This talk is not about...
**Credits**


Combinatorial Preconditioners for Large, Sparse, Symmetric, Diagonally-Dominant Linear Systems

Keith Gremban (CMU PhD 1996)

**Linear Systems**

\[ Ax = b \]

A useful way to do matrix algebra in your head:
- Matrix-vector multiplication + Linear combination of matrix columns

**Matrix-Vector Multiplication**

\[
\begin{pmatrix}
11 & 12 & 13 \\
21 & 22 & 23 \\
31 & 32 & 33
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c
\end{pmatrix}
= 
\begin{pmatrix}
11a + 12b + 13c \\
21a + 22b + 23c \\
31a + 32b + 33c
\end{pmatrix}
\]

- Using \( B^T A^T = (AB)^T \). (why do we need this, Maverick?)
- \( x^T A \) should also be interpreted as a linear combination of the rows of \( A \).

**How to Solve?**

\[ Ax = b \]

- Find \( A^{-1} \)
- Guess \( x \) repeatedly until we guess a solution
- Gaussian Elimination

Strassen had a faster method to find \( A^{-1} \).
Large 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.

Kirchoff’s Law of Current
At each node, net current flow = 0.
Consider \( v_1 \). We have
\[
2 + 3(v_2 - v_1) + (v_3 - v_1) = 0,
\]
which after regrouping yields
\[
3(v_1 - v_2) + (v_1 - v_3) = 2.
\]

Summing Up
\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-9 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4
\end{pmatrix}
= \begin{pmatrix} 2 \\ -4 \\ 2 \end{pmatrix}
\]
Given a weighted, undirected graph $G = (V, E)$, we can represent it as a Laplacian matrix.

Laplacian matrices have many interesting properties, such as:
- Diagonals $\geq 0$ denote total incident weights
- Off-diagonals $< 0$ denote individual edge weights
- Row sum = 0
- Symmetric

Solving

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

Did I say “LARGE”?

$$
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
\mathbf{v}_1 \\
\mathbf{v}_2 \\
\mathbf{v}_3 \\
\mathbf{v}_4
\end{pmatrix}
= 
\begin{pmatrix}
2 \\
0 \\
-4 \\
2
\end{pmatrix}
$$

Imagine this being the power grid of America.
**Net Current Flow**

Lemma
Suppose an $n$ by $n$ matrix $A$ is the Laplacian of a resistive network $G$ with $n$ nodes.
If $y$ is the $n$-vector specifying the voltage at each node of $G$, then $Ay$ is the $n$-vector representing the net current flow at each node.

**Power Dissipation**

Lemma
Suppose an $n$ by $n$ matrix $A$ is the Laplacian of a resistive network $G$ with $n$ nodes.
If $y$ is the $n$-vector specifying the voltage at each node of $G$, then $y' Ay$ is the total power dissipated by $G$.

**Sparsity**

Laplacians arising in practice are usually sparse.
- The $i$-th row has $(d+1)$ nonzeros if $v_i$ has $d$ neighbors.

**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.
A Model Problem

We seek a function $u(x, y)$ that satisfies Poisson’s equation in $R$ and the boundary condition in $S$.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = G(x, y)$$

$$u(x, y) = g(x, y)$$

Discretization

Imagine a uniform grid with a small spacing $h$. If $g(x,y) = 0$, this is called a Dirichlet boundary condition.
**Five-Point Difference**

Replace the partial derivatives by difference quotients

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

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

The Poisson’s equation now becomes

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

\[ - u(x, y + h) - u(x, y - h) = -h^2 G(x, y) \]

**Exercise:**
Derive the 5-pt diff. eq. from first principles.

**For each point in** \( R \)

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

\[ - u(x, y + h) - u(x, y - h) = -h^2 G(x, y) \]

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

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

\[ \begin{pmatrix}
4 & -1 & 4 & -1 & -1 \\
-1 & 4 & -1 & 4 & -1 \\
4 & -1 & 4 & -1 & -1 \\
-1 & 4 & -1 & 4 & -1 \\
4 & -1 & 4 & -1 & -1 \\
\end{pmatrix}
\begin{pmatrix}
4u(x, y) \\
-1 \\
4u(x, y) \\
-1 \\
4u(x, y) \\
\end{pmatrix}
\]

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

Really, it’s rare to see large dense matrices arising from applications.

Laplacian???

I showed you a system that is not quite Laplacian.

We’ve got way too many boundary points in a 3x3 example.

Making It Laplacian

We add a dummy variable and force it to zero.

(How to force? Well, look at the rank of this matrix first...)

Sparse Matrix Representation

A simple scheme

An array of columns, where each column $A_j$ is a linked-list of tuples $(i, x)$. 
Solving Sparse Systems

Gaussian Elimination again?
Let’s look at one elimination step.

\[
\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}
\]

Gaussian Elimination introduces fill.

\[
\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}
\]

Gaussian Elimination introduces fill.

\[
\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1
\end{pmatrix}
\]

Fill

Of course it depends on the elimination order.
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…)
When Fill Matters...

\[ Ax = b \]

- Find \( A^{-1} \)
- Guess \( x \) repeatedly until we guessed a solution
- Gaussian Elimination

Inverse Of Sparse Matrices

...are not necessarily sparse either!

\[
B = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]

\[
B^3
\]

And the winner is...

\[ Ax = b \]

- Find \( A^{-1} \)
- Guess \( x \) repeatedly until we guessed a solution
- Gaussian Elimination

Iterative Methods

Checkpoint
- How large linear system actually arise in practice
- Why Gaussian Elimination may not be the way to go
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

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

\[ \rho(G) < 1. \]

\( \rho(A) \) is the maximum absolute eigenvalue of \( A \).
Once we are given $A$ and $b$, we have no control on $A$ and $b$. How do we guarantee even convergence?

**Preconditioning**

$B^{-1}Ax = B^{-1}b$

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**

$x^{(i+1)} = x^{(i)} - (B^{-1}Ax^{(i)} - B^{-1}b)$

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

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

Hence a preconditioned iterative method is in fact 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

$I$ $B$ $A$

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 $B = L + D$.

**“Combinatorial”**

We choose to measure how well $B$ approximates $A$ by comparing combinatorial properties of (the graphs represented by) $A$ and $B$.

- Hence the term “Combinatorial Preconditioner”.

**Questions?**

**Graphs as Matrices**
**Edge Vertex Incidence Matrix**

Given an undirected graph \( G = (V, E) \),
let \( \Gamma \) be a \(|E| \times |V| \) matrix of \(-1, 0, 1\).

For each edge \((u, v)\), set \( \Gamma_{e,u} \) to -1 and \( \Gamma_{e,v} \) to 1.
Other entries are all zeros.

**Weighted Graphs**

Let \( W \) be an \(|E| \times |E| \) diagonal matrix where \( W_{e,e} \) is the weight of the edge \( e \).

\[
\begin{pmatrix}
2 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 \\
0 & 0 & 3 & 0 \\
0 & 0 & 0 & 4 \\
\end{pmatrix}
\]

**Laplacian**

The Laplacian of \( G \) is defined to be \( \Gamma W \).

\[
\begin{pmatrix}
2 & 12 & -2 & 0 & 0 \\
-2 & 11 & -4 & -3 & -8 \\
-3 & -4 & 10 & -3 & -1 \\
-3 & 19 & -3 & 6 & -1 \\
0 & -8 & -5 & -2 & -2 \\
-2 & -1 & 16 & -5 & -4 \\
-4 & -2 & -4 & 16 & -2 \\
\end{pmatrix}
\]
Properties of Laplacians

Let \( L = \Gamma^T W \Gamma \).

*Prove by example*

\[
\begin{pmatrix}
2 & -2 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 12 & -6 & 0 & -3 & 0 & 0 & 0 & 0 \\
-3 & -6 & 19 & -2 & 0 & -1 & 0 & 0 & 0 \\
-3 & -2 & 6 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & -1 & 0 & -1 & 0 \\
0 & -1 & 15 & -6 & -2 & 0 & 0 & 0 & -1 \\
0 & -3 & 9 & -4 & 0 & 0 & 0 & 0 & 0 \\
0 & -2 & 4 & 16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

Properties of Laplacians

A matrix \( A \) is Positive SemiDefinite if
\[
\forall x, x^T Ax \geq 0.
\]

Since \( L = \Gamma^T W \Gamma \), it's easy to see that for all \( x \)
\[
x^T (\Gamma^T W \Gamma) x = (W^{1/2} \Gamma x)^T (W^{1/2} \Gamma x) \geq 0.
\]

Laplacian

The Laplacian of \( G \) is defined to be \( \Gamma^T W \Gamma \).

\[
\begin{pmatrix}
1 & 2 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 3 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 4 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 3 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 2 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 3 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 3 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 3
\end{pmatrix}
\]
Graph Embedding

A Primer

Graph Embedding

**Vertex in G → Vertex in H**

**Edge in G → Path in H**

Guest

Host

Dilation

For each edge \( e \) in \( G \), define \( \text{dil}(e) \) to be the number of edges in its corresponding path in \( H \).

Congestion

For each edge \( e \) in \( H \), define \( \text{cong}(e) \) to be the number of embedding paths that uses \( e \).
Support Theory

**Disclaimer**
The presentation to follow is only "essentially correct".

---

**Support**

**Definition**
The support required by a matrix $B$ for a matrix $A$, both $n$ by $n$ in size, is defined as

$$\sigma(A/B) := \min\{\tau \in \mathbb{R} | \forall x, x^T(\tau B - A)x \geq 0\}$$

Think of $B$ supporting $A$ at the bottom.

---

**Support With Laplacians**

Life is Good when the matrices are Laplacians.

- Remember the resistive circuit analogy?
**Power Dissipation**

**Lemma**
Suppose an $n$ by $n$ matrix $A$ is the Laplacian of a resistive network $G$ with $n$ nodes.
If $y$ is the $n$-vector specifying the voltage at each node of $G$, then $y^T A y$ is the total power dissipated by $G$.

**Circuit-Speak**

\[ \sigma(A/B) := \min \{ \tau \in \mathbb{R} | \forall x, x^T (\tau B - A) x \geq 0 \} \]

Read this loud in circuit-speak:
*The support for $A$ by $B$ is the minimum number $\tau$ so that for all possible voltage settings, $\tau$ copies of $B$ burn at least as much energy as one copy of $A$.*

**Congestion-Dilation Lemma**

Given an embedding from $G$ to $H$,

\[ \sigma(G/H) \leq (\max_{h \in B(H)} \text{cong}(h)) (\max_{g \in B(G)} \text{dil}(g)) \]

**Transitivity**

\[ \sigma(A/C) \leq \sigma(A/B) \cdot \sigma(B/C) \]

**Pop Quiz**
For Laplacians, prove this in circuit-speak.
**Generalized Condition Number**

**Definition**

The generalized condition number of a pair of PSD matrices is

\[ \kappa(A, B) := \sigma(A/B) \cdot \sigma(B/A) \]

\( A \) is Positive Semi-definite iff \( \forall x, x^T A x \geq 0. \)

**Preconditioned Conjugate Gradient**

Solving the system \( Ax = b \) using PCG with preconditioner \( B \) requires at most

\[ \sqrt{\sigma(A/B) \cdot \sigma(B/A)} \log \frac{1}{\varepsilon} \]

iterations to find a solution such that

\[ ||Ax - b|| \leq \varepsilon ||b||. \]

Convergence rate is dependent on the actual iterative method used.

**Support Trees**

**Information Flow**

In many iterative methods, the only operation using \( A \) directly is to compute \( Ax^{(i)} \) in each iteration.

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

Imagine each node is an agent maintaining its value.

The update formula specifies how each agent should update its value for round \( (i+1) \) given all the values in round \( i \).
The Problem With Multiplication

Only neighbors can “communicate” in a multiplication, which happens once per iteration.

\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
v_4
\end{pmatrix}
= 
\begin{pmatrix}
v'_1 \\
v'_2 \\
v'_3 \\
v'_4
\end{pmatrix}
\]

Diameter As A Natural Lower Bound

In general, for a node to settle on its final value, it needs to “know” at least the initial values of the other nodes.

Preconditioning As Shortcutting

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

By picking B carefully, we can introduce shortcuts for faster communication.

But is it easy to find shortcuts in a sparse graph to reduce its diameter?

Square Mesh

\[
\sqrt{n}
\]

Let’s pick the complete graph induced on all the mesh points.

Mesh: \(O(n) \) edges

Complete graph: \(O(n^2) \) edges
So exactly how do we propose to solve a dense \( n \) by \( n \) system faster than a sparse one? 

\( B \) can have at most \( O(n) \) edges, i.e., sparse...

**Support Tree**

Build a Steiner tree to introduce shortcuts!

If we pick a balanced tree, no nodes will be farther than \( O(\log n) \) hops away.

Need to specify weights on the tree edges.

**Mixing Speed**

The speed of communication is proportional to the corresponding coefficients on the paths between nodes.

\[
\begin{pmatrix}
4 & -3 & -1 & 0 \\
-3 & 6 & -1 & -2 \\
-1 & -1 & 2 & 0 \\
0 & -2 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4
\end{pmatrix}
\]

**Setting Weights**

The nodes should be able to talk at least as fast as they could without shortcuts.

How about setting all the weights to \( \infty \)?
Recall PCG’s Convergence Rate
Solving the system $Ax = b$ using PCG with preconditioner $B$ requires at most
$$\sqrt{\sigma(A/B) \cdot \sigma(B/A)} \log \frac{1}{\varepsilon}$$
iterations to find a solution such that
$$||Ax - b|| \leq \varepsilon ||b||.$$

Size Matters
How big is the preconditioner matrix $B$?

The Search Is Hard
Finding the "right" preconditioner is really a tradeoff
- Solving $Bz = Ax^{(i)}$ must be fast
- $B$ should approximate $A$ well for a low iteration count

How to Deal With Steiner Nodes?
The Trouble of Steiner Nodes

Computation

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

Definitions, e.g.,

\[ \sigma(B/A) := \min\{ \tau \in \mathbb{R} | \forall x, x^T(\tau A - B)x \geq 0 \} \]

Generalized Support

Let \( B = \begin{pmatrix} T & U \\ U^T & W \end{pmatrix} \) where \( W \) is \( n \) by \( n \).

Then \( \sigma(B/A) \) is defined to be

\[ \min\{ \tau \in \mathbb{R} | \forall x, x^T A x \geq \begin{pmatrix} y \\ x \end{pmatrix}^T B \begin{pmatrix} y \\ x \end{pmatrix} \} \]

where \( y = -T^{-1}Ux \).

Circuit-Speak

\[ \sigma(B/A) := \min\{ \tau \in \mathbb{R} | \forall x, x^T A x \geq \begin{pmatrix} y \\ x \end{pmatrix}^T B \begin{pmatrix} y \\ x \end{pmatrix} \} \]

Read this loud in circuit-speak:

"The support for \( B \) by \( A \) is the minimum number \( \tau \) so that for all possible voltage settings at the terminals, \( \tau \) copies of \( A \) burn at least as much energy as one copy of \( B \)."

Thomson’s Principle

\[ \sigma(B/A) := \min\{ \tau \in \mathbb{R} | \forall x, x^T A x \geq \begin{pmatrix} y \\ x \end{pmatrix}^T B \begin{pmatrix} y \\ x \end{pmatrix} \} \]

Fix the voltages at the terminals. The voltages at the junctions will be set such that the total power dissipation in the circuit is minimized.
Racke's Decomposition Tree

Laminar Decomposition

A laminar decomposition naturally defines a tree.

Racke, FOCS 2002

Given a graph $G$ of $n$ nodes, there exists a laminar decomposition tree $T$ with all the "right" properties as a preconditioner for $G$.

Except his advisor didn’t tell him about this...