- Sherman Morrison Formula

Solving Modified Problems

- If right-hand side of linear system changes but matrix does not, then LU factorization need not be repeated to solve new system
- Only forward- and back-substitution need be repeated for new right-hand side
- This is substantial savings in work, since additional triangular solutions cost only $\mathcal{O}(n^2)$ work, in contrast to $\mathcal{O}(n^3)$ cost of factorization

Sherman-Morrison Formula

- Sometimes refactorization can be avoided even when matrix *does* change
- *Sherman-Morrison formula* gives inverse of matrix resulting from rank-one change to matrix whose inverse is already known

$$(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1}$$

  where $\boldsymbol{u}$ and $\boldsymbol{v}$ are n-vectors
- Evaluation of formula requires $\mathcal{O}(n^2)$ work (for matrix-vector multiplications) rather than $\mathcal{O}(n^3)$ work required for inversion

Rank-One Updating of Solution

- To solve linear system $(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)\boldsymbol{x} = \boldsymbol{b}$ with new matrix, use Sherman-Morrison formula to obtain

$$\begin{aligned}\boldsymbol{x} &= (\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1}\boldsymbol{b} \\ &= \boldsymbol{A}^{-1}\boldsymbol{b} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{b}\end{aligned}$$

  which can be implemented by following steps

  - Solve $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u}$ for $\boldsymbol{z}$, so $\boldsymbol{z} = \boldsymbol{A}^{-1}\boldsymbol{u}$
  - Solve $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$, so $\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{b}$
  - Compute $\boldsymbol{x} = \boldsymbol{y} + ((\boldsymbol{v}^T\boldsymbol{y})/(1 - \boldsymbol{v}^T\boldsymbol{z}))\boldsymbol{z}$

- If $\boldsymbol{A}$ is already factored, procedure requires only triangular solutions and inner products, so only $\mathcal{O}(n^2)$ work and no explicit inverses

Example: Rank-One Updating of Solution

- Consider rank-one modification

$$\begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -1 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix}$$

(with $3, 2$ entry changed) of system whose LU factorization was computed in earlier example

- One way to choose update vectors is

Original Matrix

$$\boxed{\begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix}}$$

$$\boldsymbol{u} = \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

so matrix of modified system is $\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T$

Example, continued

- Using LU factorization of $\boldsymbol{A}$ to solve $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u}$ and $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$,

$$\boldsymbol{z} = \begin{bmatrix} -3/2 \\ 1/2 \\ -1/2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

- Final step computes updated solution

Q: Under what circumstances could the denominator be zero ?

$$\boldsymbol{x} = \boldsymbol{y} + \frac{\boldsymbol{v}^T\boldsymbol{y}}{1 - \boldsymbol{v}^T\boldsymbol{z}}\boldsymbol{z} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix} + \frac{2}{1 - 1/2}\begin{bmatrix} -3/2 \\ 1/2 \\ -1/2 \end{bmatrix} = \begin{bmatrix} -7 \\ 4 \\ 0 \end{bmatrix}$$

- We have thus computed solution to modified system without factoring modified matrix

Sherman Morrison

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

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

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

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

[2] New problem:

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

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: $A\mathbf{x} + \mathbf{u}\gamma = \mathbf{b}$

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

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 $\mathbf{v}^T A^{-1}\mathbf{u} = 1$.

Tensor Product Matrics

The tensor- (or Kronecker-) product of matrices A and B is denoted as

$$C \; = \; A \otimes B$$

and is defined as the block matrix having entries

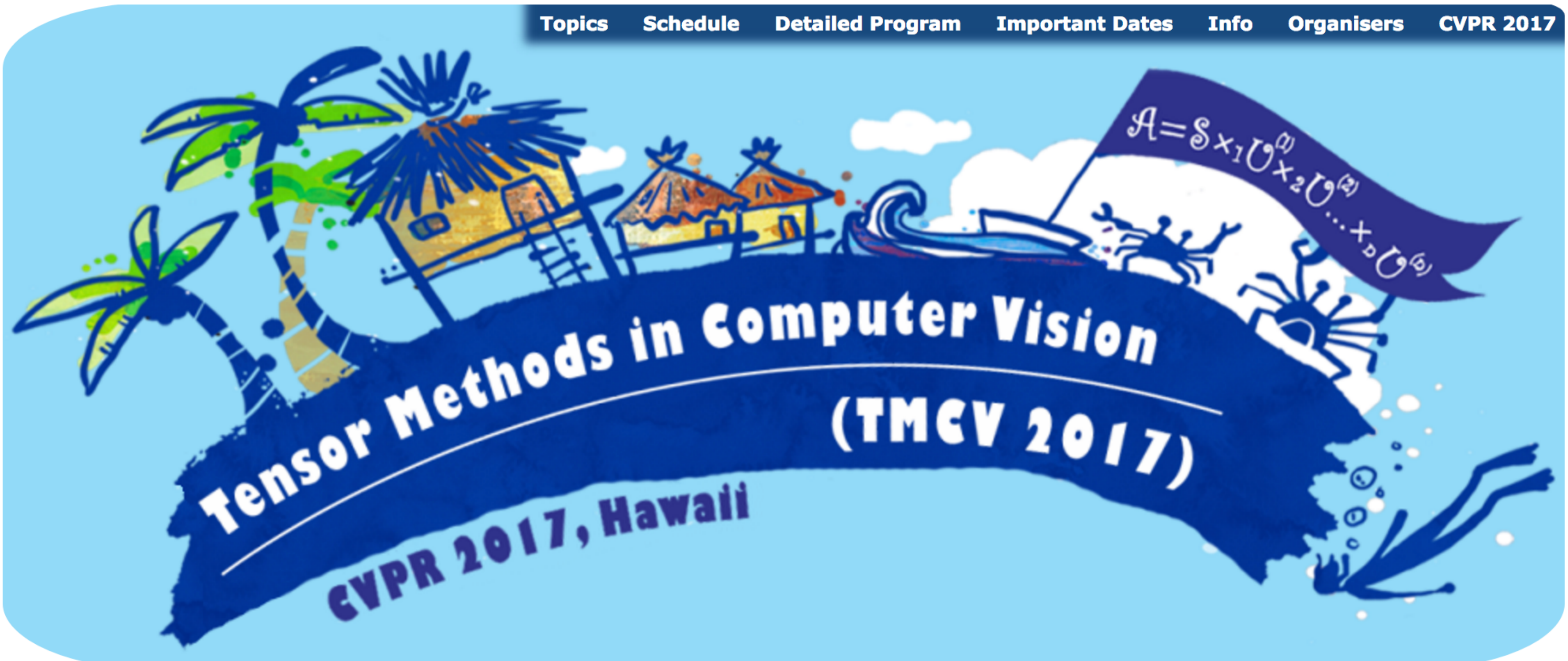
$$C \quad := \quad \begin{pmatrix} a_{11}B & a_{12}B & \cdots & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & \cdots & a_{2n}B \\ \vdots & \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & \cdots & a_{mn}B \end{pmatrix}.$$

Tensor-Product Matrices

- Tensor-product forms arise in many applications, including
  - Density Functional Theory (DFT) in computational chemistry (e.g., 7-dimensional tensors)
  - Partial differential equations
  - Image processing (e.g., multidimensional FFTs)
  - Machine learning (ML)
- Their importance in ML/AI applications is such that software developers and computer architects are now designing fast tensor-contraction engines to further accelerate tensor-product manipulations.

Tensor-Product Matrices

- In Computer Vision, there is even a conference series on this topic.



- Our interest here is to understand how tensor-product forms can yield very rapid direct solvers for systems of the form $A\mathbf{x} = \mathbf{b}$.

- There are two ways in which tensor-product-based matrices for the form $C = A \otimes B$ accelerate computation:

  1. They can be used to effect very fast matrix-vector products.
  2. They can be used to effect very fast matrix-matrix products.

- To begin, we focus on the matrix-matrix products, which is a bit easier to understand.

Product Rule for Tensor-Product Matrices

- Assume that the matrix pairs (D, A) and (E, B) are dimensioned such that the products DA and EB are well-defined.

- If

$$C \;:=\; A \otimes B \quad \text{and} \quad F \;:=\; D \otimes E,$$

  then, the matrix product FC is given by

$$\begin{aligned} FC &= (D \otimes E)\,(A \otimes B) \\ &= DA \otimes EB. \end{aligned}$$

- This result follows from the definition of the Kronecker product, $\otimes$, and has many important consequences.

Uses of the Product Rule: Inverses

$$(D \otimes E)\,(A \otimes B) \;=\; DA \otimes EB.$$

- If $C \;:=\; A \otimes B$, then

$$C^{-1} \;:=\; A^{-1} \otimes B^{-1}.$$

- Specifically,

$$\begin{aligned} C^{-1}C \;&=\; \left(A^{-1} \otimes B^{-1}\right)(A \otimes B) \;=\; A^{-1}A \otimes B^{-1}B \\ &=\; I_A \otimes I_B \;=\; I, \end{aligned}$$

  where I_A and I_B are identity matrices equal in size to A and B, repsectively.

- Thus, the inverse of C is the tensor-product of two much smaller matrices, A and B.

Uses of the Product Rule: Inverses

- Example:
  - Suppose A and B are full $N \times N$ matrices and $C = A \otimes B$ is $n \times n$ with $n = N^2$.
  - The LU factorization of C is

$$LU = (L_A \otimes L_B)(U_A \otimes U_B).$$

  - What is the cost of computing the tensor product form of LU, rather than LU directly as a function of N?
  - What is the ratio (full time over tensor-product time) when $N = 100$?

The Curse of Dimensionality

- The advantage of the tensor-product representation increases with higher dimensions.
- Suppose A_j is $N \times N$, for $j = 1, \dots, d$, and

$$C \;=\; A_d \otimes A_{d-1} \otimes \cdots \otimes A_1,$$

  with inverse

$$C^{-1} \;=\; A_d^{-1} \otimes A_{d-1}^{-1} \otimes \cdots \otimes A_1^{-1}.$$

- Tensor-product forms are *critical* for efficient computation in many large-dimensional scientific problems.
- *Application* of the tensor operator, however, will take more work, since we obviously have to touch $n = 10^7$ entries. We'll see in a moment that the cost of application is $\approx\, 2d \cdot n \cdot n^{\frac{1}{d}} \;\ll\; O(n^3)$.

- Consider $d = 7$ and $N = 10$.
  - The number of nonzeros in C (*if formed*) is N^{14}, which is 800 TB and would cost you about \$10,000 in disk drives.
  - Factorization of the full form will take about 10 minutes on the world's fastest computer in 2021, or about 600 years on my mac.
  - The factorization cost for the tensor product form is $\approx$ 5000 operations. A blink of the eye on your laptop.
  - Application of C^{-1} in tensor form will require about $2 \cdot 7 \cdot 10^8 \approx 1.4 \times 10^9$ operations, which is less than a second if you sustain $>$ 1 GFLOPS on your computer.

- With the significant reduction of memory references and operations, the cost of application of C^{-1} in the high-rank tensor case is typically dominated by the cost of transfering the right-hand side and solution vectors from and to main memory. That is, the cost scales like $cn = cN^d$, where c is some measure of the inverse memory bandwidth.

  Thus, high-rank tensors transform a compute-bound problem to a memory-bound one.

Uses of the Product Rule: Eigenvalues

- Suppose that A is an $N \times N$ matrix with the *similarity transformation* (Chapter 4),

$$A \;=\; S \Lambda S^{-1},$$

  where $S = [\mathbf{s}_1 \, \mathbf{s}_2 \cdots \mathbf{s}_N]$ is the (full) matrix of eigenvectors of A and Λ=diag(λ_i) is the diagonal matrix of corresponding eigenvalues.

  That is, $A\mathbf{s}_i \;=\; \mathbf{s}_i \, \lambda_i$.

- Let $T \mathcal{M} T^{-1}$ denote the similarity transformation for B, with eigenvector matrix T and eigenvalue matrix $\mathcal{M}$.

- Then the similarity transformation for $C = A \otimes B$ is

$$\begin{aligned} A \otimes B \;&=\; \left(S\Lambda S^{-1}\right) \otimes \left(T\mathcal{M}T^{-1}\right) \\ &=\; (S \otimes T)\,(\Lambda \otimes \mathcal{M})\,\left(S^{-1} \otimes T^{-1}\right) \\ &=\; U \mathcal{N} U^{-1}. \end{aligned}$$

- Thus, we have diagonalized C by diagonalizing two smaller systems A and B.

Fast Matrix-Vector Products

- Q: What is the cost of $C\mathbf{u}$, vs. the fast form for $(A \otimes B)\mathbf{u}$?

Fast Matrix-Vector Products via Tensor Contraction

- Consider evaluation of $\mathbf{w} = C\mathbf{v} := (A \otimes B)\mathbf{u}$.
- To avoid extra work and storage, we evaluate the product as

$$\mathbf{w} = (A \otimes I)(I \otimes B)\mathbf{u},$$

or

$$\begin{aligned} \mathbf{v} &= (I \otimes B)\mathbf{u}, \\ \mathbf{w} &= (A \otimes I)\mathbf{u}. \end{aligned}$$

- Start with $\mathbf{v} = (I \otimes B)\mathbf{u}$.

$$\underbrace{\left(\begin{array}{c} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_N \\ \hline v_{1+N} \\ v_{2+N} \\ \vdots \\ \vdots \\ v_{2N} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ v_n \end{array}\right)}_{\mathbf{v}} = \underbrace{\left(\begin{array}{c|c|c|c} B & & & \\ \hline & B & & \\ \hline & & B & \\ \hline & & & B \end{array}\right)}_{I \otimes B} \underbrace{\left(\begin{array}{c} u_1 \\ v_2 \\ \vdots \\ \vdots \\ u_N \\ \hline u_{1+N} \\ u_{2+N} \\ \vdots \\ \vdots \\ u_{2N} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ u_n \end{array}\right)}_{\mathbf{u}}$$

- In $(I \otimes B)\mathbf{u}$, B is applied M times to vectors of length M.

- We can reshape the vector $\mathbf{u}$ and output vector $\mathbf{v}$ to be $M \times N$ *matrices*, such that $\mathbf{v} = (I \otimes B)\mathbf{u}$ is computed as a *matrix-matrix* product:

$$\begin{pmatrix} v_1 & v_{1+M} & v_{\dots} & v_{\dots} & v_{\dots} \\ v_2 & v_{2+M} & v_{\dots} & v_{\dots} & v_{\dots} \\ v_{\dots} & v_{\dots} & v_{\dots} & v_{\dots} & v_{\dots} \\ v_{\dots} & v_{\dots} & v_{\dots} & v_{\dots} & v_{\dots} \\ v_M & v_{2M} & v_{\dots} & v_{\dots} & v_n \end{pmatrix} = \begin{pmatrix} b_{11} & b_{12} & b_{\dots} & b_{\dots} & b_{1M} \\ b_{21} & b_{22} & b_{\dots} & b_{\dots} & b_{2M} \\ b_{\dots} & b_{\dots} & b_{\dots} & b_{\dots} & b_{\dots} \\ b_{\dots} & b_{\dots} & b_{\dots} & b_{\dots} & b_{\dots} \\ b_{M1} & b_{M2} & b_{\dots} & b_{\dots} & b_{MM} \end{pmatrix} \begin{pmatrix} u_1 & u_{1+M} & u_{\dots} & u_{\dots} & u_{\dots} \\ u_2 & u_{2+M} & u_{\dots} & u_{\dots} & u_{\dots} \\ u_{\dots} & u_{\dots} & u_{\dots} & u_{\dots} & u_{\dots} \\ u_{\dots} & u_{\dots} & u_{\dots} & u_{\dots} & u_{\dots} \\ u_M & u_{2M} & u_{\dots} & u_{\dots} & u_n \end{pmatrix}$$

- It is convenient to relabel the indices on $\mathbf{u}$ and $\mathbf{v}$ to match the contraction indices of the tensor operator.

- Specifically, let $\mathbf{u} = (u_1\, u_2\, \ldots u_n)^T$ and U be the matrix form with entries

$$U_{ij} \;=\; u_{\hat{\imath}}, \quad \text{for } \hat{\imath} \;:=\; i + M(j-1).$$

- Then, with the same mapping for $\mathbf{b} \longrightarrow V$, we can write

$$V \;=\; BU.$$

- In index form (convenient for later...)

$$V_{ij} \;=\; \sum_{p=1}^{M} B_{ip} U_{pj}.$$

- The next step is to compute $\mathbf{w} = (A \otimes I)\mathbf{v}$:

$$
\underbrace{\left(\begin{array}{c} w_{11} \\ w_{21} \\ \vdots \\ \vdots \\ w_{M1} \\ \hline w_{12} \\ w_{22} \\ \vdots \\ \vdots \\ w_{M2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline w_{1N} \\ w_{2N} \\ \vdots \\ \vdots \\ w_{MN} \end{array}\right)}_{\mathbf{w}}
=
\underbrace{\left(\begin{array}{c|c|c|c}
\begin{matrix} a_{11} & & & & \\ & a_{11} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{11} \end{matrix} &
\begin{matrix} a_{12} & & & & \\ & a_{12} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{12} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{1N} & & & & \\ & a_{1N} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{1N} \end{matrix} \\ \hline
\begin{matrix} a_{21} & & & & \\ & a_{21} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{21} \end{matrix} &
\begin{matrix} a_{22} & & & & \\ & a_{22} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{22} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{2N} & & & & \\ & a_{2N} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{2N} \end{matrix} \\ \hline
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} \\ \hline
\begin{matrix} a_{N1} & & & & \\ & a_{N1} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{N1} \end{matrix} &
\begin{matrix} a_{N2} & & & & \\ & a_{N2} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{N2} \end{matrix} &
\begin{matrix} a_{..} & & & & \\ & a_{..} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{..} \end{matrix} &
\begin{matrix} a_{NN} & & & & \\ & a_{NN} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & a_{NN} \end{matrix}
\end{array}\right)}_{A \otimes I}
\underbrace{\left(\begin{array}{c} v_{11} \\ v_{21} \\ \vdots \\ \vdots \\ v_{M1} \\ \hline v_{12} \\ v_{22} \\ \vdots \\ \vdots \\ v_{M2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline v_{1N} \\ v_{2N} \\ \vdots \\ \vdots \\ v_{MN} \end{array}\right)}_{\mathbf{v}}
$$

- Here, the picture is less obvious than for the block-diagonal $(I \otimes B)$ case.

- To make things simpler, we've enumerated $\mathbf{v}$ and $\mathbf{w}$ with the two-index subscript in the preceding slide such that they are already in tensor form.

- With a bit of inspection, it becomes clear that $\mathbf{w} = (A \otimes I)\mathbf{v}$ is given by a contraction that is similar to the preceding one. Namely,

$$W_{ij} = \sum_{q=1}^{M} A_{jq} V_{iq} = \sum_{q=1}^{M} A^T_{qj} V_{iq} = \sum_{q=1}^{M} V_{iq} A^T_{qj}.$$

- The last form is a proper matrix-matrix product of the form $W = V A^T$.

- The complete contraction evaluation, $\mathbf{w} = (A \otimes B)\mathbf{u}$, for 2D (i.e., rank-2) tensors is thus simply,

$$W = B U A^T.$$

- Contractions for higher-rank tensors take on a similar form.

- For example, a rank-3 contraction $\mathbf{w} = (A \otimes B \otimes C)\mathbf{u}$ is evaluated as

$$w_{ijk} = \sum_{r=1}^{N_A} \sum_{q=1}^{N_B} \sum_{p=1}^{N_C} A_{kr} B_{jq} C_{ip}\, u_{pqr} = \sum_{r=1}^{N_A} A_{kr} \left[\sum_{q=1}^{N_B} B_{jq} \left(\sum_{p=1}^{N_C} C_{ip}\, u_{pqr} \right) \right].$$

- The second form on the right implements the fast evaluation,

$$(A \otimes I \otimes I)(I \otimes B \otimes I)(I \otimes I \otimes C).$$

[See Deville, F. , Mund, 2002]

- More generally, for $\mathbf{w} = (A^d \otimes A^{d-1} \otimes \cdots \otimes A^1)\mathbf{u}$, one has

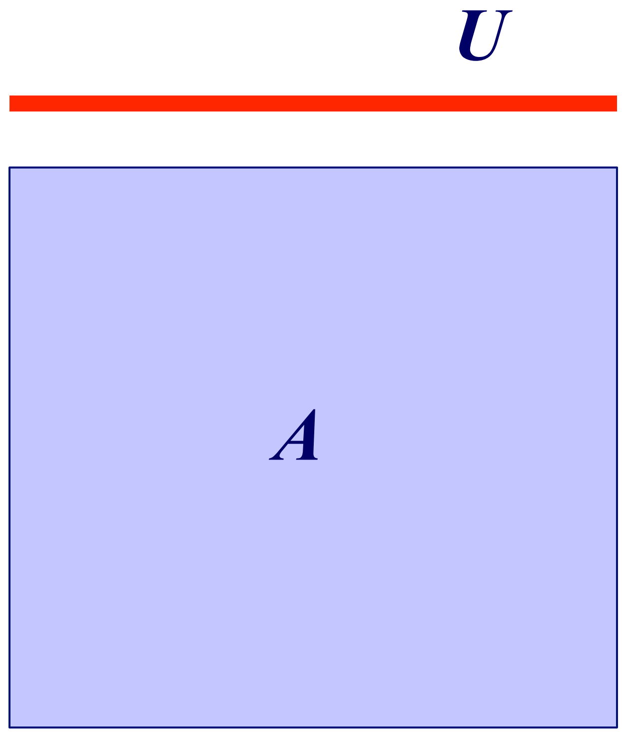
$$w_{i_1 i_2 \cdots i_d} = \sum_{j_d=1}^{N_d} A^d_{i_d j_d} \left[\sum_{j_{d-1}=1}^{N_{d-1}} A^{d-1}_{i_{d-1} j_{d-1}} \left(\cdots \sum_{j_1=1}^{N_1} A^1_{i_1 j_1} u_{j_1 j_2 \cdots j_d} \right) \right].$$

- If $N_1 = N_2 = \cdots = N_d = N$, then the amount of data movement is $N^d + dN^2$ loads for $\mathbf{u}$ and A^k and N^d stores ($N^d = n$).

- The number of operations is $2dN^d \cdot N = 2dnN = 2dn^{1+\frac{1}{d}}$, so we see that these schemes are nearly linear in n for large values of d.

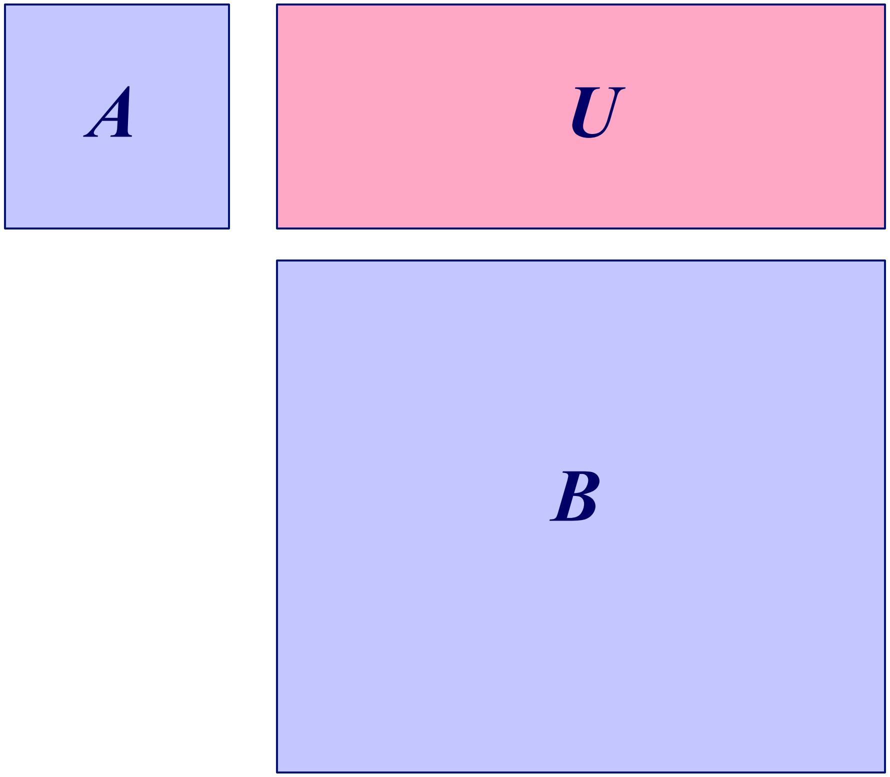
Contractions Pictorially

- 1D:



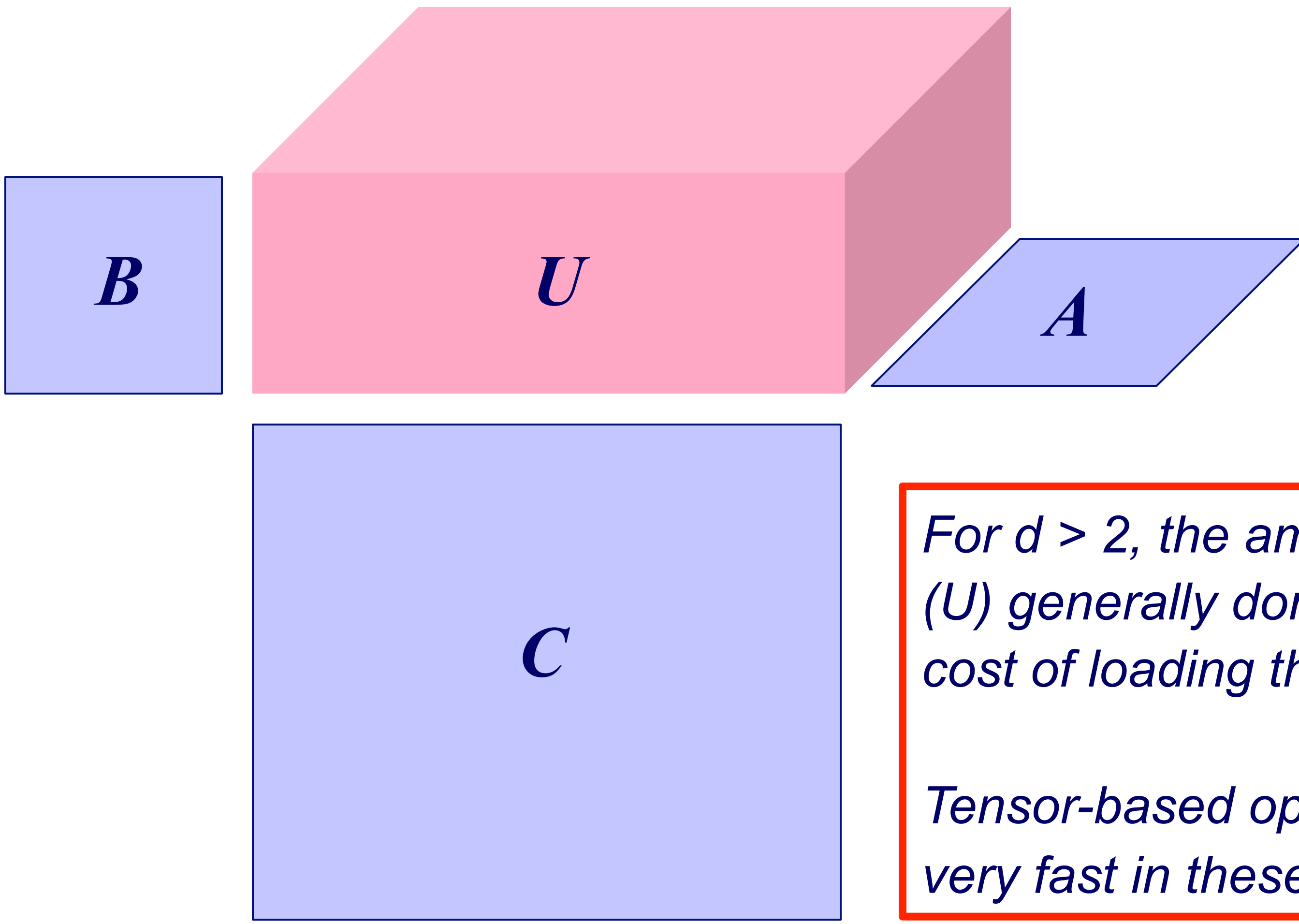
Contractions Pictorially

- 2D: $(A \otimes B)\, U$



Contractions Pictorially

- 3D: $(A \otimes B \otimes C)\, U$



For d > 2, the amount of data (U) generally dominates the cost of loading the operators.

Tensor-based operators are very fast in these cases.

Fast Solvers: Other Systems

Fast Solver Example

- Consider the system $A_{2D}\,\mathbf{u} \;=\; \mathbf{f}$:

$$
\frac{1}{h^2}
\underbrace{\left(\begin{array}{c|c|c|c}
\begin{array}{ccccc} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{array} &
\begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} & & \\ \hline
\begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} &
\begin{array}{ccccc} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{array} &
\begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} & \\ \hline
 & \begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} &
\begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} &
\begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} \\ \hline
 & & \begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} &
\begin{array}{ccccc} 4 & -1 & & & \\ -1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{array}
\end{array}\right)}_{A_{2D}}
\underbrace{\left(\begin{array}{c}
u_{11} \\ u_{21} \\ \vdots \\ \vdots \\ u_{M1} \\ \hline u_{12} \\ u_{22} \\ \vdots \\ \vdots \\ u_{M2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline u_{1N} \\ u_{2N} \\ \vdots \\ \vdots \\ u_{MN}
\end{array}\right)}_{\mathbf{u}}
=
\underbrace{\left(\begin{array}{c}
f_{11} \\ f_{21} \\ \vdots \\ \vdots \\ f_{M1} \\ \hline f_{12} \\ f_{22} \\ \vdots \\ \vdots \\ f_{M2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline f_{1N} \\ f_{2N} \\ \vdots \\ \vdots \\ f_{MN}
\end{array}\right)}_{\mathbf{f}}
$$

- This system is the 2D analog of the 1D finite-difference approximation to the heat equation.

- That is,

$$-\left[\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \;+\; \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right] \;=\; f_{ij},$$

  approximates the Poisson equation

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \;=\; f(x, y),$$

  with $u = 0$ on the boundary of the domain $\Omega = [0, M\Delta x] \times [0, N\Delta y]$.

- The details of the discretization are not our principal focus at this point.

- Here, we explore fast direct (noniterative) solution methods.

1D Poisson System

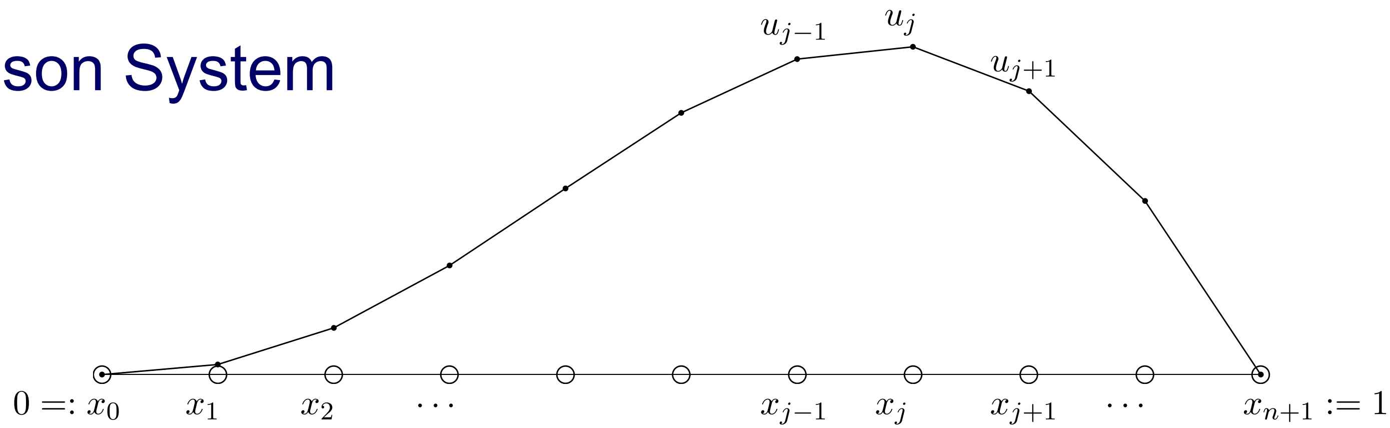


Figure 1: Finite difference grid on $\Omega := [0,1]$.

$$-\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} \quad = \quad f_j, \qquad j = 1, \ldots, n.$$

- This expression approximates the 1D differential equation $-\dfrac{d^2u}{dx^2} = f(x)$, $u(0) = u(L) = 0$.
- Each equation j relates u_{j-1}, u_j, and u_{j+1} to f_j.
- For this reason, the resulting matrix system is *tridiagonal*,

$$\underbrace{\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}}_{A_x} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{pmatrix}}_{\mathbf{u}} = \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{pmatrix}}_{\mathbf{f}}.$$

Properties of A_x

- A_x is *symmetric*, which implies it has real eigenvalues and an orthonormal set of eigenvectors satisfying $A_x \mathbf{s}_j = \lambda_j \mathbf{s}_j$, $\mathbf{s}_j^T \mathbf{s}_i = \delta_{ij}$, where the Kronecker δ_{ij} equals 1 when $i = j$ and 0 when $i \neq j$.

- A_x is also *positive definite*, which means that $\mathbf{x}^T A_x \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$. It also implies $\lambda_j > 0$. Symmetric positive definite (SPD) systems are particularly attractive because they can be solved without pivoting using Cholesky factorization, $A_x = LL^T$, or iteratively using preconditioned conjugate gradient (PCG) iteration. (For large sparse systems, PCG is typically the best option.)

- A_x is *sparse*. It has a fixed maximal number of nonzeros per row, which implies that the total number of nonzeros in A_x is linear in the problem size, n. We say that the storage cost for A_x is $O(n)$, meaning that there exists a constant C independent of n such that the total number of words to be stored is $< Cn$.

- A_x is *banded* with bandwidth $w = 1$, which implies that $k_{ij} = 0$ for all $|i - j| > w$. A consequence is that the storage bound for the Cholesky factor L is $< (w+1)n$. For the 1D case with w=1, the storage for L is thus $O(n)$. As we shall see, the work to compute the factors is $O(w^2 n)$.

- Returning to the 2D case, we see that we can express A_{2D} as $(I_y \otimes A_x) + (A_y \otimes I_x)$.

- The first term is nothing other than $\frac{\delta^2}{\delta x^2}$ being applied to each row (j) of u_{ij} and the second term amounts to applying $\frac{\delta^2}{\delta y^2}$ to each column (i) on the grid.

- For $h := \Delta x = \Delta y$, the left and right terms take on forms that we've already seen.

$$
\begin{aligned}
A_{2D} &= \begin{pmatrix} A_x & & & \\ & A_x & & \\ & & \ddots & \\ & & & A_x \end{pmatrix} + \frac{1}{h^2} \begin{pmatrix} 2I_x & -I_x & & \\ -I_x & 2I_x & \ddots & \\ & \ddots & \ddots & -I_x \\ & & -I_x & 2I_x \end{pmatrix} \\
&= (I_y \otimes A_x) \; + \; (A_y \otimes I_x)
\end{aligned}
$$

$$\frac{\partial^2 u}{\partial x^2} \;\; term$$

$$\frac{1}{h^2}\left(\begin{array}{c|c|c|c}
\begin{matrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{matrix} & & & \\ \hline
& \begin{matrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{matrix} & & \\ \hline
& & \begin{matrix} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{matrix} & \\ \hline
& & & \begin{matrix} 2 & -1 & & & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{matrix}
\end{array}\right)$$

$$\frac{\partial^2 u}{\partial y^2} \;\; term$$

$$\frac{1}{h^2}\left(\begin{array}{c|c|c|c}
\begin{matrix} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{matrix} &
\begin{matrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{matrix} & & \\
\hline
\begin{matrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{matrix} &
\begin{matrix} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{matrix} &
\begin{matrix} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{matrix} & \\
\hline
 &
\begin{matrix} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{matrix} &
\begin{matrix} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{matrix} &
\begin{matrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{matrix} \\
\hline
 & &
\begin{matrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{matrix} &
\begin{matrix} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{matrix}
\end{array}\right)$$

$$A_{2D} \;=\; (I_y \otimes A_x) \;+\; (A_y \otimes I_x),$$

- Because the A_{2D} is the sum of two systems, we can't use the tensor-product inverse directly.

- We instead use the similarity transformation introduced earlier. Specifically, compute the (small) similarity transformations

$$A_x \;=\; S_x \Lambda_x S_x^{-1}, \qquad\qquad A_y \;=; S_y \Lambda_y S_y^{-1},$$

- Noting that $I_x = S_x I_x S_x^{-1}$ and $I_y = S_y I_y S_y^{-1}$, we have

$$\begin{aligned}
A_{2D} &= (S_y I_y S_y^{-1} \otimes S_x \Lambda_x S_x^{-1}) \;+\; (S_y \Lambda_y S_y^{-1} \otimes S_x I_x S_x^{-1}) \\
&= (S_y \otimes S_x)(I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)(S_y^{-1} \otimes S_x^{-1}) \\
&= S \Lambda S^{-1}.
\end{aligned}$$

- The inverse is then $A_{2D}^{-1} = S \Lambda^{-1} S^{-1}$ (*verify!*), or

$$A_{2D}^{-1} \;=\; (S_y \otimes S_x)(I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)^{-1}(S_y^{-1} \otimes S_x^{-1}).$$

- Notice that $\Lambda := (I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)$ is diagonal and easily inverted.

- The solution to $A_{2D}\mathbf{u} = \mathbf{f}$ is thus

$$\mathbf{u} \;=\; (S_y \otimes S_x)(I_y \otimes \Lambda_x \;+\; \Lambda_y \otimes I_x)^{-1}(S_y^{-1} \otimes S_x^{-1})\mathbf{f}.$$

- In fast matrix-matrix product form, this has a particularly compact expression:

$$U \;=\; S_x[D \circ (S_x^{-1} F S_y^{-T})]S_y^T,$$

  where $W = D \circ V$ is used to denote *pointwise* multiplication of the entries of the matrix pair (D, V). That is, $w_{ij} := d_{ij} * v_{ij}$.

- Note that, for the particular 1D A_x and A_y matrices in this example that the eigenvectors are orthogonal. If we normalize the columns, then $S_x^{-1} = S_x^T$ (same for y).

Computing $||A||_2$ and $\mathbf{cond}_2(A)$.

- Recall:

$$\begin{aligned} \mathrm{cond}(A) &:= \|\, A^{-1} \,\| \cdot \|\, A \,\|, \\ \|\, A \,\| &:= \max_{\mathbf{x} \neq 0} \frac{\|\, A\mathbf{x} \,\|}{\|\, \mathbf{x} \,\|}, \\ \|\, \mathbf{x} \,\|_2 &= \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}} = \sqrt{\mathbf{x}^T \mathbf{x}}, \\ \|\, \mathbf{x} \,\|_2^2 &= \mathbf{x}^T \mathbf{x}. \end{aligned}$$

- From now on, drop the subscript "$_2$".

$$\begin{aligned} \|\, \mathbf{x} \,\|^2 &= \mathbf{x}^T \mathbf{x} \\ \|\, A\mathbf{x} \,\|^2 &= (A\mathbf{x})^T (A\mathbf{x}) = \mathbf{x}^T A^T A\, \mathbf{x}. \end{aligned}$$

- Matrix norm:

$$\begin{aligned} ||\, A \,||^2 &= \max_{\mathbf{x}\neq 0} \frac{||\, A\mathbf{x} \,||^2}{||\, \mathbf{x} \,||^2}, \\ &= \max_{\mathbf{x}\neq 0} \frac{\mathbf{x}^T A^T A\, \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \\ &= \lambda_{\max}(A^T A) \;=:\; \text{spectral radius of } (A^T A). \end{aligned}$$

- The symmetric positive definite matrix $B := A^T A$ has positive eigenvalues.

- All symmetric matrices B have a complete set of orthonormal eigenvectors satisfying

$$B\mathbf{z}_j \;=\; \lambda_j\, \mathbf{z}_j, \quad \mathbf{z}_i^T \mathbf{z}_j \;=\; \delta_{ij} \;=\; \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} .$$

- **Note:** If $\lambda_i = \lambda_j$, $i \neq j$, then can have $\mathbf{z}_i^T \mathbf{z}_j \neq 0$, but we can orthogonalize $\mathbf{z}_i$ and $\mathbf{z}_j$ so that $\tilde{\mathbf{z}}_i^T \tilde{\mathbf{z}}_j = 0$ and

$$\begin{aligned} B\tilde{\mathbf{z}}_i &= \lambda_i \tilde{\mathbf{z}}_i \qquad \lambda_i = \lambda_j \\ B\tilde{\mathbf{z}}_j &= \lambda_j \tilde{\mathbf{z}}_j. \end{aligned}$$

- Assume eigenvalues are sorted with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.
- For any $\mathbf{x}$ we have: $\mathbf{x} = c_1\mathbf{z}_1 + c_2\mathbf{z}_2 + \cdots + c_n\mathbf{z}_n$.
- Let $||\,\mathbf{x}\,|| = 1$.
- Want to find $\displaystyle \max_{||\,\mathbf{x}\,||=1} \frac{\mathbf{x}^T B\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \max_{||\,\mathbf{x}\,||=1} \mathbf{x}^T B\mathbf{x}$.
- Note:

$$
\begin{aligned}
\mathbf{x}^T\mathbf{x} &= \left(\sum_{i=1}^{n} c_i\mathbf{z}_i\right)^T \left(\sum_{j=1}^{n} c_j\mathbf{z}_j\right) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i\, c_j \mathbf{z}_i^T\mathbf{z}_j \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i\, c_j \delta_{ij} \\
&= \sum_{i=1}^{n} c_i^2 \;=\; 1.
\end{aligned}
$$

$$
\Longrightarrow\; c_1^2 \;=\; 1 - \sum_{i=2}^{n} c_i^2.
$$

$$
\begin{aligned}
\mathbf{x}^T B \mathbf{x} &= \left(\sum_{i=1}^n c_i \mathbf{z}_i \right)^T \left(\sum_{j=1}^n c_j B \mathbf{z}_j \right) \\
&= \left(\sum_{i=1}^n c_i \mathbf{z}_i \right)^T \left(\sum_{j=1}^n c_j \lambda_j \mathbf{z}_j \right) \\
&= \sum_{i=1}^n \sum_{j=1}^n c_i \, \lambda_j c_j \mathbf{z}_i^T \mathbf{z}_j \\
&= \sum_{i=1}^n \sum_{j=1}^n c_i \, \lambda_j c_j \delta_{ij} \\
&= \sum_{i=1}^n c_i^2 \lambda_i \;=\; c_1^2 \lambda_1 \,+\, c_2^2 \lambda_2 \,+\, \cdots \,+\, c_n^2 \lambda_n \\
&= \lambda_1 \left[c_1^2 \,+\, c_2^2 \beta_2 \,+\, \cdots \,+\, c_n^2 \beta_n \right], \quad 0 \,<\, \beta_i \,:=\, \frac{\lambda_i}{\lambda_1} \,\leq\, 1, \\
&= \lambda_1 \left[(1 - c_2^2 - \cdots - c_n^2) \,+\, c_2^2 \beta_2 \,+\, \cdots \,+\, c_n^2 \beta_n \right] \\
&= \lambda_1 \left[1 \,-\, (1-\beta_2) c_2^2 \,+\, (1-\beta_3) c_3^2 \,+\, \cdots \,+\, (1-\beta_n) c_n^2 \right] \\
&= \lambda_1 \left[1 \,-\, \text{some positive (or zero) numbers} \right].
\end{aligned}
$$

- Expression is maximized when $c_2 = c_3 = \cdots = c_n = 0, \implies c_1 = 1$.
- Maximum value $\mathbf{x}^T B \mathbf{x} = \lambda_{\max}(B) = \lambda_1$.
- Similarly, can show min $\mathbf{x}^T B \mathbf{x} = \lambda_{\min}(B) = \lambda_n$.

- So, $||A||^2 = \max_\lambda \lambda(A^TA) =$ spectral radius of A^TA.

- Now,
$$|| A^{-1} ||^2 \;=\; \max_{\mathbf{x}\neq 0} \frac{|| A^{-1}\mathbf{x} ||^2}{||\mathbf{x}||^2}.$$

- Let $\mathbf{x} = A\mathbf{y}$:
$$\begin{aligned} || A^{-1} ||^2 \;&=\; \max_{\mathbf{y}\neq 0} \frac{|| A^{-1}A\mathbf{y} ||^2}{||A\mathbf{y}||^2} \;=\; \max_{\mathbf{y}\neq 0} \frac{|| \mathbf{y} ||^2}{||A\mathbf{y}||^2} \;=\; \left(\min_{\mathbf{y}\neq 0} \frac{|| A\mathbf{y} ||^2}{||\mathbf{y}||^2} \right)^{-1} \\ &=\; \frac{1}{\lambda_{\min}(A^TA)}. \end{aligned}$$

- So, $\mathrm{cond}_2(A) = || A^{-1} || \cdot || A ||$,
$$\mathrm{cond}_2(A) \;=\; \sqrt{\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}}.$$

Special Types of Linear Systems

- Work and storage can often be saved in solving linear system if matrix has special properties
- Examples include
  - *Symmetric*: $\boldsymbol{A} = \boldsymbol{A}^T$, $a_{ij} = a_{ji}$ for all i, j
  - *Positive definite*: $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$
  - *Band*: $a_{ij} = 0$ for all $|i - j| > \beta$, where β is *bandwidth* of $\boldsymbol{A}$
  - *Sparse*: most entries of $\boldsymbol{A}$ are zero

Symmetric Positive Definite (SPD) Matrices

- Very common in optimization and physical processes
- Easiest example:
  - If B is invertible, then $A := B^T B$ is SPD.
- SPD systems of the form $A \underline{x} = \underline{b}$ can be solved using
  - (stable) Cholesky factorization $A = LL^T$, or
  - iteratively with the most robust iterative solver, conjugate gradient iteration (generally with preconditioning, known as preconditioned conjugate gradients, PCG).

Cholesky Factorization and SPD Matrices.

- A is SPD: $A = A^T$ and $\mathbf{x}^T A\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.
- Seek a symmetric factorization $A = \tilde{L}\tilde{L}^T$ (not LU).
  - L *not* lower triangular but not *unit* lower triangular.
  - That is, Lt_{ii} not necessarily 1.
- Alternatively, seek factorization $A = LDL^T$, where L *is* unit lower triangular and D is *diagonal*.

- Start with $LDL^T = A$.
- Clearly, $LU = A$ with $U = DL^T$.
  - Follows from uniqueness of LU factorization.
  - D is a *row scaling* of L^T and thus $D_{ii} = U_{ii}$.
  - A property of SPD matrices is that all pivots are positive.
  - (Another property is that you do not need to pivot.)

- Consider standard update step:

$$\begin{aligned} a_{ij} &= a_{ij} - \frac{a_{ik}\,a_{kj}}{a_{kk}} \\ &= a_{ij} - \frac{a_{ik}\,a_{jk}}{a_{kk}} \end{aligned}$$

- Usual multiplier column entries are $l_{ik} = a_{ik}/a_{kk}$.
- Usual pivot row entries are $u_{kj} = a_{kj} = a_{jk}$.
- So, if we factor $1/d_{kk} = 1/a_{kk}$ out of U, we have:

$$\begin{aligned} d_{kk}(a_{kj}/a_{kk}) &= d_{kk} l_{kj} \\ \longrightarrow U &= D(D^{-1}U) \\ &= DL^T. \end{aligned}$$

- For Cholesky, we have

$$A \;=\; LDL^T = L\sqrt{D}\sqrt{D}L^T = \tilde{L}\tilde{L}^T,$$

with $\tilde{L} = L\sqrt{D}$.

Symmetric Positive Definite Matrices

- If $\boldsymbol{A}$ is symmetric and positive definite, then LU factorization can be arranged so that $\boldsymbol{U} = \boldsymbol{L}^T$, which gives *Cholesky factorization*

$$\boxed{\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{L}^T}$$

  where $\boldsymbol{L}$ is lower triangular with positive diagonal entries
- Algorithm for computing it can be derived by equating corresponding entries of $\boldsymbol{A}$ and $\boldsymbol{L}\boldsymbol{L}^T$
- 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

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

Cholesky Factorization (Text)

Algorithm 2.7 Cholesky Factorization

```
for k = 1 to n                              { loop over columns }
    a_kk = sqrt(a_kk)
    for i = k+1 to n
        a_ik = a_ik / a_kk                  { scale current column }
    end
    for j = k+1 to n                        { from each remaining column,
        for i = j to n                          subtract multiple
            a_ij = a_ij - a_ik · a_jk           of current column }
        end
    end
end
```

After a row scaling, this is just standard LU decomposition, exploiting symmetry in the LU factors and A. ($U=L^T$)

Cholesky Factorization

- One way to write resulting general algorithm, in which Cholesky factor $\boldsymbol{L}$ overwrites original matrix $\boldsymbol{A}$, is

for $j = 1$ **to** n
    for $k = 1$ **to** $j - 1$
        for $i = j$ **to** n
            $a_{ij} = a_{ij} - a_{ik} \cdot a_{jk}$
        end
    end
    $a_{jj} = \sqrt{a_{jj}}$
    for $k = j + 1$ **to** n
        $a_{kj} = a_{kj} / a_{jj}$
    end
end

Cholesky Factorization, continued

- Features of Cholesky algorithm for symmetric positive definite matrices
  - All n square roots are of positive numbers, so algorithm is well defined
  - No pivoting is required to maintain numerical stability
  - Only lower triangle of $\boldsymbol{A}$ is accessed, and hence upper triangular portion need not be stored
  - Only $n^3/6$ multiplications and similar number of additions are required
- Thus, Cholesky factorization requires only about half work and ***half storage*** compared with LU factorization of general matrix by Gaussian elimination, and also avoids need for pivoting

Linear Algebra Very Short Summary

Main points:

- Conditioning of matrix cond(A) bounds our expected accuracy.
  - e.g., if cond(A) ~ 10^5 we expect at most 11 significant digits in $\underline{x}$.
  - Why?
  - We start with IEEE double precision – 16 digits. We lose 5 because condition (A) ~ 10^5, so we have 11 = 16-5.

- Stable algorithm (i.e., pivoting) important to realizing this bound.
  - Some systems don't need pivoting (e.g., SPD, diagonally dominant)
  - Unstable algorithms can sometimes be rescued with iterative refinement.

- Costs:
  - Full matrix → $O(n^2)$ storage, $O(n^3)$ work (wall-clock time)
  - Sparse or banded matrix, substantially less.

- The following slides present the book's derivation of the LU factorization process.

- I'll highlight a few of them that show the equivalence between the outer product approach and the elementary elimination matrix approach.

Example: Triangular Linear System

$$\begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix}$$

- Using back-substitution for this upper triangular system, last equation, $4x_3 = 8$, is solved directly to obtain $x_3 = 2$
- Next, x_3 is substituted into second equation to obtain $x_2 = 2$
- Finally, both x_3 and x_2 are substituted into first equation to obtain $x_1 = -1$

Elimination

- To transform general linear system into triangular form, we need to replace selected nonzero entries of matrix by zeros
- This can be accomplished by taking linear combinations of rows
- Consider 2-vector $\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$
- If $a_1 \neq 0$, then

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

Elementary Elimination Matrices

- More generally, we can annihilate *all* entries below kth position in n-vector $\boldsymbol{a}$ by transformation

$$\boldsymbol{M}_k\boldsymbol{a} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_n & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ a_{k+1} \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $m_i = a_i/a_k$, $i = k+1, \ldots, n$

- Divisor a_k, called *pivot*, must be nonzero

Elementary Elimination Matrices, continued

- Matrix $\boldsymbol{M}_k$, called *elementary elimination matrix*, adds multiple of row k to each subsequent row, with *multipliers* m_i chosen so that result is zero
- $\boldsymbol{M}_k$ is unit lower triangular and nonsingular
- $\boldsymbol{M}_k = \boldsymbol{I} - \boldsymbol{m}_k \boldsymbol{e}_k^T$, where $\boldsymbol{m}_k = [0, \ldots, 0, m_{k+1}, \ldots, m_n]^T$ and $\boldsymbol{e}_k$ is kth column of identity matrix
- $\boldsymbol{M}_k^{-1} = \boldsymbol{I} + \boldsymbol{m}_k \boldsymbol{e}_k^T$, which means $\boldsymbol{M}_k^{-1} =: \boldsymbol{L}_k$ is same as $\boldsymbol{M}_k$ except signs of multipliers are reversed

Elementary Elimination Matrices, continued

- If $\boldsymbol{M}_j$, $j > k$, is another elementary elimination matrix, with vector of multipliers $\boldsymbol{m}_j$, then

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

  which means product is essentially “union,” and similarly for product of inverses, $\boldsymbol{L}_k \boldsymbol{L}_j$

Comment on update step and $\underline{m}_k \underline{e}^T_k$

- Recall, $\underline{v} = C\,\underline{w} \in \text{span}\{C\}$.
- $\therefore\; V = (\,\underline{v}_1\, \underline{v}_2 \ldots \underline{v}_n) = C\,(\,\underline{w}_1\, \underline{w}_2 \ldots \underline{w}_n) \in \text{span}\{C\}$.

- If $C = \underline{c}$, i.e., C is a column vector and therefore of rank 1, then V is in span{C} and is of rank 1.

- All columns of V are multiples of $\underline{c}$.

- Thus, $W = \underline{c}\; \underline{r}^T$ is an n x n matrix of rank 1.
  - All columns are multiples of the first column and
  - All rows are multiples of the first row.

Elementary Elimination Matrices, continued

- Matrix $\boldsymbol{M}_k$, called *elementary elimination matrix*, adds multiple of row k to each subsequent row, with *multipliers* m_i chosen so that result is zero
- $\boldsymbol{M}_k$ is unit lower triangular and nonsingular
- $\boldsymbol{M}_k = \boldsymbol{I} - \boldsymbol{m}_k \boldsymbol{e}_k^T$, where $\boldsymbol{m}_k = [0, \ldots, 0, m_{k+1}, \ldots, m_n]^T$ and $\boldsymbol{e}_k$ is kth column of identity matrix
- $\boldsymbol{M}_k^{-1} = \boldsymbol{I} + \boldsymbol{m}_k \boldsymbol{e}_k^T$, which means $\boldsymbol{M}_k^{-1} =: \boldsymbol{L}_k$ is same as $\boldsymbol{M}_k$ except signs of multipliers are reversed

Example: Elementary Elimination Matrices

- For $\boldsymbol{a} = \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix}$,

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

and

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

Example, continued

- Note that

$$\boldsymbol{L}_1 = \boldsymbol{M}_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{L}_2 = \boldsymbol{M}_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{bmatrix}$$

and

$$\boldsymbol{M}_1\boldsymbol{M}_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 1/2 & 1 \end{bmatrix}, \quad \boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1/2 & 1 \end{bmatrix}$$

Gaussian Elimination

- To reduce general linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to upper triangular form, first choose $\boldsymbol{M}_1$, with a_{11} as pivot, to annihilate first column of $\boldsymbol{A}$ below first row
  - System becomes $\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_1\boldsymbol{b}$, but solution is unchanged
- Next choose $\boldsymbol{M}_2$, using a_{22} as pivot, to annihilate second column of $\boldsymbol{M}_1\boldsymbol{A}$ below second row
  - System becomes $\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}$, but solution is still unchanged
- Process continues for each successive column until all subdiagonal entries have been zeroed

Gaussian Elimination

- To reduce general linear system $\boldsymbol{Ax} = \boldsymbol{b}$ to upper triangular form, first choose $\boldsymbol{M}_1$, with a_{11} as pivot, to annihilate first column of $\boldsymbol{A}$ below first row
  - System becomes $\boldsymbol{M}_1\boldsymbol{Ax} = \boldsymbol{M}_1\boldsymbol{b}$, but solution is unchanged
- Next choose $\boldsymbol{M}_2$, using a_{22} as pivot, to annihilate second column of $\boldsymbol{M}_1\boldsymbol{A}$ below second row
  - System becomes $\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{Ax} = \boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}$, but solution is still unchanged
- *Technically, this should be a'_{22} , the 2-2 entry in $A' := M_1A$. Thus, we don't know all the pivots in advance.*

Gaussian Elimination, continued

- Resulting upper triangular linear system

$$\begin{aligned} \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1 \boldsymbol{A} \boldsymbol{x} &= \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_1 \boldsymbol{b} \\ \boldsymbol{M} \boldsymbol{A} \boldsymbol{x} &= \boldsymbol{M} \boldsymbol{b} \end{aligned}$$

  can be solved by back-substitution to obtain solution to original linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$

- Process just described is called *Gaussian elimination*

LU Factorization

- Product $\boldsymbol{L}_k \boldsymbol{L}_j$ is unit lower triangular if $k < j$, so

$$\boldsymbol{L} = \boldsymbol{M}^{-1} = \boldsymbol{M}_1^{-1} \cdots \boldsymbol{M}_{n-1}^{-1} = \boldsymbol{L}_1 \cdots \boldsymbol{L}_{n-1}$$

  is unit lower triangular

- By design, $\boldsymbol{U} = \boldsymbol{M}\boldsymbol{A}$ is upper triangular

- So we have

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$$

  with $\boldsymbol{L}$ unit lower triangular and $\boldsymbol{U}$ upper triangular

- Thus, Gaussian elimination produces *LU factorization* of matrix into triangular factors

LU Factorization, continued

- Having obtained LU factorization, $\boldsymbol{Ax} = \boldsymbol{b}$ becomes $\boldsymbol{LUx} = \boldsymbol{b}$, and can be solved by forward-substitution in lower triangular system $\boldsymbol{Ly} = \boldsymbol{b}$, followed by back-substitution in upper triangular system $\boldsymbol{Ux} = \boldsymbol{y}$

- Note that $\boldsymbol{y} = \boldsymbol{Mb}$ is same as transformed right-hand side in Gaussian elimination

- Gaussian elimination and LU factorization are two ways of expressing same solution process

Example: Gaussian Elimination

- Use Gaussian elimination to solve linear system

$$\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \boldsymbol{b}$$

- To annihilate subdiagonal entries of first column of $\boldsymbol{A}$,

$$\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix},$$

$$\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$

Example, continued

- To annihilate subdiagonal entry of second column of $\boldsymbol{M}_1\boldsymbol{A}$,

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = \boldsymbol{U},$$

$$\boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix} = \boldsymbol{M}\boldsymbol{b}$$

Example, continued

- We have reduced original system to equivalent upper triangular system

$$\boldsymbol{U}\boldsymbol{x} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix} = \boldsymbol{M}\boldsymbol{b}$$

which can now be solved by back-substitution to obtain

$$\boldsymbol{x} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

Example, continued

- To write out LU factorization explicitly,

$$\boldsymbol{L}_1\boldsymbol{L}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} = \boldsymbol{L}$$

so that

$$\boldsymbol{A} = \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = \boldsymbol{LU}$$