Lecture 5
Successive Overrelaxation Method (SOR)

Jinn-Liang Liu

2017/4/18

The SOR is devised by applying extrapolation to GS. This extrapolation takes the form of a weighted average between the previous iterate and the current GS iterate successively for each component

$$x_i^{(k)} = \omega \overline{x_i}^{(k)} + (1-\omega) x_i^{(k-1)} \tag{5.1}$$

where $\overline{x_i}^{(k)}$ denotes a GS iterate for the i^{th} component of $\overline{x}^{(k)} = (\overline{x_1}^{(k)}, \overline{x_2}^{(k)}, \cdots, \overline{x_N}^{(k)})^T$, and ω is the extrapolation (weighting) factor. The idea is to choose a value for ω that will accelerate the rate of convergence

$$(5.1) \quad \Leftrightarrow \quad D\overrightarrow{x}^{(k)} = \omega D\overline{x}^{(k)} + (1-\omega) D\overrightarrow{x}^{(k-1)} \tag{5.2}$$

$$\therefore \quad D\overrightarrow{x}^{(k)} = \omega[\overrightarrow{b} - L\overrightarrow{x}^{(k)} - U\overrightarrow{x}^{(k-1)}] + (1-\omega) D\overrightarrow{x}^{(k-1)} \tag{5.3}$$

In matrix form, the SOR is written as

$$[D+\omega L]\overrightarrow{x}^{(k)} = \omega\overrightarrow{b} + [-\omega U + (1-\omega)D]\overrightarrow{x}^{(k-1)} \tag{5.4}$$

$$\overrightarrow{x}^{(k)} = B\overrightarrow{x}^{(k-1)} + \overrightarrow{c} \tag{5.5}$$

and hence

$$B = [D+\omega L]^{-1}[(1-\omega)D - \omega U], \qquad \overrightarrow{c} = \omega[D+\omega L]^{-1}\overrightarrow{b}. \tag{5.6}$$

Case 1 $\quad \omega = 1 \Rightarrow SOR = GS$

Case 2 $\quad \omega = 0 \Rightarrow$ *No iteration*

Case 3 $\quad 0 < \omega < 1 \Rightarrow$ *Underrelaxation*

Case 4 $\quad 1 < \omega < 2$ *or* $0 < \omega < 2 \Rightarrow$ *Overrelaxation*

Case 5 $\quad \omega \geq 2 \Rightarrow$ *Divergent*

Algorithm SOR: Successive Overrelaxation Method Solve $A\vec{x} = \vec{b}$.

Input: N: Number of unknowns and equations; a_{ij}: Entries of A, $i, j = 1 \cdots N$; b_i: Entries of $\vec{b}$, $i = 1 \cdots N$. TOL: Error Tolerance; $\omega = 1.3$ (for example).

Output: $x_i^{(k)}$: Entries of $\vec{x}^{(k)}$(approximate solution) or Error Message.

Step 1. Choose an initial guess $\vec{x}^{(0)}$to the solution $\vec{x}$.

Step 2. For $k = 1, 2, 3 \cdots, k_{\max}$

Step 3. For $i = 1, 2, \cdots, N$

Step 4. $\sigma = 0$

Step 5. For $j = 1, 2, \cdots, i-1$

Step 6. $\sigma = \sigma + a_{ij} x_j^{(k)}$

Step 7. End j loop

Step 8. For $j = i+1, ..., N$

Step 9. $\sigma = \sigma + a_{ij} x_j^{(k-1)}$

Step 10. End j loop

Step 11. $\sigma = (b_i - \sigma)/a_{ii}$ $[\sigma = \overline{x_i}^{(k)}$ in (5.1)]

Step 12. $x_i^{(k)} = \omega\sigma + (1-\omega) x_i^{(k-1)}$

Step 13. End i loop

Step 14. If $||\vec{r}^{(k)}||_\infty < \text{TOL} = 10^{-6}$ then Stop otherwise Set $\vec{x}^{(k-1)} = \vec{x}^{(k)}$and Go To Step 2.

Step 15. End k loop

Step 16. Error: Not convergent with the max number of iterations $k_{\max}$ and TOL.

Project 5.1. Consider the 1D Poisson Problem (1.1) (with $f(x) = 2$, $g_D = 0$, and $g_N = 0$) and implement the methods FDM and SOR.

Input: N, A, $\vec{b}$, $k_{\max}$, TOL, ω (write the input in the program).

Output:

N	k	$E^{\vec{x}}$	E^u	α
5				
9				
17				
33				
65				
129				