CS 450: Numerical Anlaysis

Lecture 28
Chapter 11 Partial Differential Equations
Solving Sparse Linear Systems

Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

April 29, 2018

Sparse Linear Systems

- Finite-difference and finite-element methods for time-independent PDEs give rise to sparse linear systems:
  - *typified by the 2D Laplace equation, where for both finite differences and FEM,*

$$\underbrace{(\boldsymbol{I} \otimes \boldsymbol{T} + \boldsymbol{T} \otimes \boldsymbol{I})}_{\boldsymbol{A}} \boldsymbol{x} = \boldsymbol{b} \quad \textit{where} \quad \boldsymbol{T} = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & & \end{bmatrix}$$

  - *often have $O(1)$ nonzeros per row/column of the matrix,*
  - *for simple/regular problems matrices are near-Toeplitz (same entry along each (sub/super)diagonal), permitting fast solvers via e.g. FFT.*
- *Direct methods* apply LU or other factorization to $\boldsymbol{A}$, while *iterative methods* refine $\boldsymbol{x}$ by minimizing $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$, e.g. via Krylov subspace methods.
  - *Direct methods provide a high-accuracy solution, but may not be effective at leveraging sparsity to reduce cost.*
  - *Iterative methods effectively leverage sparsity by computing matrix-vector products with $\boldsymbol{A}$, but may require many iterations to achieve high-accuracy.*

Direct Methods for Sparse Linear Systems

- It helps to think of $\boldsymbol{A}$ as the adjacency matrix of graph $G = (V, E)$ where $V = \{1, \ldots n\}$ and $a_{ij} \neq 0$ if and only if $(i, j) \in E$:

  The graph is invariant under permutation of vertices, i.e. for any permutation matrix $\boldsymbol{P}$, reordering V accordingly transeforms the adjacency matrix of the graph into $\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P}$. Note that such reorderings of variables essentially do not change the linear system,

$$\boldsymbol{P}^T \boldsymbol{A} \boldsymbol{P} \underbrace{\boldsymbol{P}^T \boldsymbol{x}}_{\hat{\boldsymbol{x}}} = \boldsymbol{b}$$

- Factorizing the lth row/column in Gaussian elimination corresponds to removing node i, with nonzeros (new edges) introduces for each k, l such that (i, k) and (i, l) are in the graph.
  - *creates clique (fully connected subgraph) among neighbors of vertex i,*
  - *different orderings of vertices can result in radically different amounts of fill,*
  - *finding optimal ordering to reduce fill is NP complete.*

Vertex Orderings for Sparse Direct Methods

- Select the node of minimum degree at each step of factorization:

  Each step minimizes work, but not necessarily amount of fill at that step (depends on whether neighbors are already connected).

- Graph partitioning also serves to bound fill, remove vertex separator $S \subset V$ so that $V \setminus S = V_1 \cup \cdots \cup V_k$ become disconnected, then order $V_1, \ldots, V_k, S$:

  Matrix takes on the form

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & & & \boldsymbol{A}_{1S} \\ & \ddots & & \vdots \\ & & \boldsymbol{A}_{kk} & \boldsymbol{A}_{kS} \\ \boldsymbol{A}_{S1} & \cdots & \boldsymbol{A}_{Sk} & \boldsymbol{A}_{SS} \end{bmatrix}$$

  where each $\boldsymbol{A}_{ii}$ for $i \in \{1, \ldots, k\}$ can be factored independently.

- *Nested dissection* ordering partitions graph into halves recursively, ordering each separator last.

Sparse Iterative Methods

- Direct sparse factorization is ineffective in memory usage and/or cost for many typical sparsity matrices, motivating iterative methods:

  Many such basic iterative methods corresponds to linear fixed point iterations defined by linear system:

$$\boldsymbol{M}\boldsymbol{x}_{k+1} = \boldsymbol{N}\boldsymbol{x}_k + \boldsymbol{b}$$

  which can be written in the form $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x} + \boldsymbol{M}^{-1}\boldsymbol{b}$*. We desire to have a fixed point whenever* $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$*, which implies*

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{N}\boldsymbol{x} + \boldsymbol{A}\boldsymbol{x},$$

  so generally $\boldsymbol{M}$ *and* $\boldsymbol{N}$ *are chosen so that* $\boldsymbol{M} - \boldsymbol{N} = \boldsymbol{A}$*. Moreover, to achieve convergence we need* $\rho(\boldsymbol{g}) = \rho(\boldsymbol{M}^{-1}\boldsymbol{N}) < 1$*.*

Sparse Iterative Methods

- The *Jacobi method* is the simplest iterative solver:

  We split up $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{L} + \boldsymbol{U}$ where $\boldsymbol{D}$ is diagonal while $\boldsymbol{L}$ and $\boldsymbol{U}^T$ are strictly lower triangular. Jacobi iteration uses a fixed point scheme with $\boldsymbol{M} = \boldsymbol{D}$ and $\boldsymbol{N} = -(\boldsymbol{L} + \boldsymbol{U})$, yielding a diagonal system of equations,

  $$\boldsymbol{D}\boldsymbol{x}^{(k+1)} = (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}^{(k)} + \boldsymbol{b}.$$

  The cost of each iteration of Jacobi is proportional to SpMV with $\boldsymbol{A}$.

- The Jacobi method converges if $\boldsymbol{A}$ is strictly row-diagonally-dominant:

  A strictly row-diagonally-dominant $\boldsymbol{A}$ satisfies $|a_{ii}| < \sum_{j \neq i} |a_{ij}|$, which implies that for $\boldsymbol{B} = \boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})$,

  $$\left| \sum_j b_{ij} \right| = \left| \sum_{j \neq i} a_{ij}/a_{ii} \right| < 1,$$

  so that $\rho(\boldsymbol{M}^{-1}\boldsymbol{N}) = \rho(\boldsymbol{B}) < 1$. For the Laplace problem, this condition holds, and the method corresponds to averaging neighbors. However, the coefficient in linear convergence is $\cos(\pi h)$ (goes to 1 with decreasing h).

Gauss-Seidel Method

- The Jacobi method takes weighted sums of $\boldsymbol{x}^{(k)}$ to produce each entry of $\boldsymbol{x}^{(k+1)}$, while Gauss-Seidel uses the latest available values, i.e. to compute $x_i^{(k+1)}$ it uses a weighted sum of

$$x_1^{(k+1)}, \ldots x_{i-1}^{(k+1)}, x_i^{(k)}, \ldots, x_n^{(k)}.$$

We can define the method by the splitting $\boldsymbol{M} = \boldsymbol{D} + \boldsymbol{L}$ so that we have

$$(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{x}^{(k+1)} = \boldsymbol{U}\boldsymbol{x}^{(k)}.$$

The Gauss-Seidel method performs an in-order traversal of the directed acyclic adjacency graph induced by the vertex ordering, and updates each vertex by taking newly computed values from incoming edges and values from the previous iteration from outgoing edges.

- Gauss-Seidel provides somewhat better convergence than Jacobi:
  Convergence and efficiency depend on vertex ordering and connectivity:
  - *for 2-D Poisson, spectral radius is $\cos^2(\pi h)$,*
  - *computational cost is same as Jacobi, but less parallelism available.*

Successive Over-Relaxation

- The *successive over-relaxation* (SOR) method seeks to improve the spectral radius achieved by Gauss-Seidel, by choosing

$$\boldsymbol{M} = \frac{1}{\omega}\boldsymbol{D} + \boldsymbol{L}, \quad \boldsymbol{N} = \Big(\frac{1}{\omega} - 1\Big)\boldsymbol{D} - \boldsymbol{U}$$

In the resulting iterative scheme, we have

$$\Big(\frac{1}{\omega}\boldsymbol{D} + \boldsymbol{L}\Big)\boldsymbol{x}^{(k+1)} = ((1-\omega)\boldsymbol{D} + \boldsymbol{U})\boldsymbol{x}^{(k)}$$

so that if $\boldsymbol{x}_{GS}^{(k+1)}$ is the iterate produced by Gauss-Seidel, SOR instead produces

$$\boldsymbol{x}^{(k+1)} = (1-\omega)\boldsymbol{x}^{(k)} + \omega\boldsymbol{x}_{GS}^{(k+1)}.$$

- The parameter ω in SOR controls the 'step-size' of the iterative method:
  - *over-relaxation corresponds to $\omega > 1$,*
  - *under-relaxation corresponds to $\omega < 1$,*
  - *generally best choice of $\omega \in (0, 2)$ is hard to determine.*

Conjugate Gradient

- The solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a minima of the quadratic optimization problem,

$$\min_{\boldsymbol{x}} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||_2^2$$

We can leverage this and employ optimization methods such as conjugate gradient (CG) in the case when $\boldsymbol{A}$ is SPD.

- Conjugate gradient works by picking $\boldsymbol{A}$-orthogonal descent directions

Ensures search directions make progress and converge in at most n iterations.

- The convergence rate of CG is linear with coefficient $\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$ where $\kappa = \text{cond}(\boldsymbol{A})$:

This convergence rate motivates techniques to improve the conditioning of $\boldsymbol{A}$ to accelerate convergence.

Preconditioning

- Preconditioning techniques choose matrix $\boldsymbol{M} \approx \boldsymbol{A}$ and solve the linear system

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

  For example at each step of CG, we replace multiplication with $\boldsymbol{A}$ by multiplication by $\boldsymbol{M}^{-1}\boldsymbol{A}$:
  - *cost of iteration depends on difficulty of applying $\boldsymbol{M}^{-1}$,*
  - *convergence rate depends on how close $\boldsymbol{M}$ is to $\boldsymbol{A}$.*

- $\boldsymbol{M}$ is a usually chosen to be an effective approximation to $\boldsymbol{A}$ with a simple structure:
  - *Jacobi preconditioning takes $\boldsymbol{M} = \boldsymbol{D}$ where $\boldsymbol{D}$ is the diagonal of $\boldsymbol{A}$,*
  - *incomplete factorization (ILU) uses $\boldsymbol{A} \approx \boldsymbol{L}\boldsymbol{U}$ where the sparsity pattern of $\boldsymbol{L}$ and $\boldsymbol{U}$ is restricted to that of $\boldsymbol{A}$ (factorization is then generally inexact), and employs $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{U}$.*