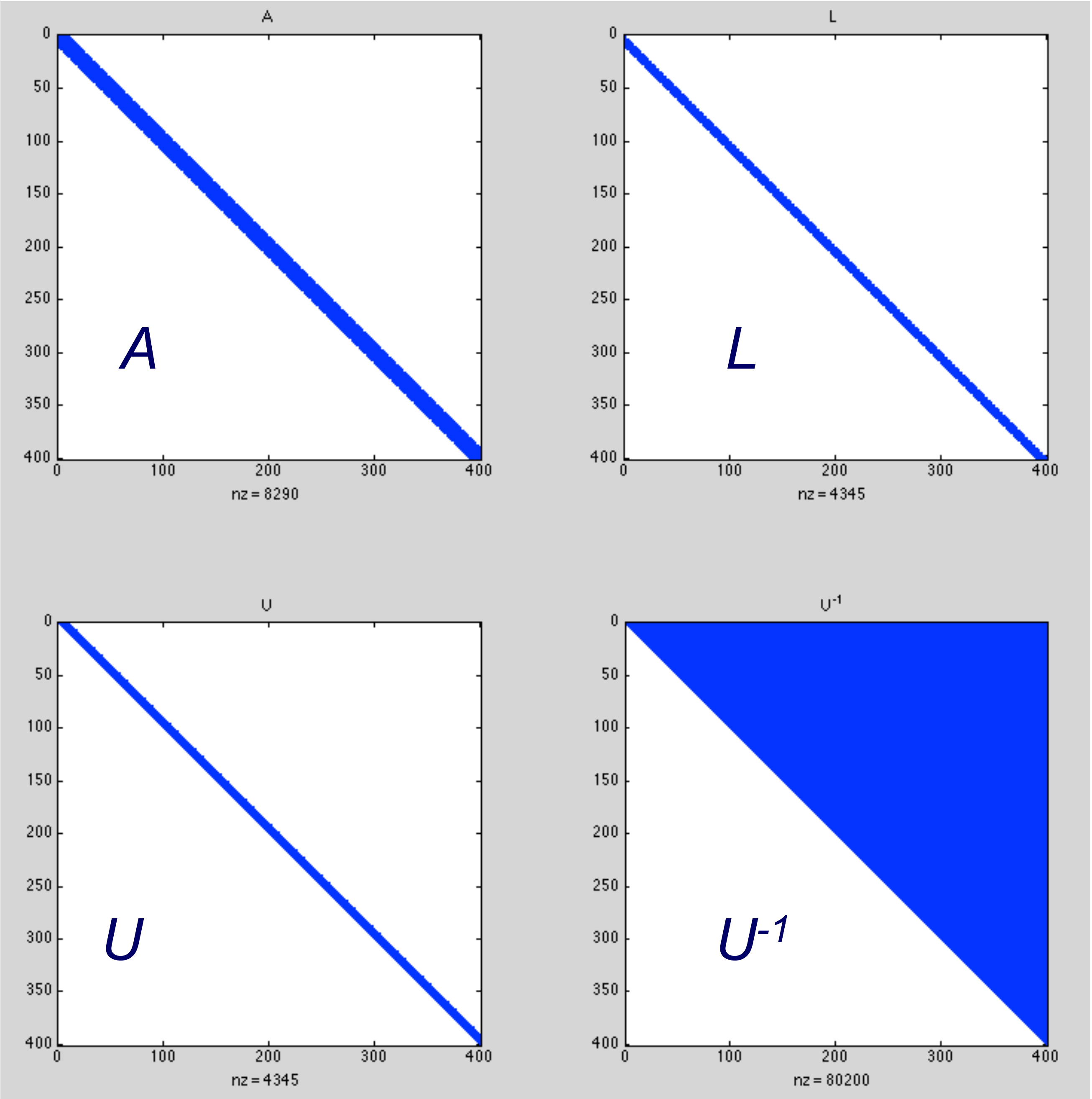
- The LU factors preserve the nonzero structure of A (unless there is pivoting, in which case, the bandwidth of L can grow by at most 2x).

- Storage / solve costs for LU is ~ 2nb

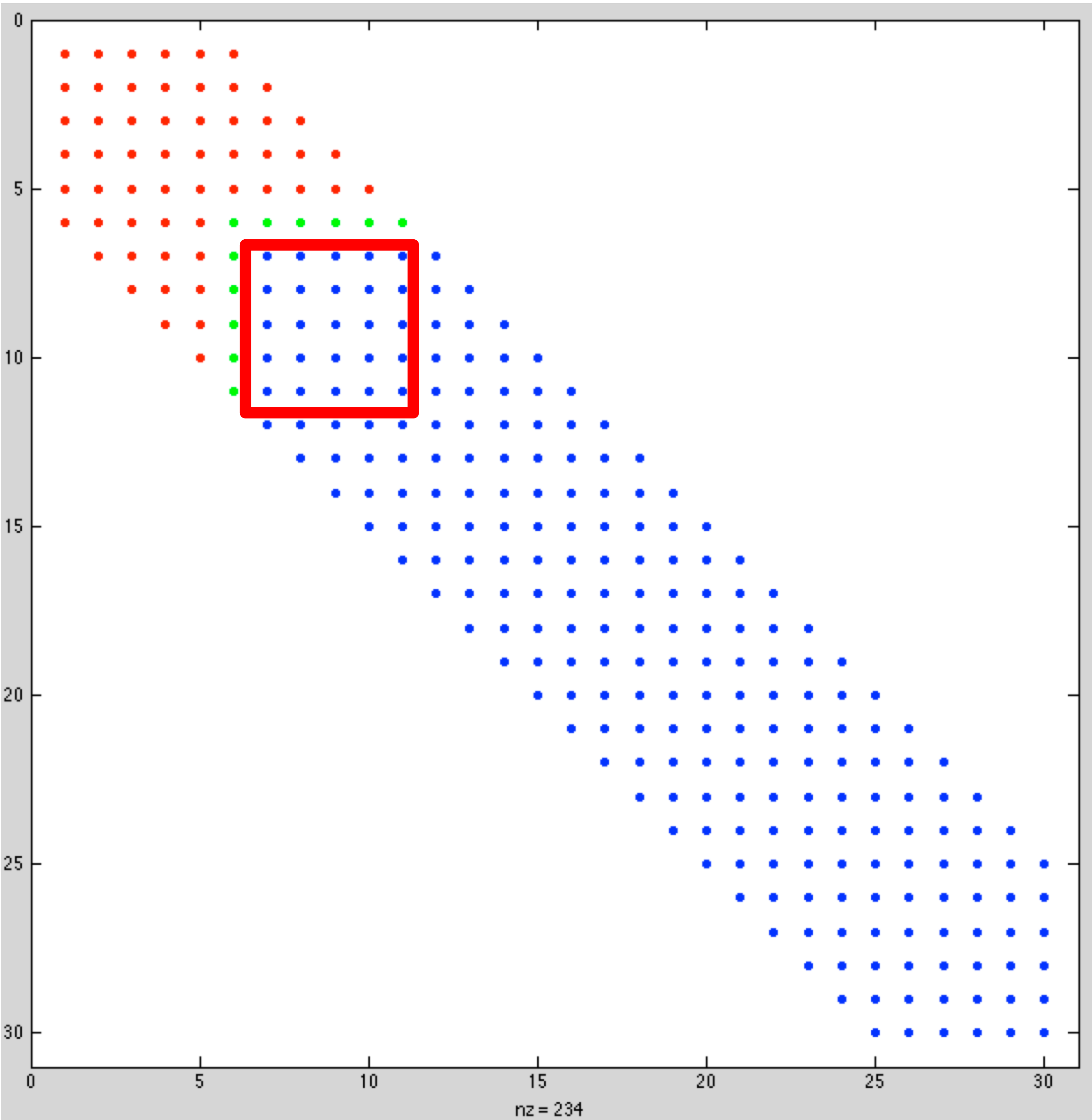
- Factor cost is ~ $n\,b^2 \ll n^3$

Band Matrices

Definitely do not invert **A** or **L** or **U** for banded systems!

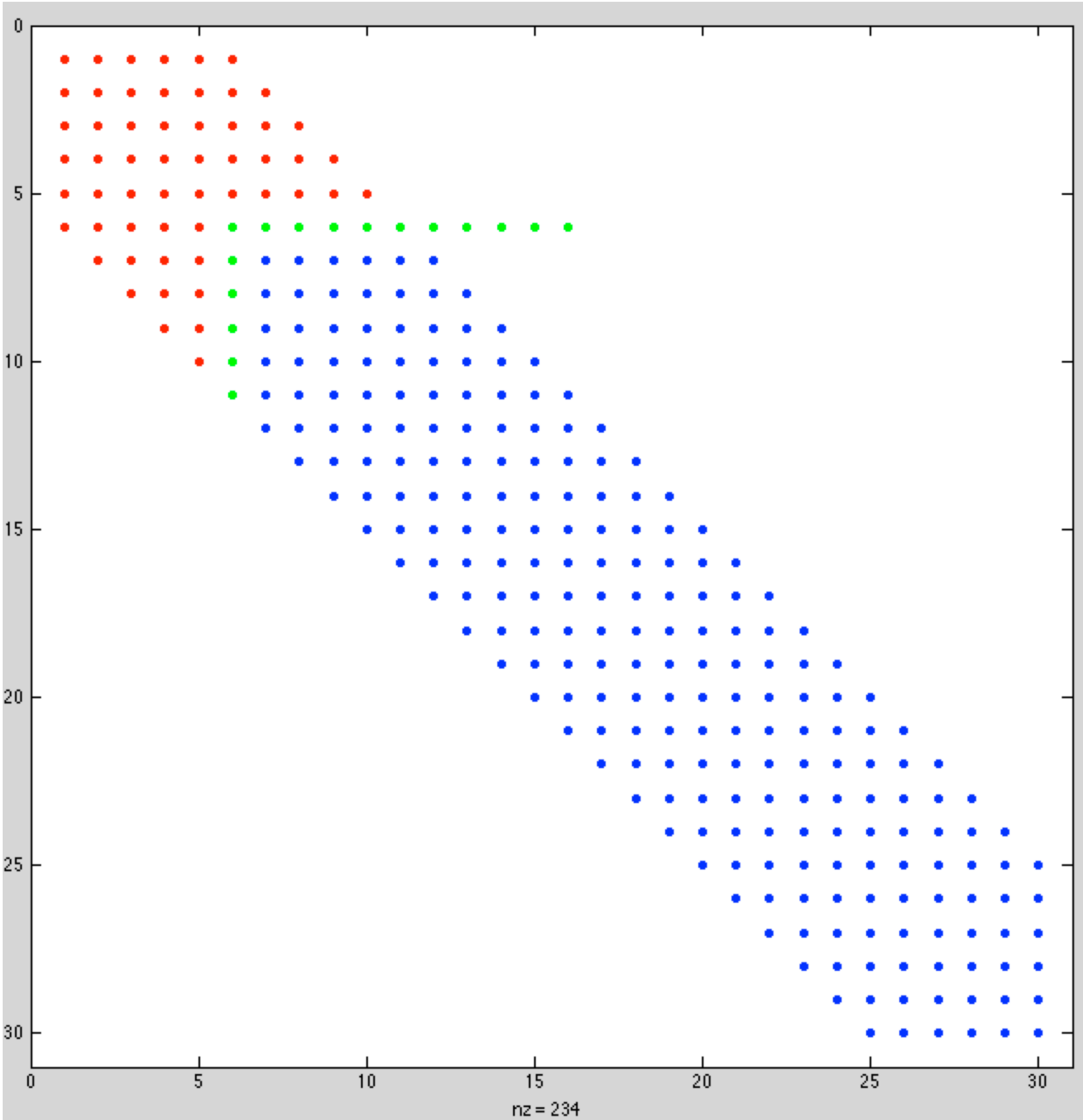


Cost of Banded Factorization



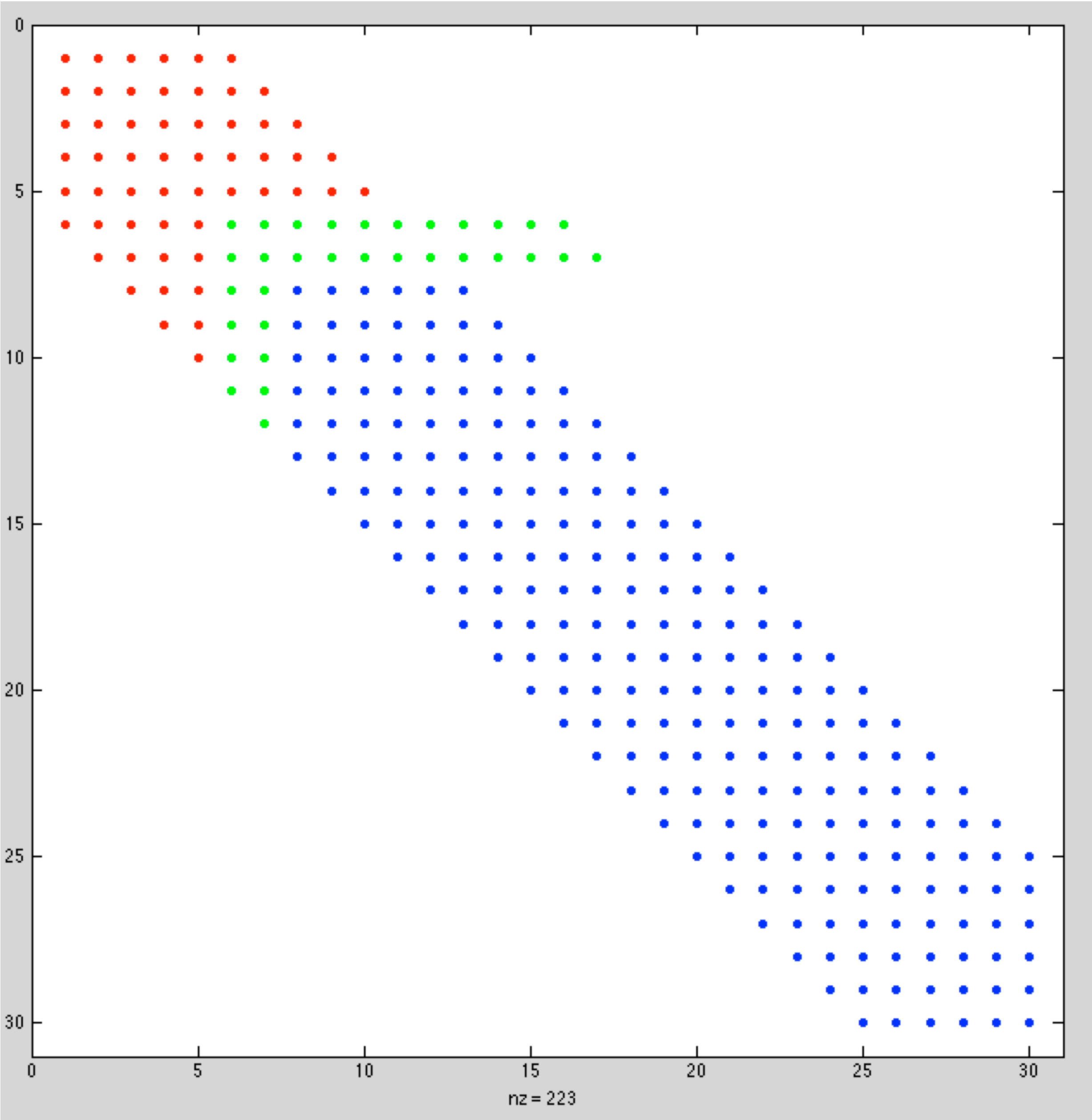
- Active submatrix for matrix with bandwidth b is (b x b).
- Work for outer product is $\mathbf{cr}^T$, which is outer product of two vectors of length b.
- So, total work is ~ n x (b^2) x 2 operations to convert A into LU.
- If we have pivoting, then bandwidth of U can grow by 2x.
- Note that if b=1, matrix is ***tridiagonal*** and factor cost is O(n) - ***optimal!***

Cost of Banded Factorization



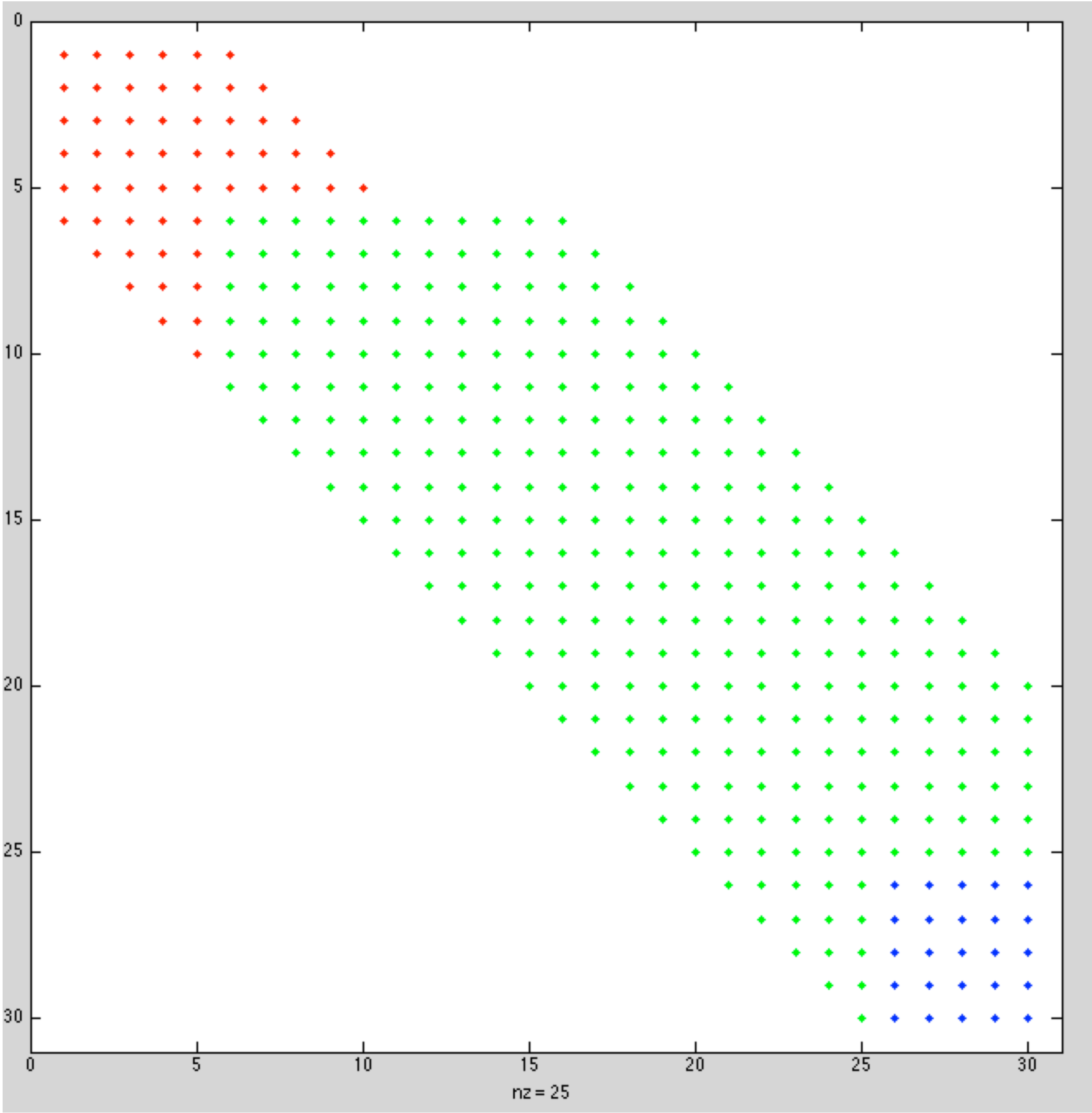
- Pivoting can pull a row that has 2b nonzeros to right of diagonal.
- U can end up with bandwidth 2b.

Cost of Banded Factorization



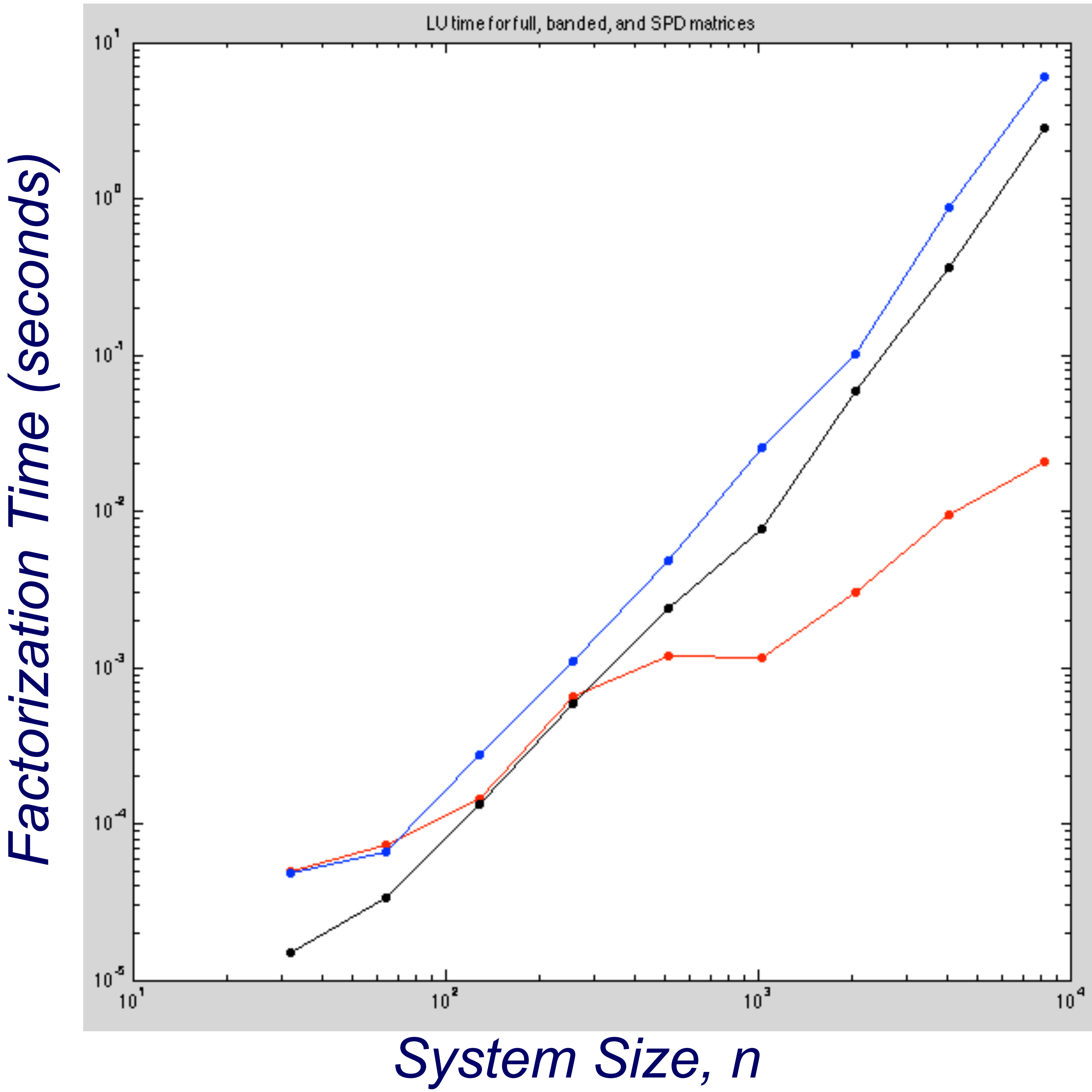
- Pivoting can pull a row that has 2b nonzeros to right of diagonal.
- U can end up with bandwidth 2b.

Cost of Banded Factorization



- Pivoting can pull a row that has 2b nonzeros to right of diagonal.
- U can end up with bandwidth 2b.

Solver Times, Banded, Cholesky (SPD), Full



Solver Times, Banded, Cholesky (SPD), Full

```
% Demo of banded-matrix costs

clear all;

for pass=1:2;
beta=10;

for k=4:13; n = 2^k;

   R=9*eye(n) + rand(n,n); S=R'*R; A=spalloc(n,n,1+2*beta);
   for i=1:n; j0=max(1,i-beta);j1=min(n,i+beta);
       A(i,j0:j1)=R(i,j0:j1);
   end;

   tstart=tic; [L,U]=lu(A); tsparse(k) = toc(tstart);
   tstart=tic; [L,U]=lu(R); tfull(k) = toc(tstart);
   tstart=tic; [C]=chol(S); tchol(k) = toc(tstart);
   nk(k)=n;
   sk(k)= (2*(n^3)/3)/(1.e9*tfull(k)); % GFLOPS
   ck(k)= (2*(n^3)/3)/(1.e9*tchol(k)); % GFLOPS

   [n tsparse(k) tfull(k) tchol(k)]

end;
loglog(nk,tsparse,'r.-',nk,tfull,'b.-',nk,tchol,'k.-')
axis square; title('LU time for full, banded, and SPD matrices')
```

Tridiagonal Matrices

- Consider tridiagonal matrix

$$\boldsymbol{A} = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{bmatrix}$$

- Gaussian elimination without pivoting reduces to

$d_1 = b_1$
for $i = 2$ **to** n
  $m_i = a_i / d_{i-1}$
  $d_i = b_i - m_i c_{i-1}$
end

Tridiagonal Matrices, continued

- LU factorization of $\mathbf{A}$ is then

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ m_2 & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & m_{n-1} & 1 & 0 \\ 0 & \cdots & 0 & m_n & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} d_1 & c_1 & 0 & \cdots & 0 \\ 0 & d_2 & c_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & d_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & d_n \end{bmatrix}$$

- Cost of solving $\mathbf{Ax} = \mathbf{b}$ without pivoting is $\sim 8n$ ops

Block Factorization

- Consider 2×2 block partition,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$$

- Perform block Gaussian elimination,

$$\mathbf{U} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{0} & \mathbf{S}_{22} \end{bmatrix}$$

- Here, $\mathbf{S}_{22} := \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$, is the *Schur complement*,
- Note that

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_{11} & \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I}_{22} \end{bmatrix},$$

as can be verified by showing that $\mathbf{LU} = \mathbf{A}$.

Block Factorization

- Block factorizations can be used in many ways.

- We've seen one already, in which we replace inefficient rank-1 updates with memory-efficienty rank-b updates, which lead to matrix-matrix products bearing the brunt of the computational effort

- The *Sherman-Morrison formula* is another instance of using block-factorization

Sherman Morrison

[1] Solve $A\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$:

$$A \longrightarrow LU \ \ (\ O(n^3) \text{ work }\ )$$

Solve $L\tilde{\mathbf{y}} = \tilde{\mathbf{b}}$,

Solve $U\tilde{\mathbf{x}} = \tilde{\mathbf{y}}$ ($O(n^2)$ work).

[2] New problem:

$$\left(A - \mathbf{u}\mathbf{v}^T\right)\mathbf{x} = \mathbf{b}. \quad \text{(different } \mathbf{x} \text{ and } \mathbf{b}\text{)}$$

Key Idea:

- $\left(A - \mathbf{u}\mathbf{v}^T\right)\mathbf{x}$ differs from $A\mathbf{x}$ by only a small amount of information.
- Rewrite as: $\quad A\mathbf{x} + \mathbf{u}\gamma = \mathbf{b}$

$$\gamma := -\mathbf{v}^T\mathbf{x} \quad \longleftrightarrow \quad \mathbf{v}^T\mathbf{x} + \gamma = 0$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} A & \mathbf{u} \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} A & \mathbf{u} \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

Eliminate for γ:

$$\begin{bmatrix} A & \mathbf{u} \\ 0 & 1 - \mathbf{v}^T A^{-1}\mathbf{u} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -\mathbf{v}^T A^{-1}\mathbf{b} \end{pmatrix}$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} A & \mathbf{u} \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

Eliminate for γ:

$$\begin{bmatrix} A & \mathbf{u} \\ 0 & 1 - \mathbf{v}^T A^{-1}\mathbf{u} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -\mathbf{v}^T A^{-1}\mathbf{b} \end{pmatrix}$$

$$\gamma = -\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}\mathbf{b}$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} A & \mathbf{u} \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

Eliminate for γ:

$$\begin{bmatrix} A & \mathbf{u} \\ 0 & 1 - \mathbf{v}^T A^{-1}\mathbf{u} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -\mathbf{v}^T A^{-1}\mathbf{b} \end{pmatrix}$$

$$\gamma = -\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}\mathbf{b}$$

$$\mathbf{x} = A^{-1}\left(\mathbf{b} - \mathbf{u}\gamma\right) = A^{-1}\left[\mathbf{b} + \mathbf{u}\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}\mathbf{b}\right]$$

Sherman Morrison

Extended system:

$$\begin{aligned} A\mathbf{x} + \gamma\mathbf{u} &= \mathbf{b} \\ \mathbf{v}^T\mathbf{x} + \gamma &= 0 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} A & \mathbf{u} \\ \mathbf{v}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ 0 \end{pmatrix}$$

Eliminate for γ:

$$\begin{bmatrix} A & \mathbf{u} \\ 0 & 1 - \mathbf{v}^T A^{-1}\mathbf{u} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ -\mathbf{v}^T A^{-1}\mathbf{b} \end{pmatrix}$$

$$\gamma = -\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}\mathbf{b}$$

$$\mathbf{x} = A^{-1}\left(\mathbf{b} - \mathbf{u}\gamma\right) = A^{-1}\left[\mathbf{b} + \mathbf{u}\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}\mathbf{b}\right]$$

$$\left(A - \mathbf{u}\mathbf{v}^T\right)^{-1} = A^{-1} \; + \; A^{-1}\mathbf{u}\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1} \mathbf{v}^T A^{-1}.$$

Sherman Morrison: Potential Singularity

- Consider the modified system: $\left(A - \mathbf{u}\mathbf{v}^T\right)\mathbf{x} \;=\; \mathbf{b}$.
- The solution is

$$\begin{aligned}\mathbf{x} \;&=\; \left(A - \mathbf{u}\mathbf{v}^T\right)^{-1}\mathbf{b} \\ &=\; \left[I + A^{-1}\mathbf{u}\left(1 - \mathbf{v}^T A^{-1}\mathbf{u}\right)^{-1}\mathbf{v}^T A^{-1}\right] A^{-1}\mathbf{b}.\end{aligned}$$

- If $1 \;-\; \mathbf{v}^T A^{-1}\mathbf{u} \;=\; 0$, failure.
- Why?

Sherman Morrison: Potential Singularity

- Let $\tilde{A} := \left(A - \mathbf{u}\mathbf{v}^T\right)$ and consider,

$$\begin{aligned} \tilde{A}\,A^{-1} &= \left(A - \mathbf{u}\mathbf{v}^T\right) A^{-1} \\ &= \left(I - \mathbf{u}\mathbf{v}^T A^{-1}\right). \end{aligned}$$

- Look at the product $\tilde{A}A^{-1}\mathbf{u}$,

$$\begin{aligned} \tilde{A}\,A^{-1}\mathbf{u} &= \left(I - \mathbf{u}\mathbf{v}^T A^{-1}\right) \mathbf{u} \\ &= \mathbf{u} - \mathbf{u}\mathbf{v}^T A^{-1}\mathbf{u}. \end{aligned}$$

- If $\mathbf{v}^T A^{-1}\mathbf{u} = 1$, then

$$\tilde{A}\,A^{-1}\mathbf{u} \;=\; \mathbf{u} - \mathbf{u} \;=\; 0,$$

  which means that $\tilde{A}$ is singular since we assume that A^{-1} exists.

- Thus, an unfortunate choice of $\mathbf{u}$ and $\mathbf{v}$ can lead to a singular modified matrix and this singularity is indicated by $\mathbf{v}^T A^{-1}\mathbf{u} = 1$.

Sherman-Morrison Example

- **Q:** What is the cost of solving $\mathbf{Ax} = \mathbf{b}$ if $\mathbf{A}$ is $n \times n$ and of the form below?

$$A = \begin{bmatrix} 1.0 & -.1 & -.1 & -.1 & -.1 & -.1 & -.1 & -.1 \\ -.1 & 1.0 & -.1 & -.1 & -.1 & -.1 & -.1 & -.1 \\ -.1 & -.1 & 1.0 & -.1 & -.1 & -.1 & -.1 & -.1 \\ -.1 & -.1 & -.1 & 1.0 & -.1 & -.1 & -.1 & -.1 \\ -.1 & -.1 & -.1 & -.1 & 1.0 & -.1 & -.1 & -.1 \\ -.1 & -.1 & -.1 & -.1 & -.1 & 1.0 & -.1 & -.1 \\ -.1 & -.1 & -.1 & -.1 & -.1 & -.1 & 1.0 & -.1 \\ -.1 & -.1 & -.1 & -.1 & -.1 & -.1 & -.1 & 1.0 \end{bmatrix}$$

- **A:** $O(n)$!