# Lecture 6
# Symmetric SOR (SSOR)

Jinn-Liang Liu

2017/4/18

**Example 6.1.** Consider the linear system

$$\begin{bmatrix} 1 & 2 \\ 2 & 3.999 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 7.999 \end{bmatrix}, \quad (A\overrightarrow{x} = \overrightarrow{b}) \tag{6.1}$$

The solution is $\overrightarrow{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. Making a small change in the right hand side of the equations to

$$\begin{bmatrix} 1 & 2 \\ 2 & 3.999 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4.001 \\ 7.998 \end{bmatrix}, \quad (A\widetilde{x} = \widetilde{b}) \tag{6.2}$$

gives the solution $\widetilde{x} = \begin{bmatrix} -3.999 \\ 4 \end{bmatrix}$. We only perturb $\overrightarrow{b} = \begin{bmatrix} 4 \\ 7.999 \end{bmatrix}$ to $\widetilde{b} = \begin{bmatrix} 4.001 \\ 7.998 \end{bmatrix}$, why does the solution $\overrightarrow{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ change to $\widetilde{x} = \begin{bmatrix} -3.999 \\ 4 \end{bmatrix}$ by so much? ($\left\| \overrightarrow{b} - \widetilde{b} \right\|_\infty =?$, $\left\| \overrightarrow{x} - \widetilde{x} \right\|_\infty =?$)

The **condition number** associated with the linear system

$$A\overrightarrow{x} = \overrightarrow{b} \tag{6.3}$$

gives a bound on how inaccurate the approximation of $\overrightarrow{x}$ will be when the system is solved by an approximation method. Note that for iterative methods such as JM, GS, and SOR we only obtain an approximate solution $\overrightarrow{x}^{(k)}$ to the exact solution $\overrightarrow{x}$. Another way to view this is that the vector $\overrightarrow{b}$ is perturbed to $\widetilde{b}$ so that

$$A\overrightarrow{x}^{(k)} = \widetilde{b}. \tag{6.4}$$

The condition number of (6.1) denoted by Cond($A$) is defined to be the maximum ratio of the relative error in $\overrightarrow{x}$ divided by the relative error in $\overrightarrow{b}$ in some norm $\|\cdot\|$, i.e.,

$$\text{Cond}(A) = \max_{\overrightarrow{b}} \frac{\left\| \overrightarrow{x} - \overrightarrow{x}^{(k)} \right\| \left\| \overrightarrow{b} \right\|}{\left\| \overrightarrow{x} \right\| \left\| \overrightarrow{b} - \widetilde{b} \right\|}. \tag{6.5}$$

So now the question is: If the data $\overrightarrow{b}$ is perturbed a little bit, will we get very large error in $\overrightarrow{x}$? If yes, we say that the matrix $A$ is *ill-conditioned* and is *well-conditioned* otherwise. The larger the Cond$(A)$, the more ill-condition of $A$ will be. Further computations on (6.5) yield

$$\begin{aligned}
\text{Cond}(A) &= \max \frac{\left\|A^{-1}\overrightarrow{b} - A^{-1}\widetilde{b}\right\| \left\|\overrightarrow{b}\right\|}{\left\|A^{-1}\overrightarrow{b}\right\| \left\|\overrightarrow{b} - \widetilde{b}\right\|} = \max \frac{\left\|A^{-1}\overrightarrow{b} - A^{-1}\widetilde{b}\right\|}{\left\|\overrightarrow{b} - \widetilde{b}\right\|} \frac{\left\|\overrightarrow{b}\right\|}{\left\|A^{-1}\overrightarrow{b}\right\|} \\
&= \max \frac{\left\|A^{-1}\overrightarrow{b} - A^{-1}\widetilde{b}\right\|}{\left\|\overrightarrow{b} - \widetilde{b}\right\|} \frac{\|A\overrightarrow{x}\|}{\|\overrightarrow{x}\|} = \|A^{-1}\| \cdot \|A\| \qquad (6.6)
\end{aligned}$$

where the matrix norm of any matrix $A$ is defined by

$$\begin{aligned}
\|A\| &= \max\left\{\|A\overrightarrow{y}\| : \text{ for any } \overrightarrow{y} \in \mathcal{R}^N \text{ with } \|\overrightarrow{y}\| \le 1\right\} \\
&= \max_{\overrightarrow{y} \neq 0} \frac{\|A\overrightarrow{y}\|}{\|\overrightarrow{y}\|} \qquad (6.7)
\end{aligned}$$

**Theorem 6.1.** Let $A$ be an $m \times n$ real matrix. Then

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}| \quad \text{(the maximum of absolute row sums).} \qquad (6.8)$$

**Example 6.2.** Find the condition number of $A$ in Example 6.1.

$$\begin{aligned}
A &= \begin{bmatrix} 1 & 2 \\ 2 & 3.999 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} -3999 & 2000 \\ 2000 & -1000 \end{bmatrix}, \qquad (6.9) \\
\|A\|_\infty &= 5.999, \quad \left\|A^{-1}\right\|_\infty = 5999, \\
\text{Cond}(A) &= \|A\|_\infty \left\|A^{-1}\right\|_\infty = 5.999 \times 5999 \approx 36000. \qquad (6.10)
\end{aligned}$$

It is very large and hence (6.1) is very ill-conditioned.

**Question**: If we are given an ill system, can we make it better before solving it?

**Example 6.3.** For the system

$$\begin{bmatrix} 1 & 2 \\ 0 & 10^{-20} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 10^{-20} \end{bmatrix}, \qquad (6.11)$$

can you make it better conditioned without changing the solution? Compare the condition numbers between the old and new systems.

A **preconditioner** $P$ of a matrix $A$ is a matrix such that $P^{-1}A$ has a smaller condition number than $A$. Preconditioners are useful when using an iterative method to solve a large, sparse linear system for $\overrightarrow{x}$ since the rate of convergence for most iterative linear solvers degrades as the condition number of a matrix increases. Instead of solving the original linear system (6.1), one may solve either the left preconditioned system via

$$P^{-1}A\overrightarrow{x} = P^{-1}\overrightarrow{b} \tag{6.12}$$

or the right preconditioned system via

$$AP^{-1}\overrightarrow{y} = \overrightarrow{b}, \quad P^{-1}\overrightarrow{y} = \overrightarrow{x} \tag{6.13}$$

in which we hope that the new matrix $P^{-1}A$ or $AP^{-1}$ is much better conditioned than $A$ provided that the computation of the new matrix is efficient. The three systems (6.1), (6.12), and (6.13) are equivalent so long as the preconditioner matrix $P$ is nonsingular.

**Example 6.4.** What is your preconditioner for Example 6.3?

Replacing $\overrightarrow{x}^{(k)}$ and $\overrightarrow{x}^{(k-1)}$ by $\overrightarrow{x}$, (3.7) is written as

$$\overrightarrow{x} = -D^{-1}(L+U)\overrightarrow{x} + D^{-1}\overrightarrow{b} \tag{6.14}$$

which is equivalent to

$$D^{-1}A\overrightarrow{x} = D^{-1}\overrightarrow{b} \tag{6.15}$$

Therefore, $D^{-1}$ is the **Jacobi preconditioner** of the matrix $A$, which is one of the simplest forms of preconditioning. The preconditioners of $A$ so far are:

Table 6.1. Preconditioners of $A$.

| | | |
|---|---|---|
| JM | $A^{-1} \approx D^{-1} =: P_{\text{JM}}^{-1}$ | Symmetric |
| GS | $A^{-1} \approx (D+L)^{-1} =: P_{\text{GS}}^{-1}$ | Non-symmetric |
| SOR | $A^{-1} \approx (D+wL)^{-1} =: P_{\text{SOR}}^{-1}$ | Non-symmetric |
| SSOR | $A^{-1} \approx (D+wL)^{-1}(D+wU)^{-1} =: P_{\text{SSOR}}^{-1}$ | Symmetric? |

The convergence rate of iterative methods depends on spectral properties of the coefficient matrix $A$. $A\overrightarrow{x} = \lambda\overrightarrow{x}$, $(\lambda_i, \overrightarrow{x_i})$ is an eigenpair of $A$ if $A\overrightarrow{x}_i = \lambda_i\overrightarrow{x}$ and $\overrightarrow{x}_i \neq 0$. The spectral radius of $A$ is defined as $\rho(A) = \max_{1 \leq i \leq N} |\lambda_i|$ and the spectrum of $A$ is denoted by $\sigma(A) = \{\lambda_i\}_{i=1}^{N}$. Hence one way attempt to transform $A\overrightarrow{x} = \overrightarrow{b}$ into one that is equivalent in the sense that it has the same solution, but that has more favorable spectral properties.

If we assume that the coefficient matrix $A$ is *symmetric*, then **SSOR** combines two SOR sweeps (a forward SOR sweep followed by a backward SOR sweep) together in such a way that the resulting iteration matrix is similar to a symmetric matrix. We say that

$$A \sim B, \text{ if } \exists Q \text{ s.t. } Q^{-1}BQ = A.$$

The similarity of the SSOR iteration matrix to a symmetric matrix permits the application of SSOR as a *preconditioner* for other iterative schemes for symmetric matrices. Indeed, this is the *primary motivation* for SSOR since its convergence rate, with an optimal value of $\omega$, is usually *slower* than the convergence rate of SOR with optimal $\omega$.

| Table 6.2. Iterative Methods in Component Form | |
|---|---|
| JM | $x_i^{(k)} = (b_i - \sum_{i\neq j} a_{ij}x_j^{(k-1)})/a_{ii}$ |
| GS | $x_i^{(k)} = (b_i - \sum_{j<i} a_{ij}x_j^{(k)} - \sum_{j>i} a_{ij}x_j^{(k-1)})/a_{ii}$ |
| SOR | $x_i^{(k)} = \omega\overline{x}_i^{(k)} + (1-\omega)x_i^{(k-1)}$ |
| FGS | $x_i^{(k)} = (b_i - \sum_{j<i} a_{ij}x_j^{(k)} - \sum_{j>i} a_{ij}x_j^{(k-1)})/a_{ii}$ |
| BGS | $x_i^{(k)} = (b_i - \sum_{j>i} a_{ij}x_j^{(k)} - \sum_{j<i} a_{ij}x_j^{(k-1)})/a_{ii}$ |

| Table 6.3. Iterative Methods in Matrix Form | |
|---|---|
| JM | $D\overrightarrow{x}^{(k)} = -(L+U)\overrightarrow{x}^{(k-1)} + \overrightarrow{b}$ |
| GS | $(D+L)\overrightarrow{x}^{(k)} = -U\overrightarrow{x}^{(k-1)} + \overrightarrow{b}$ |
| SOR | $(D+\omega L)\overrightarrow{x}^{(k)} = (-\omega U + (1-\omega)D)\overrightarrow{x}^{(k-1)} + \omega\overrightarrow{b}$ |
| FGS | $(D+L)\overrightarrow{x}^{(k)} = -U\overrightarrow{x}^{(k-1)} + \overrightarrow{b}$ |
| BGS | $(D+U)\overrightarrow{x}^{(k)} = -L\overrightarrow{x}^{(k-1)} + \overrightarrow{b}$ |
| SSOR | $\overrightarrow{x}^{(k)} = B_1B_2\overrightarrow{x}^{(k-1)} + \omega(2-\omega)(D+\omega U)^{-1}D(D+\omega L)^{-1}\overrightarrow{b}$<br>$B_1 = (D+\omega U)^{-1}(-\omega L + (1-\omega)D)$ : Backward SOR Sweep<br>$B_2 = (D+\omega L)^{-1}(-\omega U + (1-\omega)D)$ : Forward SOR Sweep |

***Algorithm SSOR: Symmetric Successive Overrelaxation Method***

**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 $i = 1, 2, \cdots, N$ (Forward)

**Step 4.** $\sigma = 0$

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

**Step 6.** $\sigma = \sigma + a_{ij} x_j^{(k-\frac{1}{2})}$

**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.** $\sigma = (b_i - \sigma)/a_{ii}$

**Step 12.** $x_i^{(k-\frac{1}{2})} = \omega\sigma + (1 - \omega) x_i^{(k-1)}$

**Step 13.** For $i = N, N - 1, ......, 1$ (Backward)

**Step 14.** $\sigma = 0$

**Step 15.** For $j = 1, 2, ......, i - 1$

**Step 16.** $\sigma = \sigma + a_{ij} x_j^{(k-\frac{1}{2})}$

**Step 17.** End $j$ loop

**Step 18.** For $j = i + 1, ......, N$

**Step 19.** $\sigma = \sigma + a_{ij} x_j^{(k)}$

**Step 20.** End $j$ loop

**Step 21.** $\sigma = (b_i - \sigma)/a_{ii}$

**Step 22.** $x_i^{(k)} = \omega\sigma + (1-\omega)x_i^{(k-\frac{1}{2})}$

**Step 23.** End $i$ loop

**Step 24.** If $||\overrightarrow{r}^{(k)}||_\infty < \text{TOL} = 10^{-6}$ then Stop otherwise Set $\overrightarrow{x}^{(k-1)} = \overrightarrow{x}^{(k)}$ and Go To Step 2.

**Step 25.** End $k$ loop

**Step 26.** Error: Not convergent with the max number of iterations $k_{\max}$ and TOL.

**Project 6.1.** Consider the 1D Poisson Problem (1.1) (with $f(x) = 2$, $g_D = 0$, and $g_N = 0$) and implement the methods FDM and SSOR.

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

**Output:**

| $N$ | $k$ | $E^{\overrightarrow{x}}$ | $E^u$ | $\alpha$ |
|---|---|---|---|---|
| 5 | | | | |
| 9 | | | | |
| 17 | | | | |
| 33 | | | | |
| 65 | | | | |
| 129 | | | | |