ECE 4960

Spring 2017

Lecture 11

Matrix Iterative Methods

Edwin C. Kan
School of Electrical and Computer Engineering
Cornell University

Iterative Matrix Solvers

- For VERY large problems where any amount of fill-in cannot be tolerated, we will NEED to use iterative solvers.
- For many circuit simulation and finite-element based simulation, direct solvers are more reliable, and often faster due to the symbolic LU factorization executed only once.
- For $Ax = b$, we can find an initial guess $x^{(0)}$, a T matrix and a constant C vector that the following operation can be performed

$$x^{(k)} = Tx^{(k-1)} + C$$

$$\frac{\left\| b - Ax^{(k)} \right\|}{\left\| b - Ax^{(k-1)} \right\|} < 1$$

Convergence criterion

Residual vector at the k-th iteration: $b - Ax^{(k)}$

Jacobi Iteration Example for Choice of T

$$\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}$$

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

Column pivoting and diagonal conditioning

$$\begin{aligned} x_1 &= \frac{1}{10}x_2 - \frac{1}{5}x_3 + \frac{3}{5} \\ x_2 &= \frac{1}{11}x_1 + \frac{1}{11}x_3 - \frac{3}{11}x_4 + \frac{25}{11} \\ x_3 &= -\frac{1}{5}x_1 + \frac{1}{10}x_2 + \frac{1}{10}x_4 - \frac{11}{10} \\ x_4 &= -\frac{3}{8}x_2 + \frac{1}{8}x_3 + \frac{15}{8} \end{aligned}$$

$$T = \begin{pmatrix} 0 & \frac{1}{10} & -\frac{1}{5} & 0 \\ \frac{1}{11} & 0 & \frac{1}{11} & -\frac{3}{11} \\ \frac{1}{5} & \frac{1}{10} & 0 & \frac{1}{10} \\ 0 & -\frac{3}{8} & \frac{1}{8} & 0 \end{pmatrix}$$

Jacobi and Gauss-Seidel Iterative Solvers

- Decomposing A into $D - L - U$ where D is the diagonal of A, $-L$ is the lower triangular part of A (without diagonal) and $-U$ is the upper triangular part of A (without diagonal)

Jacobi iterative method

$$(D-L-U)x = b \quad \Rightarrow \quad Dx = (L+U)x + b$$

$$x = D^{-1}(L+U)x + D^{-1}b \quad \Rightarrow \quad x^{(k)} = D^{-1}(L+U)x^{(k-1)} + D^{-1}b$$

Gauss-Seidel iterative method

$$x^{(k)} = (D-L)^{-1}Ux^{(k-1)} + (D-L)^{-1}b$$

Convergence: $\left\|(D-L)^{-1}U\right\| < 1$

Hacker Practice

Use the Jacobi iterations to solve the same problem below.

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 3 \\ 4 & 5 & 6 & 0 & 0 \\ 0 & 7 & 8 & 0 & 9 \\ 0 & 0 & 0 & 10 & 0 \\ 11 & 0 & 0 & 0 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Use $D^{-1}b$ as your initial guess and $\|x\|_2 < 10^{-7}$ as the convergence criteria. Observe how many iterations are needed for convergence and what precision has been achieved against the direct solver in your previous hacker practice by checking the evolution of the residual vector: $\|b - Ax^{(k)}\|_2$.

Now check the first and infinite norms of $\|D^{-1}(L+U)\|$. Which one gives a better indication of the convergence property?

Successive Over Relaxation (SOR)

- The objective of the iterative solver is to reduce $\|b - Ax\|_2$, which gives an idea how far we are away from convergence.

SOR method

$$r^{(k)} = b - Ax^{(k)}$$

$$x^{(k)} = x^{(k-1)} + \omega D^{-1} r^{(k-1)}$$

- $\omega < 1$: under-relaxation.
- $\omega > 1$ (e.x., $\omega =$ 2 to 10): **successive over-relaxation (SOR)**

Diagonal Matrix Conditioning

- When the matrix cannot be made into diagonal dominant, in iterative methods, we can choose to solve:

$$\left(A+\xi I\right)x=b$$

- In the beginning of Gauss-Seidel or SOR, we can choose a large ξ to give diagonal dominance, and then gradually decrease ξ to 0.

Hacker Practice

Use the SOR iterations to solve the problem below with $\omega = 2$.

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 3 \\ 4 & 5 & 6 & 0 & 0 \\ 0 & 7 & 8 & 0 & 9 \\ 0 & 0 & 0 & 10 & 0 \\ 11 & 0 & 0 & 0 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Use $D^{-1}b$ as your initial guess and as the convergence criteria. Observe how many iterations are needed for convergence and what precision has been achieved against the direct solver in your previous hacker practices. Print your iterations in the following format:

Iter. k $\quad x_1^{(k)} \quad x_1^{(k+1)} \quad \|\Delta x\|_2 \quad \|r^{(k)}\|_2$

Thoughts on Iterative Matrix Solvers

- These iterative methods can have their specific implementations to make the computation faster for different platforms.
- On a serial platform, during $x^{(k)}$ calculations, we can use as many already known elements in $x^{(k)}$, then the Jacobi iteration will become the Gauss-Seidel iteration.
- This does increase the data dependence within the present iteration, and may not be the most efficient way for vectorized- or parallel-computing platforms.
- For very large matrix computation, the speed of the iterative solver is often specific to the problem and to the platform.

How Does the Iterative Solver Converge?

- Correct answer: Considering $x^{(k-1)}$ has to converge to $x^{(k)}$, the spectral radius (i.e., the magnitude of the largest eigenvalue) of the iterative matrix is < 1.
  - For Jacobi iteration: $||D^{-1}(L+U)|| < 1$
  - For Gauss Seidel iteration: $||(D-L)^{-1}U|| < 1$
  - For SOR: $||1 - \omega D^{-1}A|| < 1$
- Practical answer: Choose **diagonally dominant pivoting** by row and column permutation before applying the iterative method (this is in general called **preconditioning**).
- The preconditioning method for iterative solvers can be life or death just like choice of pivoting in direct solvers, but it is out of the scope of this class.