Solving Linear Systems

- How do you solve matrix equation $Ax=b$?
- Standard answer: multiply both sides by $A^{-1}$
- What’s wrong with this answer in real life?
  - Often takes too long
  - Real problems tend to be sparse, but inverses aren’t
  - Precision issues
- A better answer, sometimes:
  - Gaussian elimination
  - Equivalently, compute LU factorization and back-substitute
    - A good idea if your problem is dense and unstructured
    - Can solve for multiple $b$ without much more work
    - Problems are rarely dense and unstructured
- What about sparse systems?
  - Iterative methods!!!

First Iterative Method

- Solve the following linear system:
  - You have 10 seconds...

\[
\begin{pmatrix}
100 & 3 & -2 \\
1 & 200 & 5 \\
-4 & 3 & 100
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix} = 
\begin{pmatrix}
800 \\
1000 \\
500
\end{pmatrix}
\]

- Okay, what about approximately?
  - Answer pretty close to $x_1 = (8, 5, 5)^T$
  - Gives $Ax_1 = (805, 1033, 483)^T$
  - Error is $e_1 = (-5, -33, 17)^T$
  - Can improve approximation by doing same thing with $e_1$ as RHS and adding result to $x_1$
    - And again, and again, and again
  - Get $x_2 = (7.95, 4.835, 5.17)^T$
  - Actual answer is $x = (7.9585, 4.8309, 5.1734)^T$
  - Error falls off exponentially, so convergence is very fast

A Little Harder

- Solve this one

\[
\begin{pmatrix}
100 & -1 & -4 \\
100 & 100 & 3 \\
100 & 100 & 100
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix} = 
\begin{pmatrix}
100 \\
200 \\
300
\end{pmatrix}
\]

Approximate answer is $(1,1,1)^T$

Can do similar thing to last time
What Just Happened?

- Write $A = L + S$
  - $L$ should be large (“most of $A$”) and easy to invert
  - $S$ should be “small”
- $L^{-1}$ ought to be a decent approximation of $A^{-1}$
- $x_1 = L^{-1} b$
- $r_1 = b - Ax_1$
- $x_{k+1} = x_k + L^{-1} r_k$
- $r_{k+1} = b - Ax_{k+1}$
- Repeat until happy/bored
- Note: no inversion, just matrix*vector multiplications
- When, and how quickly, does this converge to the right answer?
  - See blackboard

Names for What We Just Did

- Jacobi iteration
  - Break matrix into $D+S$
  - $D =$ diagonal matrix
  - $S =$ small other stuff
- Gauss-Seidel iteration
  - Break matrix into $L + S$
  - $L =$lower triangular
  - $S =$ small other stuff

Spectral Radius

- Before analyzing, helps to introduce a definition:
- **Definition:** The spectral radius $\rho$ of a symmetric matrix $M$ is $|\lambda_{\text{max}}|$
- Note that norm of $M^n x$ is bounded by $\rho^n |x|$
  - So $\rho$ measures how fast powers of $M$ converge to zero
  - We’ll use this often to analyze iterative methods
  - We’ve done this before

More Iterative Methods

- For rest of today, all matrices will be:
  - $n \times n$ square matrices
  - Symmetric
  - Positive definite
  - In particular, non-degenerate
- These seem to be very restrictive constraints, but they’re not that bad
- Suppose that I give you a black box that can solve symmetric, positive definite linear systems.
- I claim you can use this to solve any nondegenerate square linear system
- How?
  - Solve $A^T A x = A^T x$
  - What about sparsity???
Turning a Linear Problem into a Quadratic One

- Let 
  \[ f(x) = \frac{1}{2} x^T A x - b x + c \]

- At what vector \( x \) is \( f \) minimized?
  - See blackboard
  - So minimizing \( f \) gives us the solution if \( A \) is symmetric and positive definite
  - What happens if \( A \) is unsymmetric?
  - What happens if \( A \) is symmetric and negative definite?
  - What happens if \( A \) is symmetric and neither positive nor negative definite?
  - Let’s look at some pictures
  - Pictures taken from Jonathan Shewchuk’s “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain”
    - I highly recommend reading it

Graph of quadratic form \( f(x) = \frac{1}{2} x^T A x - b^T x + c \). The minimum point of this surface is the solution to \( A x = b \).
The eigenvectors of $A$ are directed along the axes of the paraboloid defined by the quadratic form $f(x)$. Each eigenvector is labeled with its associated eigenvalue.

Steepest Descent

- How should we minimize the quadratic form $f$?
- A very natural approach is \textit{steepest descent}
- Find the direction of steepest decrease and take a step in that direction
- Two obvious questions:
  - What direction is it? \textit{Gradient}
  - How big a step should we take? \textit{Line Search}
- Before answering these, a little notation
  - $x =$ actual solution
  - $x_i =$ guess at $i$th iteration
  - $e_i = x_i - x =$ error term
  - $r_i = b - Axe_i$
- Rough answers
- See blackboard
The method of Steepest Descent.

**Steepest Descent Redux**

\[ r_i = b - Ax_i \]
\[ x_{i+1} = x_i + \alpha_i r_i \]
\[ \alpha_i = \frac{r_i^T r_i}{r_i^T Ar_i} \]
\[ e_{i+1} = e_i + \frac{r_i^T r_i}{r_i^T Ar_i} r_i \]

- **Implementation note:**
  - Running time dominated by 2 matrix-vector multiplications
- **Can cut this to 1 (after first iteration)**
  \[ r_{i+1} = r_i - \alpha_i Ar_i \]
- **Can cause roundoff errors to accumulate**
- **So should use top equation every once in a while too**

---

Direction of steepest descent
Line search
Note orthogonality
Line search

**f(x(i) + \alpha r(i))**

Solid arrows: Gradients.
Dotted arrows: Slope along search line.
Analyzing Steepest Descent

- See blackboard
- Nicer to bound energy norm of error vects
  \[ ||e_i||_A = e_i^T A e_i \]
- Can show:
  \[ ||e_{(i+1)}||_A^2 = ||e_{(i)}||_A^2 \left( 1 - \frac{\left( \sum_j \xi_j^2 \lambda_j^2 \right)^2}{\left( \sum_j \xi_j^2 \lambda_j^3 \right) \left( \sum_j \xi_j^3 \lambda_j \right)} \right) \]
- For 2-d case, let \( \mu = \xi_1 / \xi_2 \) and let \( \kappa = \lambda_1 / \lambda_2 \)
  (\( \xi_1 \) is from bigger eigenvalue)
- Thing in parentheses becomes:
  \[ 1 - \frac{(\kappa^2 + \mu^2)^2}{(\kappa + \mu^2)(\kappa^3 + \mu^2)} \]

Steepest Descent converges to the exact solution on the first iteration if the error term is an eigenvector.

Steepest Descent converges to the exact solution on the first iteration if the eigenvalues are all equal.
Why You Already Did this in Calculus Class

- See blackboard