CMPSC/Math 451, Numerical Computation

Wen Shen

Department of Mathematics, Penn State University

Fixed point iterative solvers for linear systems

Problem: Find approximate solution to $Ax = b$, where $A \in \boldsymbol{R}^{n \times n}$ has properties:

- Large, n is very big, for example $n = \mathcal{O}(10^6)$.
- A is sparse, with a large percent of 0 entries.
- A is structured. (meaning: the product Ax can be computed efficiently)

Idea: Avoiding computing A^{-1}. Perform the "cheap" operation Ax.

Jacobi iterations

Want to solve:
$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{cases}$$

Rewrite it:
$$\begin{cases} x_1 &= \frac{1}{a_{11}}\left(b_1 \qquad - a_{12}x_2 - \cdots - a_{1n}x_n\right) \\ x_2 &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1 \qquad - \cdots - a_{2n}x_n\right) \\ \vdots & \\ x_n &= \frac{1}{a_{nn}}\left(b_2 - a_{n1}x_1 - a_{n2}x_2 - \cdots 0\right) \end{cases}$$

In a compact form:
$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right), \qquad i = 1, 2, \cdots, n$$

This gives the Jacobi iterations:

- Choose a start point, $x^0 = (x_1^0, x_2^0, \cdots, x_n^0)^t$.
- for $k = 0, 1, 2, \cdots$ until stop criteria

  for $i = 1, 2, \cdots, n$

  $$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^k \right)$$

  end

  end

Choices of starting vector x^0: Anything goes.

- A vector with all entries 1: $x_i^0 = 1$ for all i, or $x_i = b_i/a_{ii}$.
- The load vector: $x^0 = b$;
- Best choice: $x_i = b_i/a_{ii}$, for $i = 1, 2, \cdots, n$.

Stop Criteria could be any combinations of the following

- $\left\|x^k - x^{k-1}\right\| \leq \varepsilon$ for certain vector norms.
- Residual $r^k = Ax^k - b$ is small, i.e., $\left\|r^k\right\| \leq \varepsilon$.
- Max number of iteration reached.

About the algorithm:

- Must make 2 vectors for the computation, x^k and x^{k+1}.
- Non-sequential. Great for parallel computing.

Jacobi iterations; example

Example 1. Solve the following system with Jacobi iterations.

$$\begin{cases} 2x_1 - x_2 & = & 0 \\ -x_1 + 2x_2 - x_3 & = & 1 \\ \qquad -x_2 + 2x_3 & = & 2 \end{cases}$$

Given the exact solution $x = (1, 2, 2)^t$.
Answer. Choose x^0 by $x_i^0 = b_i/a_{ii}$:

$$x^0 = (0,\ 1/2,\ 1)^t$$

The iteration is

$$\begin{cases} x_1^{k+1} & = & \frac{1}{2}x_2^k \\ x_2^{k+1} & = & \frac{1}{2}(1 + x_1^k + x_3^k) \\ x_3^{k+1} & = & \frac{1}{2}(2 + x_2^k) \end{cases}$$

We run a couple of iterations, and get

$$
\begin{aligned}
x^1 &= (0.25,\ 1,\ 1.25)^t \\
x^2 &= (0.5,\ 1.25,\ 1.5)^t \\
x^3 &= (0.625,\ 1.5,\ 1.625)^t
\end{aligned}
$$

Observations:

- Looks like it is converging. Need to run more steps to be sure.
- Rather slow convergence rate.

Gauss-Seidal iterations

Recall Jacobi iteration

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1,j\neq i}^{n} a_{ij}x_j^k\right) = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^k - \sum_{j=i+1}^{n} a_{ij}x_j^k\right)$$

If the computation is done in a sequential way, for $i = 1, 2, \cdots$, then in the first summation term, all x_j^k are already computed for step $k+1$. We will replace all these x_j^k with x_j^{k+1}.

Gauss-Seidal iterations

Use the latest computed values of x_i.

for $k = 0, 1, 2, \cdots$, until stop criteria

for $i = 1, 2, \cdots, n$

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right)$$

end

end

About the algorithm:

- Need only one vector for both x^k and x^{k+1}, saves memory space.
- Not good for parallel computing.

Example 2. Try it on the same Example 1, with $x^0 = (0, 0.5, 1)^t$.
The iteration now is:

$$\begin{cases} x_1^{k+1} &= \frac{1}{2}x_2^k \\ x_2^{k+1} &= \frac{1}{2}(1 + x_1^{k+1} + x_3^k) \\ x_3^{k+1} &= \frac{1}{2}(2 + x_2^{k+1}) \end{cases}$$

We run a couple of iterations:

$$\begin{aligned} x^1 &= (0.25,\ 1.125,\ 1.5625)^t \\ x^2 &= (0.5625,\ 1.5625,\ 1.7813)^t \end{aligned}$$

Observation: Converges a bit faster than Jacobi iterations.

SOR (Successive Over Relaxation)

SOR is a more general iterative method.
A version based on Gauss-Seidal.

$$x_i^{k+1} = (1-w)x_i^k + w \cdot \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right)$$

Note the second term is the Gauss-Seidal iteration multiplied with w.
w: relaxation parameter.
Usual value: $0 < w < 2$ (for convergence reason)

- $w = 1$: Gauss-Seidal
- $0 < w < 1$: under relaxation
- $1 < w < 2$: over relaxation

Example 3. Try this on the same example with $w = 1.2$. General iteration is now:

$$\begin{cases} x_1^{k+1} & = & -0.2x_1^k + 0.6x_2^k \\ x_2^{k+1} & = & -0.2x_2^k + 0.6 * (1 + x_1^{k+1} + x_3^k) \\ x_3^{k+1} & = & -0.2x_3^k + 0.6 * (2 + x_2^{k+1}) \end{cases}$$

With $x^0 = (0,\ 0.5,\ 1)^t$, we get

$$\begin{aligned} x^1 &= (0.3, 1.28, 1.708)^t \\ x_2 &= (0.708, 1.8290, 1.9442)^t \end{aligned}$$

Observation: faster convergence than both Jacobi and G-S.

Writing all methods in standard form

Want to solve $Ax = b$. We change it into a fixed-point problem $x = Mx + y$ for some matrix M and vector y, with fixed point iteration $x^{k+1} = Mx^k + y$.

Splitting of the matrix A:

$$A = L + D + U$$

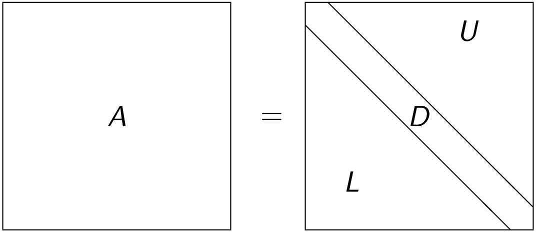

Figure: Splitting of A.

Now we have

$$Ax = (L + D + U)x = Lx + Dx + Ux = b$$

Jacobi iterations:

$$Dx^{k+1} = b - Lx^k - Ux^k$$

so

$$x^{k+1} = D^{-1}b - D^{-1}(L + U)x^k = y_J + M_J x^k$$

where

$$y_J = D^{-1}b, \qquad M_J = -D^{-1}(L + U).$$

Gauss-Seidal:

$$Dx^{k+1} + Lx^{k+1} = b - Ux^k$$

so

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

where

$$y_{GS} = (D+L)^{-1}b, \qquad M_{GS} = -(D+L)^{-1}U.$$

SOR:

$$x^{k+1} = (1-w)x^k + wD^{-1}(b - Lx^{k+1} - Ux^k)$$

$$\Rightarrow \quad Dx^{k+1} = (1-w)Dx^k + wb - wLx^{k+1} - wUx^k$$

$$\Rightarrow \quad (D+wL)x^{k+1} = wb + [(1-w)D - wU]x^k$$

so

$$x^{k+1} = (D+wL)^{-1}b + (D+wL)^{-1}[(1-w)D - wU]x^k = y_{SOR} + M_{SOR}x^k$$

where

$$y_{SOR} = (D+wL)^{-1}b, \qquad M_{SOR} = (D+wL)^{-1}[(1-w)D - wU].$$

Analysis for errors and convergence

Iteration $\quad x^{k+1} = y + Mx^k \quad$ for solving $\quad Ax = b$
Assume s is the solution: $\quad As = b, \quad s = y + Ms.$
Define the error vector: $\quad e^k = x^k - s$

$$e^{k+1} = x^{k+1} - s = y + Mx^k - (y + Ms) = M(x^k - s) = Me^k.$$

This gives the propagation of error:

$$e^{k+1} = M\, e^k.$$

Take the norm on both sides:

$$\left\| e^{k+1} \right\| = \left\| Me^k \right\| \le \|M\| \cdot \left\| e^k \right\|$$

This implies:

$$\left\| e^k \right\| \le \|M\|^k \left\| e^0 \right\|, \qquad e^0 = x^0 - s.$$

Theorem *If* $\|M\| < 1$ *for some norm* $\|\cdot\|$, *then the iterations converge in that norm.*

NB! Convergence only depends on the iteration matrix M.

Check our methods: $A = D + L + U$.

- Jacobi: $M_J = -D^{-1}(L + U)$. Determined by A;
- G-S: $M_{GS} = -(D + L)^{-1}U$. Determined by A;
- SOR: $M_{SOR} = (D + wL)^{-1}[(1 - w)D - wU]$.
  One can adjust w to get a smallest possible $\|M\|$.
  More flexible.

Check the same example we have been using. We have

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix},$$

$$L = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \qquad U = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}.$$

The iteration matrix for each method:

$$M_J = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{pmatrix}, \qquad M_{GS} = \begin{pmatrix} 0 & 0.5 & 0 \\ 0 & 0.25 & 0.5 \\ 0 & 0.125 & 0.25 \end{pmatrix},$$

$$M_{SOR} = \begin{pmatrix} -0.2 & 0.6 & 0 \\ -0.12 & 0.16 & 0.6 \\ -0.072 & 0.096 & 0.16 \end{pmatrix}$$

We list their various norms:

M	l_1 norm	l_2 **norm**	l_∞ norm
Jacobi	1	**0.707**	1
G-S	0.875	**0.5**	0.75
SOR	0.856	**0.2**	0.88

The l_2 norm is the most significant one. We see now why SOR converges fastest.

Convergence Theorem. *If A is diagonal dominant, i.e.,*

$$|a_{ii}| > \sum_{j=1, j \neq i}^{n} |a_{ij}|, \qquad \textit{for every} \;\; i = 1, 2, \cdots, n.$$

Then, all three iteration methods converge for all initial choice of x^0.

NB! If A is not diagonal dominant, it might still converge, but there is no guarantee.