Solving Linear Systems $Ax=b$
Linear Systems

\[ Ax = b \]

- \( A \) is an \( n \) by \( n \) matrix
- \( x \) is an \( n \) by 1 vector
- \( b \) is a known vector
- \( x \) is unknown vector

known unknown known

\( n \) by \( n \) matrix \( n \) by 1 vector
Linear Systems in The Real World
Circuit Voltage Problem

Given a resistive network and the net current flow at each terminal, find the voltage at each node.

A node with an external connection is a terminal. Otherwise it’s a junction.

Conductance is the reciprocal of resistance. Its unit is siemens (S).
Kirchoff’s Law of Current

At each node, the net current flow = 0.

Consider \( v_1 \). We have

\[
2 + 3(v_2 i - v_1) + (v_3 i - v_1) = 0;
\]

which after regrouping yields

\[
3(v_1 i - v_2) + (v_1 i - v_3) = 2;
\]

I = VC
Summing Up

\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2 \\
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4 \\
\end{pmatrix}
= 
\begin{pmatrix}
2 \\
0 \\
-4 \\
2 \\
\end{pmatrix}
\]

Here \( A \) represents conductances, \( v \) represents voltages at nodes, and \( Av \) is the net current flow at each node.
Solving

\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
2 \\
2 \\
0 \\
3
\end{pmatrix}
= 
\begin{pmatrix}
2 \\
0 \\
-4 \\
2
\end{pmatrix}
\]

Here \(A\) represents conductances, \(\mathbf{v}\) represents voltages at nodes, and \(A\mathbf{v}\) is the net current flow at each node.
Sparse Matrix

An \( n \) by \( n \) matrix is **sparse** when there are \( O(n) \) nonzeros.

A reasonably-sized power grid has way more junctions and each junction has only a couple of neighbors.

\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2 \\
\end{pmatrix}
\]
Sparse Matrix Representation

A simple scheme

An array of columns, where each column $A_j$ is a linked-list of tuples $(i, x)$.
Let $G(x, y)$ and $g(x, y)$ be continuous functions defined in $R$ and $S$ respectively, where $R$ and $S$ are respectively the region and the boundary of the unit square (as in the figure).
Modeling a Physical System

We seek a function \( u(x, y) \) that satisfies Poisson’s equation in \( \mathbb{R} \)

\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x; y)
\]

and the boundary condition \( u(S; y) = g(x; y) \)

Example: \( g(x, y) \) specifies fixed temperature at each point on the boundary, \( G(x, y) = 0, u(x, y) \) is the temperature at each interior point.
Example

Temperature on three sides of a square plate is 0 and on one side is $\pi$.

Discretization

Imagine a uniform grid with a small spacing $h$. 
Finite Difference Method

Replace the partial derivatives by difference quotients

\[ \frac{\partial^2 u}{\partial y^2} = \frac{u(x + h; y) + u(x - h; y) - 2u(x; y)}{h^2} \]

\[ \frac{\partial^2 u}{\partial x^2} = \frac{u(x; y + h) + u(x; y - h) - 2u(x; y)}{h^2} \]

Poisson's equation now becomes

\[ \frac{\partial^2 u}{\partial y^2} = \frac{u(x; y + h) + u(x; y - h) - 2u(x; y)}{h^2} = \frac{\Delta}{h^2} G(x; y) \]

Exercise:

Derive the 5-pt diff. eqt. from first principle (limit).
For each point in \( \mathbb{R} \)

\[
4u(x; y) \quad u(x + h; y) \quad u(x - h; y) \\
\]

\[
i \quad u(x; y + h) \quad u(x; y - h) = \frac{1}{h^2}G(x; y)
\]

The total number of equations is \( \frac{1}{h^2} \).

Now write them in the matrix form, we’ve got one BIG linear system to solve!

\[
4u(x; y) \quad u(x + h; y) \quad u(x - h; y) \\
\]

\[
i \quad u(x; y + h) \quad u(x; y - h) = \frac{1}{h^2}G(x; y)
\]
An Example

Consider $u_{3,1}$, we have

$$4u(3; 1) \mid u(4; 1) \mid u(2; 1)$$

$$\mid u(3; 2) \mid u(3; 0) = \mid h^2 G(x; y)$$

which can be rearranged

$$4u(3; 1) \mid u(2; 1) \mid u(3; 2)$$

$$= \mid h G(3; 1) + u(4; 1) + u(3; 0)$$
An Example

4u(x; y) = u(x + h; y) - u(x; y) - u(x; y + h) + u(x; y - h) = h^2 G(x; y)

Each row and column can have a maximum of 5 nonzeros.

\[
\begin{pmatrix}
4 & -1 & -1 & -1 & -1 \\
-1 & 4 & -1 & -1 & -1 \\
-1 & 4 & -1 & -1 & -1 \\
-1 & 4 & -1 & -1 & -1 \\
-1 & 4 & -1 & -1 & -1
\end{pmatrix}
\begin{pmatrix}
u_{11} \\
u_{12} \\
u_{13} \\
u_{21} \\
u_{22} \\
u_{23} \\
u_{31} \\
u_{32} \\
u_{33}
\end{pmatrix}
= 
\begin{pmatrix}
b_{11} \\
b_{12} \\
b_{13} \\
b_{21} \\
b_{22} \\
b_{23} \\
b_{31} \\
b_{32} \\
b_{33}
\end{pmatrix}
How to Solve?

Ax = b

Find $A^{-1}$

Direct Method: Gaussian Elimination

Iterative Method: Guess $x$ repeatedly until we guess a solution
Inverse Of Sparse Matrices

...are not necessarily sparse!

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
2 & 1 & 3 \\
2 & 2 & 1
\end{pmatrix}
\quad \begin{pmatrix}
4 & -2 & -3 & -1 & 10 & 1 & 3 & 2 & -1 & 5 \\
11 & 7 & 11 & 9 & 12 & 12 & 15 & 17 & 12 & 11 \\
14 & 5 & 14 & 12 & -7 & -9 & 12 & 11 & 15 & 11 \\
11 & 11 & 12 & 11 & 11 & 14 & 11 & 13 & 11 & 11 \\
\end{pmatrix}
\]
Direct Methods
Solution by LU Decomposition

\[
\begin{bmatrix}
A
\end{bmatrix} = \begin{bmatrix}
L & 0 \\
U
\end{bmatrix}
\]

\[A \begin{bmatrix} x \end{bmatrix} = LU \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} \]

Solve \[Ly = b\] for \(y\),
then solve \[Ux = y\] for \(x\).
Transform A to U through Gaussian Elimination

\[
\begin{bmatrix}
1 & 3 & 1 \\
1 & 2 & 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 2 & -1 & 3 \\
2 & 2 & 1
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 2 & -1 & 3 \\
-2 \cdot \text{row 3} & +1 \cdot \text{row 3} & -3 \cdot \text{row 3} \\
-6 & 7 & -8
\end{bmatrix}
\]
L specified by multiples of pivots

\[ A \]

\[ \begin{bmatrix} 1 & 1 & 1 \\ L_{1} & \cdot & \cdot \end{bmatrix} \times \begin{bmatrix} \text{\textbullet} \\ \text{\textbullet} \\ \text{\textbullet} \end{bmatrix} \]

\[ \downarrow \]

\[ \begin{bmatrix} 111 \\ L_{1} \end{bmatrix} \times \begin{bmatrix} \text{\textbullet} \\ \text{\textbullet} \end{bmatrix} \]

L_{ij} is multiple of j’th pivot used to eliminate i’th entry in column j.
Pivoting Creates Fill

\[
\begin{bmatrix}
1 & 3 \\
1 & 2 \\
2 & 2 \\
-1 & 4 \\
3 & 4 & 1
\end{bmatrix}
\]
\[ \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix} \]

pivot

\[ \begin{bmatrix} 1 & 2 & -1 & 3 \\ -1 & 2 & -1 & 47 \\ -3 & -6 & 47 & 1-8 \end{bmatrix} \]

"fill" (in green)
Reordering rows and columns doesn’t change solution
... but might reduce fill
Graph Model of Matrix

Symmetric matrix $\leftrightarrow$ undirected graph

Matrix:
\[
\begin{bmatrix}
    a & b & c & d & e \\
    x & x & xx & xx & xx \\
    x & x & x & x & x \\
    x & x & x & x & x \\
    x & x & x & x & x \\
\end{bmatrix}
\]
Graph View of Partitioning
Symmetric, positive definite matrices

undirected graph

pirots are nonzero
independent of order
Min-Degree Heuristic

Alternatively, eliminate vertex with smallest fill.

- simple, greedy algorithm
- works reasonably well in practice (doesn't create too much fill)

(Mankowitz, 57)
(Tinney & Walker, 67)
Idea: Eliminating separator last prevents fill between components
Nested Dissection

1. Find small node separator
2. Place separator vertices last in the order
3. Recurse on connected components

( George, 73; Lipton, Rose, Tanjan, 79)
Nested Dissection Example
Analysis of Nested Dissection

Lipton, Rose, Tarjan, 79.

$\sqrt{n}$ separation theorem

$O(m \cdot \log n)$ fill

$O(n^{3/2})$ work

Gilbert, Tarjan, 87

$O(m \cdot \log n)$ fill, $O(n^{3/2})$ work

on planar graphs, graphs with bounded genus, and bounded degree & $O(\sqrt{n})$ separators
Results Concerning Fill

Finding an elimination order with minimal fill is hopeless
Garey and Johnson-GT46, Yannakakis SIAM JADM 1981

\( O(\log n) \) Approximation
Sudipto Guha, FOCS 2000
Nested Graph Dissection and Approximation Algorithms

\( \Omega(n \log n) \) lower bound on fill
(Maverick still has not dug up the paper...)
Iterative Methods
Algorithms Apply to Laplacians

Given a positively weighted, undirected graph $G = (V, E)$, we can represent it as a Laplacian matrix.

$$
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
$$
Laplacians

Laplacians have many interesting properties, such as

- Diagonals, $\sum_0$ denotes total incident weights
- Off-diagonals $< 0$ denotes individual edge weights

- Row sum $= 0$, column sum $= 3$

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>-3</th>
<th>-1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>-2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Symmetric

![Graph with vertices v1, v2, v3, v4 and edges connecting them with weights 1, 2, and 1]
Earlier I showed you a system that is not quite Laplacian.

Boundary points don’t have 4 neighbors

$$
\begin{pmatrix}
4 & -1 & -1 \\
-1 & 4 & -1 \\
-1 & -1 & 4 \\
-1 & 4 & -1 \\
-1 & -1 & 4 \\
-1 & -1 & -1
\end{pmatrix}
\begin{pmatrix}
u_{11} \\
u_{12} \\
u_{13} \\
u_{21} \\
u_{22} \\
u_{23} \\
u_{31} \\
u_{32} \\
u_{33}
\end{pmatrix}
=
\begin{pmatrix}
b_{11} \\
b_{12} \\
b_{13} \\
b_{21} \\
b_{22} \\
b_{23} \\
b_{31} \\
b_{32} \\
b_{33}
\end{pmatrix}
$$
Making It Laplacian

We add a dummy variable and force it to zero.

\[
\begin{pmatrix}
4 & -1 & -1 & -1 & -2 \\
-1 & 4 & -1 & -1 & -1 \\
-1 & -1 & 4 & -1 & -2 \\
-1 & -1 & 4 & -1 & -1 \\
-1 & -1 & 4 & -1 & -2 \\
-2 & -1 & -2 & -1 & -1 & -2 & 12
\end{pmatrix}
\]
The Basic Idea

Start off with a guess $x^{(0)}$.

Use $x^{(i)}$ to compute $x^{(i+1)}$ until convergence.

We hope

- the process converges in a small number of iterations
- each iteration is efficient
The RF Method [Richardson, 1910]

\[ x^{(i+1)} = x^{(i)} + (Ax^{(i)} - b) \]

Domain \( \xrightarrow{A} \) Range

Units are all wrong! \( x^{(i)} \) is voltage, residual is current!
Why should it converge at all?

\[ x^{(i+1)} = x^{(i)} + (Ax^{(i)} - b) \]
It only converges when...

\[ x^{(i+1)} = x^{(i)} + (Ax^{(i)} + b) \]

**Theorem**

A first-order stationary iterative method

\[ x^{(i+1)} = Gx^{(i)} + k \]

converges iff

\[ \frac{1}{\lambda}(G) < 1. \]

\( \rho(A) \) is the maximum absolute eigenvalue of G.
Fate?

Once we are given the system, we do not have any control on A and b.

How do we guarantee convergence?
Preconditioning

\[ B^{-1}A x = B^{-1}b \]

Instead of dealing with \( A \) and \( b \), we now deal with \( B^{-1}A \) and \( B^{-1}b \).

The word “preconditioning” originated with Turing in 1948, but his idea was slightly different.
Preconditioned RF

\[ x^{(i+1)} = x^{(i)} \pm (B^{-1}A x^{(i)} \pm B^{-1}b) \]

Since we may precompute \( B^{-1}b \) by solving \( By = b \),
each iteration is dominated by computing \( B^{-1}A x^{(i)} \), which is

- a multiplication step \( Ax^{(i)} \) and
- a direct-solve step \( Bz = Ax^{(i)} \).

Hence a preconditioned iterative method is in fact

\[ \text{a hybrid} \]
The Art of Preconditioning

We have a lot of flexibility in choosing $B$.

- Solving $Bz = Ax^{(i)}$ must be fast
- $B$ should approximate $A$ well for a low iteration count

Trivial

What’s the point?
Classics

\[ x^{(i+1)} = x^{(i)} + (B^{-1}Ax^{(i)} - B^{-1}b) \]

Jacobi

Let \( D \) be the diagonal sub-matrix of \( A \).
Pick \( B = D \).

Gauss-Seidel

Let \( L \) be the lower triangular part of \( A \) w/ zero diagonals
Pick \( B = L + D \).

Conjugate gradient

Move in an orthogonal direction in each