# Computational Astrophysics

ASTR 660, Spring 2021
計算天文物理

**Lecture 5 Linear Systems**

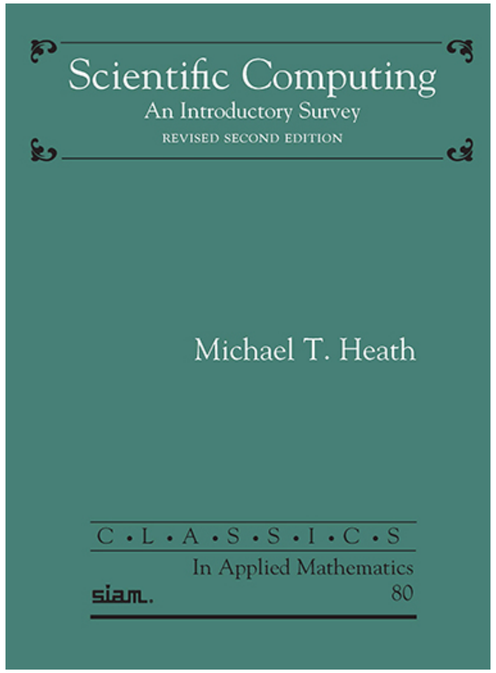
Instructor: Prof. Kuo-Chuan Pan
kuochuan.pan@gapp.nthu.edu.tw

# Class website

https://kuochuanpan.github.io/courses/109ASTR660_CA/

# Plan for today

- Introduction to linear systems
- Solving linear systems




**References:**
"Scientific Computing: An introductory survey", Michael Heath

https://books.google.com.tw/books?id=f6Z8DwAAQBAJ&hl=zh-TW


# Introduction to Linear Systems

# Linear System


The most basic task in linear algebra (all scientific computing), is to solve for the unknowns in a set of linear algebraic equations.

$$a_{11}x_1+a_{12}x_2 + ... + a_{1N}x_N = b_1$$
$$a_{21}x_1+a_{22}x_2 + ... + a_{2N}x_N = b_2$$
$$...$$
$$a_{M1}x_1+a_{M2}x_2 + ... + a_{MN}x_N = b_M$$

# Linear System




The equations can be written in matrix form as

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{1N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{1N} \\ \dots & & & & \\ a_{M1} & a_{M2} & a_{M3} & \dots & a_{MN} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_M \end{bmatrix}$$

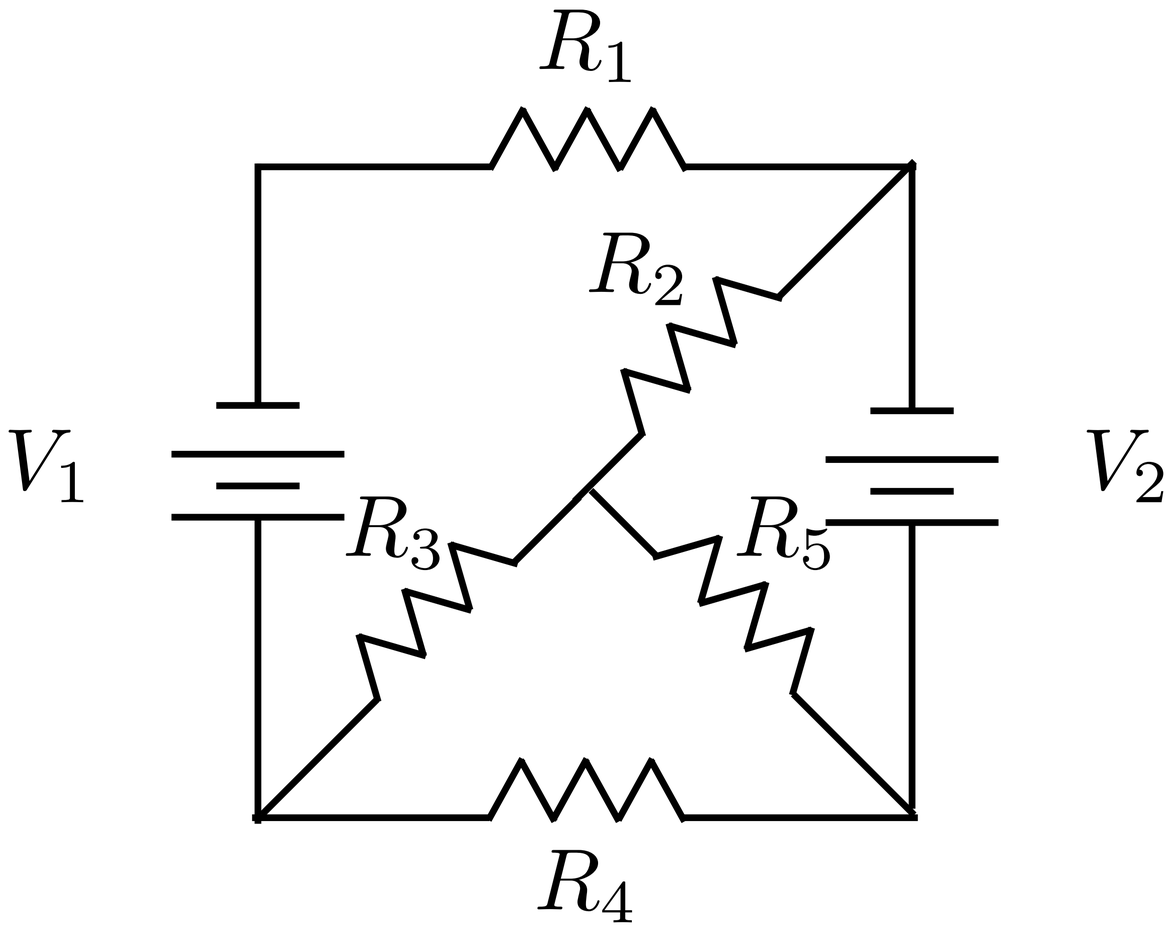
# Linear System



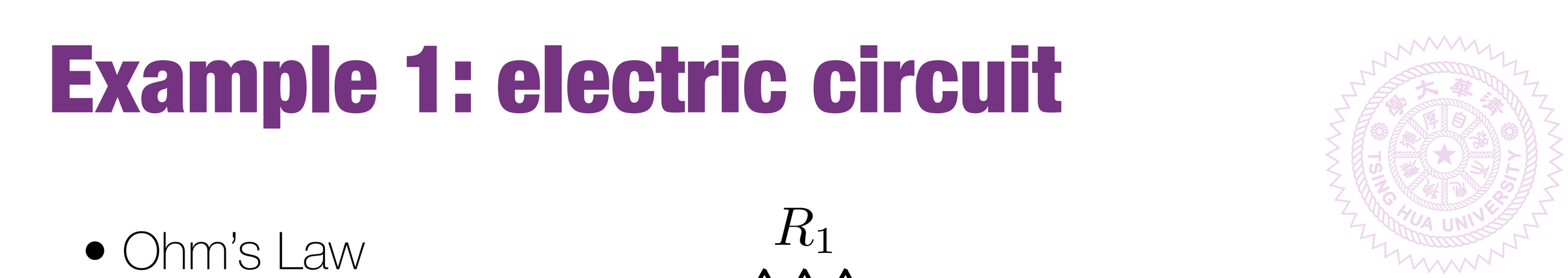

The equations can be written in matrix form as

Or $$A \cdot x = b$$

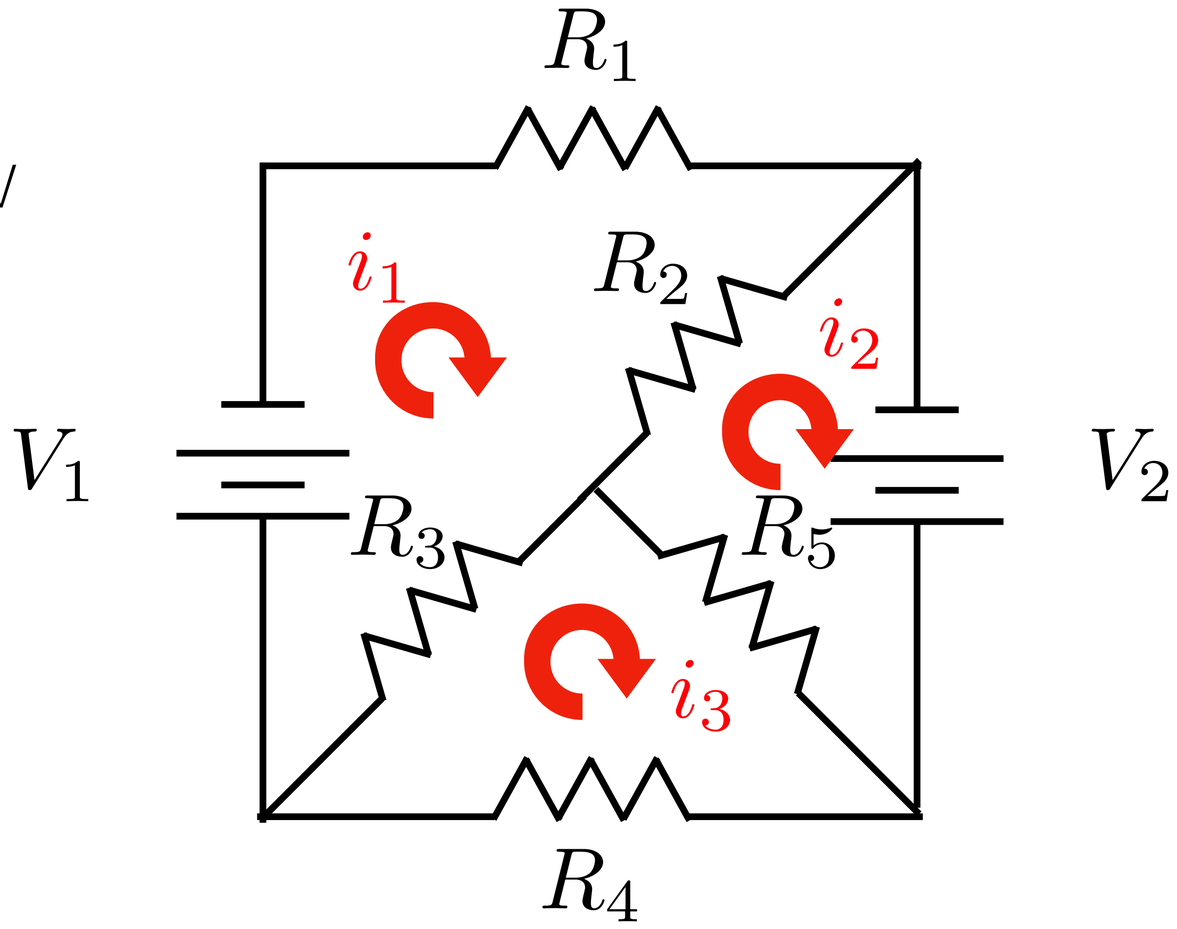
# Example 1: electric circuit



# Example 1: electric circuit



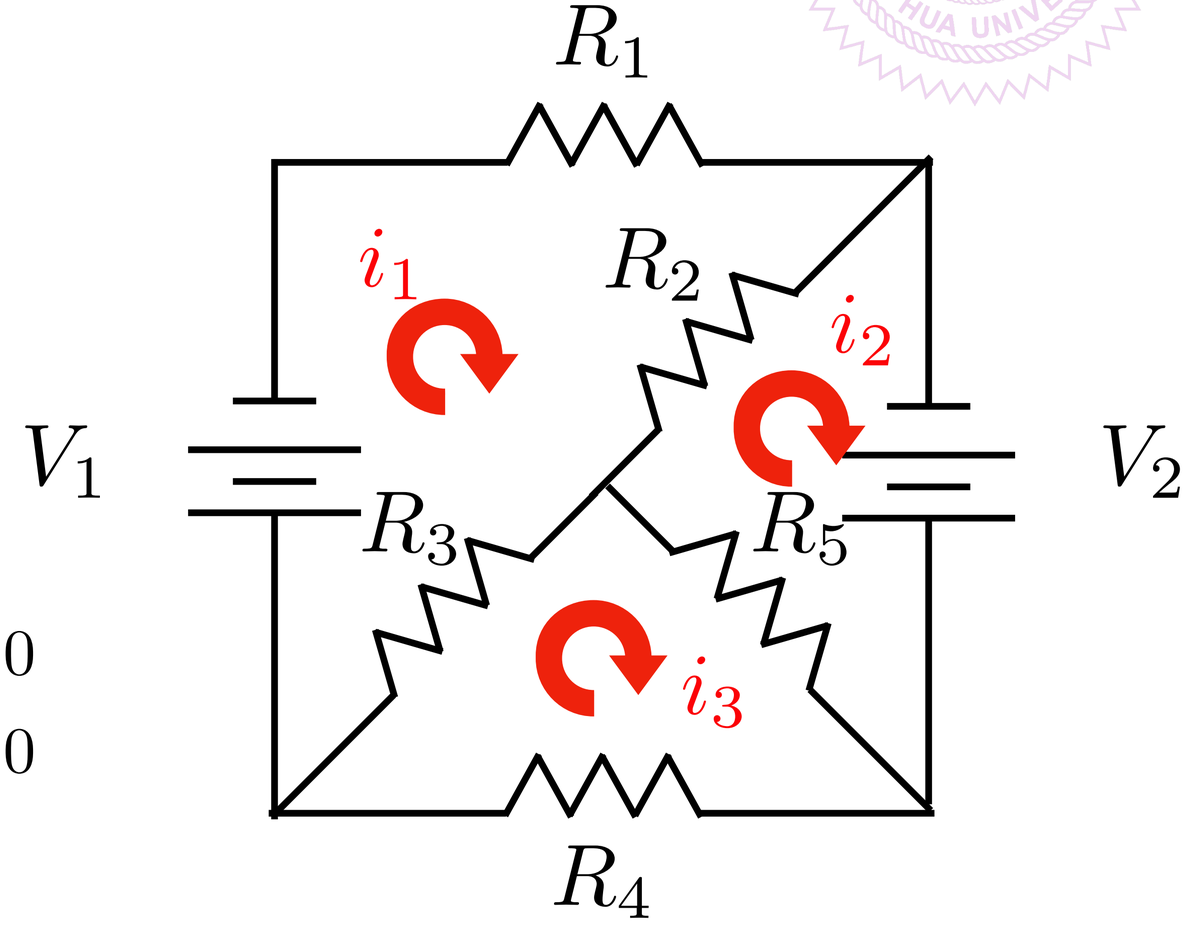
- Ohm's Law
- Kirchhoff's Law



# Example 1: electric circuit



- Ohm's Law
- Kirchhoff's Law

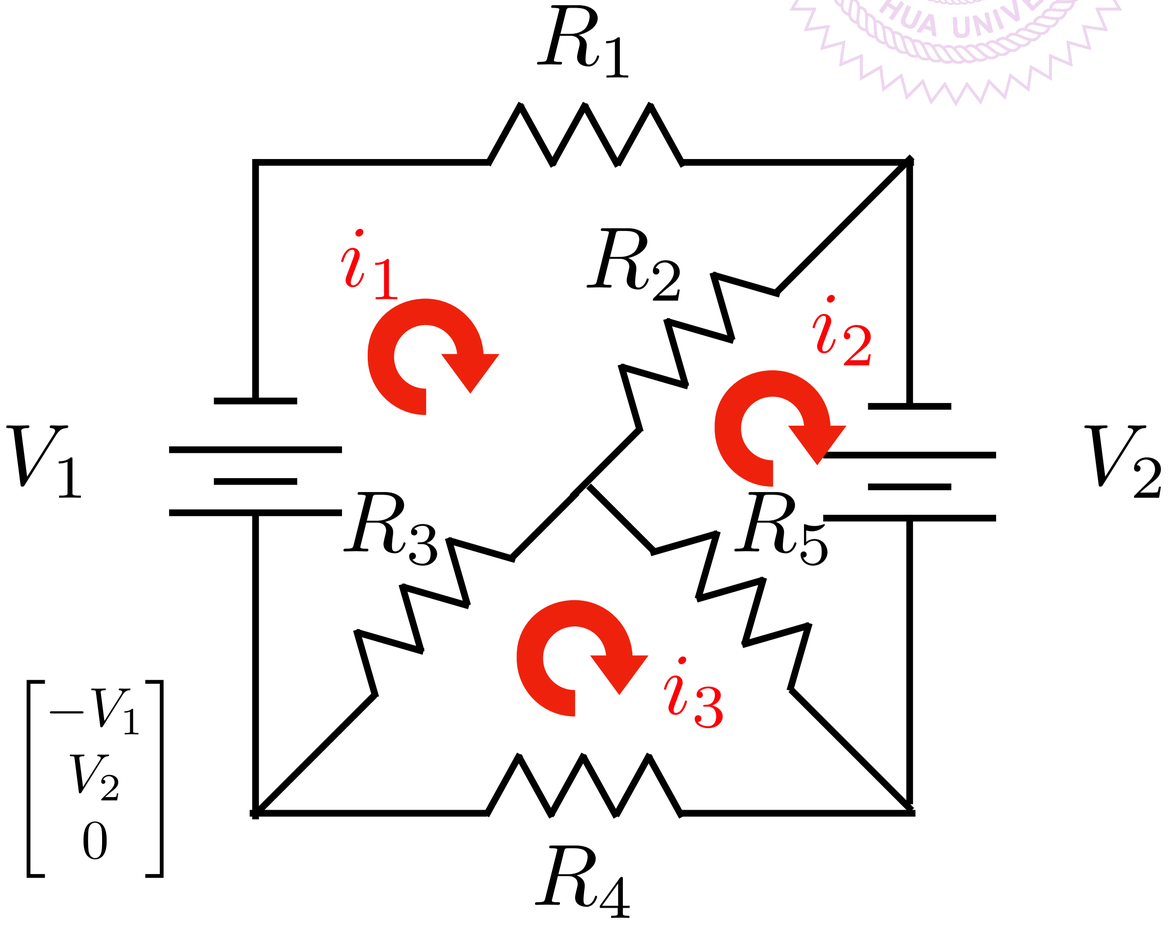



$$
\begin{aligned}
i_1 R_1 + (i_1 - i_2) R_2 + (i_1 - i_3) R_3 + V_1 &= 0 \\
(i_2 - i_1) R_2 + (i_2 - i_3) R_5 - V_2 &= 0 \\
(i_3 - i_1) R_3 + i_3 R_4 + (i_3 - i_2) R_5 &= 0
\end{aligned}
$$

# Example 1: electric circuit



- Ohm's Law
- Kirchhoff's Law




$$\begin{bmatrix} R_1+R_2+R_3 & -R_2 & -R_3 \\ -R_2 & R_2+R_5 & -R_5 \\ -R_3 & -R_5 & R_3+R_4+R_5 \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} -V_1 \\ V_2 \\ 0 \end{bmatrix}$$

# Example 2: Special Relativity



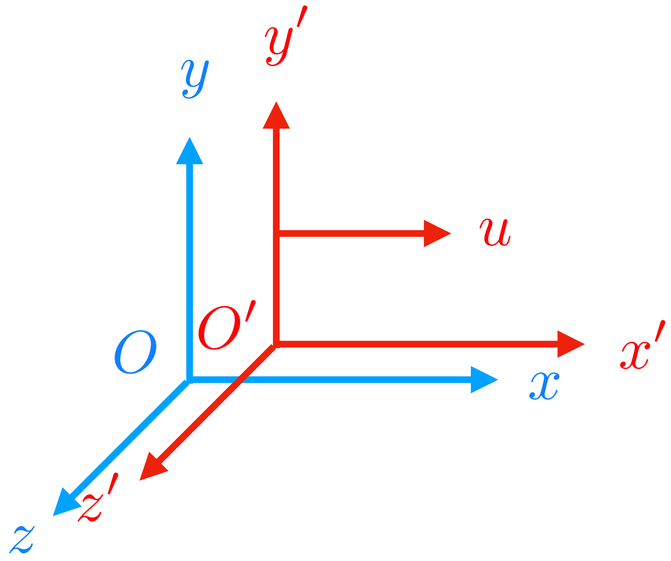
- The principle of relativity
- The constancy of light speed

$$t' = \gamma t - \gamma u/c^2 x$$

$$x' = \gamma x - \gamma u t$$

$$y' = y$$

$$z' = z$$



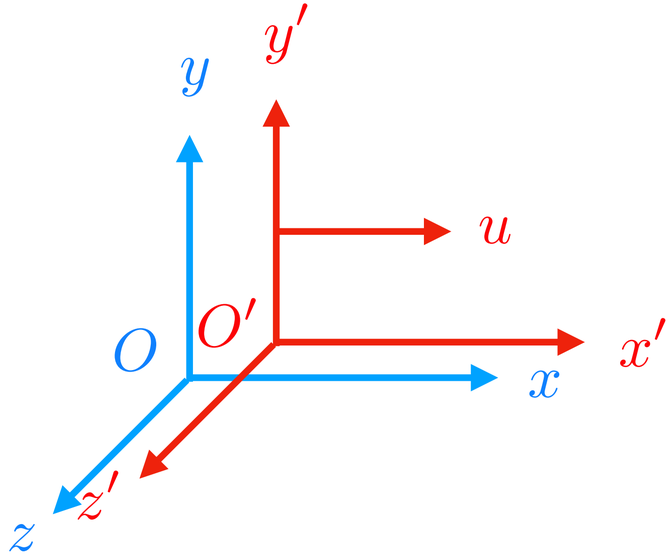

$$\gamma = \frac{1}{\sqrt{1 - u^2/c^2}}$$

# Example 2: Special Relativity



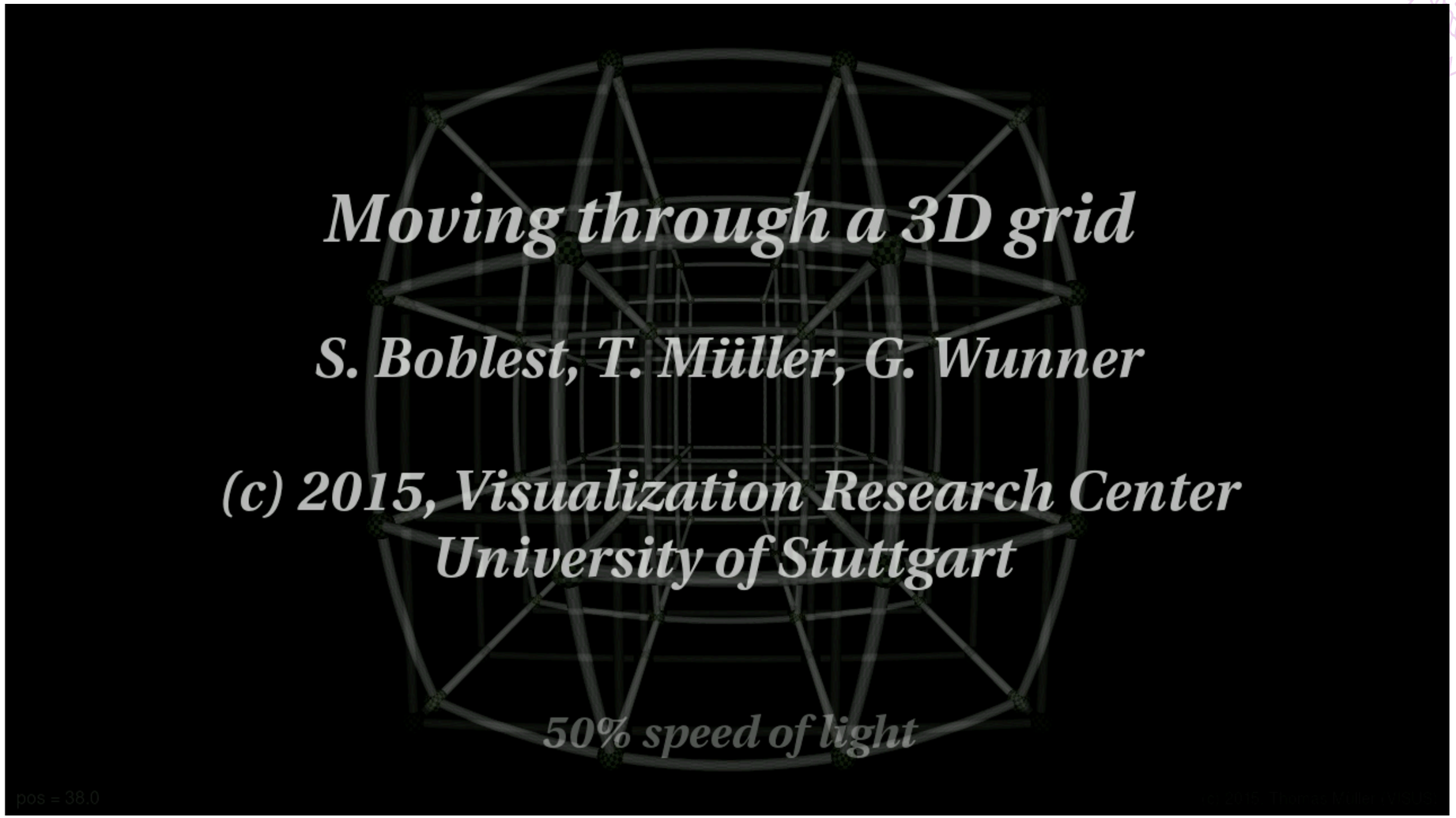

- The principle of relativity
- The constancy of light speed

$$\begin{bmatrix} \gamma & \gamma u/c^2 & 0 & 0 \\ \gamma u & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} t' \\ x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t \\ x \\ y \\ z \end{bmatrix}$$




$$\gamma = \frac{1}{\sqrt{1 - u^2/c^2}}$$

# Example 2: Special Relativity



# Linear Systems




`numpy/scipy` could do all the work for us,
but if we treat it as a black box ….

See linear_system.ipynb


# Math: Linear Systems

# Linear System




A is a M x N matrix.

If M=N, then there are as many equations as unknowns, and there is a good "chance" of solving for a unique solution set of $x_i$'s

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

# Linear System

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

Unique solution requires **A** is non-singular:

1. A has an inverse
2. det(A) is not zero
3. Rank(A) = N
4. For any vector z != 0, Az != 0

# Linear System




A is a M x N matrix.

If M<N, or if M=N but the equations are degenerate. In this case there can be either

1. No solution, or
2. Infinity solutions

If M>N, more equations than unknowns, there is in general no solution —> Overdetermined.

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

# Example: Non-singularity




Consider a linear system,

$$Ax = \begin{bmatrix} 2 & 3 \\ 5 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b$$

has a unique solution regardless of the values of b, since A is non-singular.

# Example: Singularity




Consider a linear system,

$$Ax = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b$$

has a singular matrix A. Depending on the values of b, it could be no solution or infinity solutions.
For example, if b = [8  14].T, then there is no solution.
If b = [4  8].T, then x = [c  (4-2c)/3].T is a solution for any real number c.

# Sensitivity and conditioning


Before solving the linear systems, we need to understand the sensitivity of the solution to perturbation of input data.

To measure such perturbation, the concept of “magnitude”, “magnitude”, or “modulus” can be generalized to the concept of norms for vectors and matrices.

# Vector Norms




p-norms

$$||x||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

# Vector Norms




p-norms

$$||x||_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Special cases are: first, second, and infinity norms

$$||x||_1 = \sum_{i=1}^{n} |x_i|, \qquad ||x||_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \qquad ||x||_\infty = \max_{1 \le i \le n} |x_i|,$$

“Manhatten norm” “Euclidean norm”

# Vector Norms




In general, for any n-vector

$$||x||_1 \leq ||x||_2 \leq ||x||_\infty$$

$$||x||_1 \leq \sqrt{n}||x||_2,\ ||x||_2 \leq \sqrt{n}||x||_\infty,\ \text{and}\ ||x||_1 \leq n||x||_\infty$$

For a given n, norms differ by at most a constant, and hence are equivalent.
Therefore, we can use whichever norm is most convenient in given context

# Properties of Vector Norms




For any vector norm:

$$||x|| > 0 \text{ if } x \neq 0$$

$$||\gamma x|| = |\gamma| \cdot ||x|| \text{ for any scalar } \gamma$$

$$||x + y|| \leq ||x|| + ||y|| \text{ (triangle inequality)}$$

$$\text{or } ||x|| - ||y|| \leq ||x - y|| \text{ (triangle inequality)}$$

# Matrix Norms

We also need some way to measure the size or magnitude of matrices.

Matrix norm is “induced” by a given vector norm x

$$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

# Matrix Norms




Matrix norm is “induced” by a given vector norm x

$$||A|| = \max_{x \neq 0} \frac{||Ax||}{||x||}$$

$$||A||_1 = \max_j \sum_{i=1}^{n} |a_{ij}|,$$

Maximum absolute column sum

$$||A||_\infty = \max_i \sum_{j=1}^{n} |a_{ij}|,$$

Maximum absolute row sum

The matrix norm corresponding to the vector 2-norm is not so easy to compute (depending on the knowledge of eigenvalues and optimization)

# Properties of Matrix Norms




Any matrix norm satisfies

$\|\boldsymbol{A}\| > 0$ if $\boldsymbol{A} \neq \boldsymbol{0}$

$\|\gamma \boldsymbol{A}\| = |\gamma| \cdot \|\boldsymbol{A}\|$ for any scalar $\gamma$

$\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$

$\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{B}\|$

$\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|$ for any vector $\boldsymbol{x}$

May not hold for more general matrix norms

# Matrix condition number




Condition number

$$\mathrm{cond}(A) = ||A|| \cdot ||A^{-1}||$$

By convenient, if A is singular: $\mathrm{cond}(A) = \infty$

Larger condition number implies nearly singular

# Compute condition number




- Involves matrix inverse, it is non-trivial to compute.
- Matrix norm is easily computed as maximum absolute column sum (row sum)
- Estimating $||A^{-1}||$ at low cost is more challenging

# Compute condition number




- From properties of norms, if $\boldsymbol{Az} = \boldsymbol{y}$, then

$$\frac{\|\boldsymbol{z}\|}{\|\boldsymbol{y}\|} \leq \|\boldsymbol{A}^{-1}\|$$

  This bound is achieved for optimally chosen **y**
- Good library packages for linear systems provide efficient and reliable condition estimator
- Condition number is useful in assessing accuracy of approximate solution

# Error bounds




Let x be solution to $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$, and let $\hat{\boldsymbol{x}}$ be solution to $\boldsymbol{A} \cdot \hat{\boldsymbol{x}} = \boldsymbol{b} + \Delta\boldsymbol{b}$

If $\Delta\boldsymbol{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$, then

$$\boldsymbol{b} + \Delta\boldsymbol{b} = \boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}\Delta\boldsymbol{x}$$

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

$$\Delta\boldsymbol{b} = \boldsymbol{A}\Delta\boldsymbol{x}$$

# Error bounds

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b} \longrightarrow ||\boldsymbol{b}|| = ||\boldsymbol{A} \cdot \boldsymbol{x}|| \leq ||\boldsymbol{A}|| \cdot ||\boldsymbol{x}||$$

$$\Delta \boldsymbol{b} = \boldsymbol{A} \Delta \boldsymbol{x} \longrightarrow ||\Delta \boldsymbol{x}|| = ||\boldsymbol{A}^{-1} \Delta \boldsymbol{b}|| \leq ||\boldsymbol{A}^{-1}|| \cdot ||\Delta \boldsymbol{b}||$$

$$\frac{||\Delta \boldsymbol{x}||}{||\boldsymbol{x}||} \leq ||\boldsymbol{A}^{-1}|| \cdot ||\Delta \boldsymbol{b}|| \frac{||\boldsymbol{A}||}{||\boldsymbol{b}||}$$

$$\boxed{\frac{||\Delta \boldsymbol{x}||}{||\boldsymbol{x}||} \leq \mathrm{cond}(\boldsymbol{A}) \frac{||\Delta \boldsymbol{b}||}{||\boldsymbol{b}||}}$$

# Error bounds




Similar result holds for relative changes in the entries of the matrix A.

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$$

$$(\boldsymbol{A} + \boldsymbol{E}) \cdot \hat{\boldsymbol{x}} = \boldsymbol{b}$$

$$\frac{||\Delta \boldsymbol{x}||}{||\hat{\boldsymbol{x}}||} \leq \mathrm{cond}(\boldsymbol{A}) \frac{||\boldsymbol{E}||}{||\boldsymbol{A}||}$$

If input data are accurate to machine precision, then bound for relative error in solution x becomes

$$\frac{||\Delta \boldsymbol{x}||}{||\boldsymbol{x}||} \leq \mathrm{cond}(\boldsymbol{A}) \epsilon_{mach}$$

# Residual




Residual vector is defined by $\boldsymbol{r = b - A\hat{x}}$

In theory, if A is non-singular, then $||\hat{x} - x|| = 0$ if, and only if, $||\boldsymbol{r}|| = 0$.

# Residual

$$\boldsymbol{r = b - A\hat{x}}$$

$$\boldsymbol{(A + E) \cdot \hat{x} = b}$$

$$\boldsymbol{||r|| = ||b - A\hat{x}|| = ||E\hat{x}|| \leq ||E|| \cdot ||\hat{x}||}$$

$$\boldsymbol{\frac{||r||}{||A|| \cdot ||\hat{x}||} \leq \frac{||E||}{||A||}}$$

$$\boldsymbol{\frac{||\Delta x||}{||\hat{x}||} \leq \mathrm{cond}(A) \frac{||E||}{||A||}} \longrightarrow \boldsymbol{\frac{||\Delta x||}{||\hat{x}||} \leq \mathrm{cond}(A) \frac{||r||}{||A|| \cdot ||\hat{x}||}}$$


# Solving linear systems

# Solving linear systems




What type of linear systems is easy to solve?

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

What type of transformation of a linear system leaves the solution unchanged?

$$\boldsymbol{M}\boldsymbol{A}\boldsymbol{z} = \boldsymbol{M}\boldsymbol{b}$$ M is non-singular

$$\boldsymbol{z} = (\boldsymbol{M}\boldsymbol{A})^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{b} = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{x}$$

# Permutation matrix


- Permutation matrix P has one 1 in each row and column and zeros elsewhere
- $P^T$ reverse permutation, so $P^{-1} = P^T$
- Premultiplying both sides of system -> solution unchanged.

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} v_3 \\ v_1 \\ v_2 \end{bmatrix}$$

# Triangular Linear Systems




What type of linear systems is easy to solve?

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

Lower triangular matrix

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$-6x_1 - 4x_2 = -6$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$x_2 = \left(-6 - \left(-6x_1\right)\right) / (-4)$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$x_2 = 3$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$x_2 = 3$$

$$x_1 + 2x_2 + 2x_3 = 3$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$x_2 = 3$$

$$x_3 = (3 - (x_1 + 2x_2))/2$$

# Example: Lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

$$x_1 = \frac{1}{-1} = -1$$

$$x_2 = 3$$

$$x_3 = -1$$

# Example: Lower triangular matrix


$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$x_1 = b_1/l_{11},$$

$$x_2 = (b_2 - (l_{21}x_1))/l_{22}$$

$$x_3 = (b_3 - (l_{31}x_1 + l_{32}x_2))/l_{33}$$

# Lower Triangular Matrix




Lower triangular matrix

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Forward-substitution:

$$x_1 = b_1/l_{11}, \qquad x_i = \left( b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right) /l_{ii}, i = 2, ..., n$$

# Lower Triangular Matrix




Forward-substitution:

```
for j = 1 to n
    if  l_jj = 0 then stop      # stop if matrix is singular
    x_j = b_j / l_jj            # compute solution component
    for i = j + 1 to n
        b_i = b_i - l_ij x_j    # update right-hand side
    end
end
```

~$n^2$ multiplications and additions

# Exercise: Lower Triangular Matrix


Write a fortran program to solve this lower triangular matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ -6 & -4 & 0 \\ 1 & 2 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -6 \\ 3 \end{bmatrix}$$

# Upper Triangular Matrix




Upper triangular matrix

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Back-substitution:

$$x_n = b_n / u_{nn}, \qquad x_i = \left( b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{ii}, i = 2, ..., n$$

# Upper Triangular Matrix




Back-substitution:

```
for j = n to 1
    if  u_jj = 0 then stop        # stop if matrix is singular
    x_j = b_j / u_jj              # compute solution component
    for i = 1 to j-1
        b_i = b_i - u_ij x_j      # update right-hand side
    end
end
```

# Exercise: Upper Triangular Matrix


Write a fortran program to solve this upper triangular matrix

$$\begin{bmatrix} 1 & 2 & 2 \\ 0 & -4 & -6 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ 1 \end{bmatrix}$$

# Elementary Elimination Matrices


Goal: transfer a general linear system into a triangular linear system

This can be accomplished by taking linear combinations of rows

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

# Elementary Elimination Matrices


More generally, we can annihilate all entries below k-th position in n-vector **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 $a_k$ (pivot) must be nonzero

# Elementary Elimination Matrices


- $M_k$ is an unit lower triangular matrix and non-singular
- $M_k = I - m_k e_k^T$, where $m_k = [0,\ldots,0,m_{k+1},\ldots m_n]^T$ and $e_k$ is k-th column of identity matrix
- $M_k^{-1} = I + m_k e_k^T$, which means $M_k^{-1} = L_k$ is same as $M_k$ except signs of multipliers are reversed
- If $M_j$, $j > k$, is another elementary elimination matrix, with vector of multipliers $m_k$, then

$$M_k M_j = I - m_k e_k^T - m_j e_j^T + m_k e_k^T m_j e_j^T$$

$$= I - m_k e_k^T - m_j e_j^T$$

# Example: Elementary Elimination Matrices




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

$$M_1 a = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \qquad M_2 a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix}$$

# Example: Elementary Elimination Matrices




Note that

$$L_1 = M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad L_2 = M_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1/2 & 1 \end{bmatrix}$$

$$M_1 M_2 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 1/2 & 1 \end{bmatrix} \qquad L_1 L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1/2 & 1 \end{bmatrix}$$

# LU decomposition




Or "Gauss Elimination"

- Choose $M_1$ with $a_{11}$ as pivot to annihilate first column of A
- Next choose $M_2$ with $a_{22}$ as pivot to annihilate second column of $M_1$A below second row
- The resulting matrix MA is an upper triangular matrix, which can be solved by back-substitution to obtain the solution

$$M_{n-1}....M_2 M_1 Ax = M_{n-1}.....M_1 b$$

$$MAx = Mb$$

# LU decomposition




$$MAx = Mb$$

$$MA = U$$

$$M^{-1} = L$$

$$\boxed{\boldsymbol{A = LU}}$$

Problem becomes

$$\boldsymbol{LUx = b}$$

$$\boldsymbol{Ly = b}$$ Solved by forward-substitution

$$\boldsymbol{Ux = y}$$ Solved by back-substitution

# Example: LU decomposition




Use Gaussian elimination to solve linear system:

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

# Example: LU decomposition




$$\boldsymbol{M_1 A} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 4 & 9 & -3 \\ -2 & -3 & 7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix}$$

$$\boldsymbol{M_1 b} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 8 \\ 10 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$

$$\boldsymbol{M_2 M_1 A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 4 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix} = U,$$

$$\boldsymbol{M_2 M_1 b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 8 \end{bmatrix} = \boldsymbol{Mb}$$

# Example: LU decomposition




The system is reduced to

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

Back-substitution,

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

# Example: LU decomposition




$$\boldsymbol{L_1 L_2} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} = \boldsymbol{L}$$

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

# LU decomposition




```
for k = 1 to n-1
    if  a_kk = 0 then stop      # stop if pivot is singular
    for i = k +1 to n
        m_ik = a_ik / a_kk      # compute multipliers
    end
    for j = k +1 to n
        for i = k +1 to n
            a_ij = a_ij - m_ik a_kj   # apply transformation to remaining submatrix
        end
    end
end
```

# Exercise: LU decomposition




Write a fortran program to solve it

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

# Pivoting


- Gaussian elimination breaks down if the leading diagonal entry of remaining unreduced matrix is zero at any stage
- Fix: if diagonal entry in column k is zero, then interchange row k with some subsequent row having nonzero entry in column k and then proceed as usual
- If failed, skip to next column
- Zero on diagonal causes the resulting U matrix to be singular but LU factorization can still be completed
- Subsequent back-substitution will fail, however, as it should for singular matrix

# Partial Pivoting


- In principle, any nonzero value will do as pivot, but in practice pivot should be chosen to minimize error propagation.
- To avoid amplifying previous rounding errors, multipliers should not exceed 1 in magnitude
- This can be accomplished by choosing entry of largest magnitude on or below diagonal as pivot at each stage -> Partial pivoting
- Partial pivoting is essential in practice for stability.

# LU decomposition with partial pivoting


for k = 1 to n-1
  Find index p such that $|a_{pk}| >= |a_{ik}|$ for k <= i <= n
  If p != k then
    Interchange rows k and p
  If akk=0 then continue to next k
  for i = k +1 to n
    $m_{ik} = a_{ik} / a_{kk}$  # compute multipliers
  end
  for j = k +1 to n
    for i = k +1 to n
      $a_{ij} = a_{ij} - m_{ik} a_{kj}$  # apply transformation to remaining submatrix
    end
  end
end


# Solving modified problems

# Modified problems


- If the right-hand side of linear system changes but matrix does not, the the LU factorization need not be repeated to solve the new system ($n^3$)
- Only forward- and back-substitution need be repeated for new right-hand side ($n^2$)

# Modified problems


- Sometime refactorization can be avoided even when matrix does change $(A - uv^T)x = b$
- Sherman-Morrison formula gives inverse of matrix resulting from rank-one change to matrix whose inverse is already known

$$(A - uv^T)^{-1} = A^{-1} + A^{-1}u(I - v^T A^{-1}u)^{-1}v^T A^{-1}$$

  u,v are n-vectors
- Evaluation of formula requires $O(n^2)$ work rather than $O(n^3)$ work for inversion

# Special types of linear systems




- Symmetric: $A = A^T$,
- Positive definite: $x^T A x > 0$ for all $x \neq 0$
- Banded: $a_{ij}=0$ for all $|i-j| > c$, where c is the bandwidth.
  -> if c=1 -> tridiagonal matrix
- Sparse: most entries of A are zero

# Symmetric & Positive define matrices


- If A is symmetric and positive definite, then LU factorization be be arraigned so that U =$L^T$, which gives Cholesky factorization

$$A = LL^T$$

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

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


# Software for linear systems

# Software for linear systems


- `LINPACK`: can solve both general dense systems and special systems (such as symmetric or banded)
- `LAPACK` is a replacement for `LINPACK` (especially for high-performance parallel computing)
- `MATLAB` and Python's `numpy` and `spicy` are based on `LAPACK`

# Software for linear systems


- High-level routines in `LINPACK` and `LAPACK` are based on lower-level Basic Linear Algebra Subprograms (`BLAS`)
- `BLAS` encapsulate basic operations on vectors and matrices so they can be optimized for given computer architecture while high-level routines that call them remain portable

# Problem Set 4

https://kuochuanpan.github.io/courses/109ASTR660_CA/

# Next lecture




- Non-Linear equations


# Extra

# Compete Pivoting


- Is a more exhaustive strategy in which largest entry in entire remaining unreduced sub matrix is permuted into diagonal pivot position
- Requires interchanging columns as well as rows.
- Better stability but more expensive

# Small pivots




For $A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}$

If rows are not interchanged, then the multiplier is -1/eps

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

$$\boldsymbol{U} = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$$

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} \neq \boldsymbol{A}$$

# Gauss-Jordan Elimination




- Learned in high school
- Reduced to diagonal rather than triangular form
- Can be used to calculate the inverse of a matrix

# Gauss-Jordan Elimination




$$\begin{bmatrix} 1 & ... & 0 & -m_1 & 0 & ... & 0 \\ ... & & & & & & \\ 0 & ... & 1 & -m_{k-1} & 0 & ... & 0 \\ 0 & ... & 0 & 1 & 0 & ... & 0 \\ 0 & ... & 1 & -m_{k+1} & 0 & ... & 0 \\ ... & & & & & & \\ 0 & ... & 0 & -m_n & 0 & ... & 1 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_{k-1} \\ a_k \\ a_{k+1} \\ \vdots \\ 1 \end{bmatrix}$$

$$m_i = a_i / a_k, i = 1, ..., n$$

~$n^3/2$ multiplications and additions