[6] Indirect solvers : iterative methods for sparse linear systems

→ We are going to look at a couple of simple & basic methods first, including:

(i) Jacobi,
(ii) Gauss-Seidel,
(iii) Successive overrelaxation (SOR),

and later we also study more effective ones:

(iv) Conjugate gradient (CG),
(v) descent method,
(vi) preconditioner,
(v) multigrid methods, etc

§6.1. Stationary iterative methods

→ Before we begin studying iterative methods, let's first take a look at "stationary iterative methods" by splitting the matrix $\underset{\sim}{A}$.

→ Consider we are interested in solving a linear system

$$\boxed{\underset{\sim}{A}U = f} \quad \cdots \quad (1)$$

as a result of discretizing the elliptic eqn:

$$u_{xx} + u_{yy} = f.$$

→ the simplest type of iterative method for solving (1) has the form

$$\boxed{U^{(k+1)} = \underset{\sim}{G}U^{(k)} + c} \quad \cdots \quad (2)$$

where the matrix $\underset{\sim}{G}$ and the vector c are chosen so that a fixed pt. of the equation

$$\boxed{U = \underset{\sim}{G}U + c} \quad \cdots \quad (3)$$

is a soln to (1).

→ (3) is called stationary if $\underset{\sim}{G}$ and c are constant over all iterations.

→ One way to obtain $\underset{\sim}{G}$ is by splitting $\underset{\sim}{A}$ into

$$\boxed{\underset{\sim}{A} = \underset{\sim}{M} - \underset{\sim}{N}}, \quad \underset{\sim}{M}: \text{nonsingular}. \quad \cdots (4)$$

→ $\underset{\sim}{A}U = f \iff \underset{\sim}{M}U - \underset{\sim}{N}U = f$

$\iff U - \underset{\sim}{M}^{-1}\underset{\sim}{N}U = \underset{\sim}{M}^{-1}f$

$\iff$ Setting $\underset{\sim}{G} = \underset{\sim}{M}^{-1}\underset{\sim}{N}$, & $c = \underset{\sim}{M}^{-1}f$, we get the iteration scheme as

$$U^{(k+1)} = \underset{\sim}{G}U^{(k)} + c, \quad \cdots ②$$

which can be implemented as

$$\boxed{\underset{\sim}{M}U^{(k+1)} = \underset{\sim}{N}U^{(k)} + f} \quad \cdots ⑤$$

→ In ⑤ we solve a linear system with $\underset{\sim}{M}$ at each k, instead of solving the original system with $\underset{\sim}{A}$.

→ From ②, we see that the iteration scheme converges if

$$\boxed{\rho(\underset{\sim}{G}) = \rho(\underset{\sim}{M}^{-1}\underset{\sim}{N}) < 1}, \text{ where}$$

$$\rho(\underset{\sim}{G}) = \max\{|\lambda| \mid \lambda : \text{eigenvalues of } \underset{\sim}{G}\}$$

is the spectral radius.

→ The smaller the $\rho(\underset{\sim}{G})$, the faster the convergence. For rapid convergence, we should choose $\underset{\sim}{M}$ & $\underset{\sim}{N}$ s.t $\rho(\underset{\sim}{M}^{-1}\underset{\sim}{N})$ is as small as possible.

→ Note that the cost of solving ⑤ is determined by the linear system with $\underset{\sim}{M}$.

→ As an extereme case, if $\underset{\sim}{M} = \underset{\sim}{A}$, then the scheme converges in a single iteration,

→ In practice, $\underset{\sim}{M}$ is chosen to approximate $\underset{\sim}{A}$ in some sense, but is usually constrained to have a simple form, such as diagonal or triangular, so that the linear system (5) is easy to solve.

§6.2. Jacobi Method:

→ The simplest choice for $\underset{\sim}{M} = \text{diag}(\underset{\sim}{A})$, $\underset{\sim}{A} = \underset{\sim}{M} - \underset{\sim}{N}$.

Let's have the followings:

(i) $\underset{\sim}{D} \equiv \text{diag}(\underset{\sim}{A})$,

(ii) $\underset{\sim}{L} \equiv$ strict lower matrix of $\underset{\sim}{A}$,

(iii) $\underset{\sim}{U} \equiv$ strict upper matrix of $\underset{\sim}{A}$,

then

(iv) $\boxed{\underset{\sim}{M} = \underset{\sim}{D}}$ ··· ①

(v) $\underset{\sim}{A} = \underset{\sim}{M} - \underset{\sim}{N} = \underset{\sim}{D} - \underset{\sim}{N}$

$\Rightarrow \underset{\sim}{L} + \underset{\sim}{D} + \underset{\sim}{U}$

$\Rightarrow \boxed{\underset{\sim}{N} = -(\underset{\sim}{L} + \underset{\sim}{U})}$ ··· ②

→ ① & ② give the splitting of $\underset{\sim}{A}$.

→ $\underset{\sim}{D}$ is invertible if $\underset{\sim}{A}$ has no zero entries in diagonal.

→ In this case, we obtain the iterative scheme, known as the Jacobi method:

$$\boxed{\underset{\sim}{U}^{(k+1)} = -\underset{\sim}{D}^{-1}(\underset{\sim}{L} + \underset{\sim}{U})\,\underset{\sim}{U}^{(k)} + \underset{\sim}{D}^{-1}\underset{\sim}{f}} \quad \cdots ③$$

→ To write ③ in the componentwise form, we revisit Eqn ⑤ (⑤a ~ ⑤e) in [3] ordering the unknowns & eqns.

→ We reorganize (5) into ;

(5b) (5c) = (5e) − ((5a) + (5d)) , or

$$\begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & 1 & -4 & 1 & \\ & & & 1 & -4 \end{bmatrix} \begin{bmatrix} U_{1,j} \\ U_{2,j} \\ \vdots \\ U_{i,j} \\ \vdots \\ U_{m,j} \end{bmatrix} = h^2 \begin{bmatrix} f_{1,j} \\ f_{2,j} \\ \vdots \\ f_{i,j} \\ \vdots \\ f_{m,j} \end{bmatrix} - \begin{bmatrix} U_{1,j+1} + U_{1,j+1} \\ U_{2,j+1} + U_{2,j+1} \\ \vdots \\ U_{i,j+1} + U_{i,j+1} \\ \vdots \\ U_{m,j+1} + U_{m,j+1} \end{bmatrix} \leftarrow (i\text{th})$$

$(= \underset{\sim}{A})$ $(= U_j)$ $(= f_j)$

$$\rightarrow \boxed{U_{ij}^{(k+1)} = \frac{1}{4}\left(U_{i-1,j}^{(k)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k)} + U_{i,j+1}^{(k)} \right) - \frac{h^2}{4} f_{ij}} \quad \cdots (4)$$

→ (4) can be thought that the new approximate $U_{ij}^{(k+1)}$ is simply the average of the previous soln components at the four neighboring grid pts.

Rmk.

① The Jacobi method does NOT always converge, but it is guranteed to converge under conditions that are often satisfied in practice (e.g., if the matrix is diagonally dominant by rows,

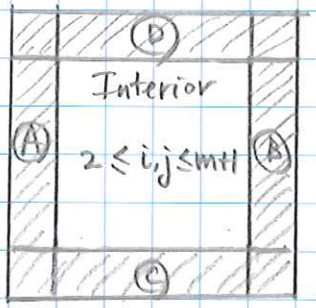
$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \,, \quad \forall i \,)$$

② Unfortunately, the convergence rate of the Jacobi method is in general unacceptably slow.

Rmk. A pseudo-code for Jacobi looks like;

```
for iter = 0 ; maxIter
    for j = 2 ; m+1
        for i = 2 ; m+1
            unew(i,j) = 0.25*( u(i-1,j) + u(i+1,j) + u(i,j-1) + u(i,j+1) )
                        - h^2 * f(i,j)
        end for
    end for
    u = unew
end for
```

→ BCs are stored in those four GCs band regions



GCs
- Ⓐ → u(1, :)
- Ⓑ → u(m+2, :)
- Ⓒ → u(:, 1)
- Ⓓ → u(:, m+2)

→ Two arrays u & unew need to be used.

§6.3. Gauss - Seidel Method.

→ There may be some who wish to save the unew array space from the Jacobi method, and try the following:

```
for iter = 0 : maxIter
    for j = 2 : m+1
        for i = 2 : m+1
            u(i,j) = 0.25*(u(i-1,j) + u(i+1,j) + u(i,j-1) + u(i,j+1))
                     - h^2 * f(i,j)
        end for
    end for
end for
```

→ This will give a different result though, which is known as the Gauss - Seidel method, given as

$$\boxed{v_{ij}^{(k+1)} = \frac{1}{4}\left(v_{i-1,j}^{(k+1)} + v_{i+1,j}^{(k)} + v_{i,j-1}^{(k+1)} + v_{i,j+1}^{(k)}\right) - \frac{h^2}{4} f_{ij}} \quad \cdots (1)$$

→ This can be considered as a lucky coding error! However, the method does work!

→ The major difference from the Jacobi method is to use the already updated data at $(i-1,j)$ & $(i,j-1)$ in order to update the data at (i,j).

→ Mathematically, this idea of using the latest updated information to update the new grid data makes the Gauss-Seidel scheme faster in convergence than the Jacobi method.

→ In general, one can write the GS method in matrix terms using the splitting of the form:

$$\begin{cases} M = D + L, \\ N = -U, \end{cases}$$

(updated information)

which corresponds to the matrix form:

$$\begin{aligned} v^{(k+1)} &= (D+L)^{-1}(-U v^{(k)}) + (D+L)^{-1} f \\ &= (D+L)^{-1}(f - U v^{(k)}). \end{aligned}$$

Rmk ① GS is faster than Jacobi in convergence.

② GS only requires one vector array $u(:)$.

③ GS needs the updates successively, whereas Jacobi can update its soln in any order. Hence, Jacobi is easier for parallelization.

④ GS does NOT always converge, but can converge under a weaker condition than Jacobi (e.g., if the matrix is SPD), but still too slow.

SPD: symmetric positive definite

§6.4. Successive Overrelaxation.

→ The convergence rate of GS can be accelerated by a technique called "successive overrelaxation (SOR)".

→ The idea is based on the fact that
(i) GS update moves $U_{ij}^{(k)}$ in the right direction, but
(ii) the direction is too conservative

→ So let's be bit more aggressive by pushing the GS update bit further (i.e., overrelaxing, or extrapolation based on the GS update).

→ Step 1: $U_{ij}^{(k+1),GS} = \frac{1}{4}\left(U_{i+1,j}^{(k+1)} + U_{i+1,j}^{(k)} + U_{i,j-1}^{(k+1)} + U_{i,j+1}^{(k)}\right) - \frac{h^2}{4} f_{ij}$

(GS update)

Step 2: $U_{ij}^{(k+1)} = U_{ij}^{(k)} + \omega\left(U_{ij}^{(k+1),GS} - U_{ij}^{(k)}\right)$

(relaxation)

→ Note that Step 2 is equivalent to

$$U_{ij}^{(k+1)} = (1-\omega)\, U_{ij}^{(k)} + \omega\, U_{ij}^{(k+1),GS} \quad \cdots (1)$$

→ (1) is then a weighted average of
(i) the current $U_{ij}^{(k)}$, &
(ii) the updated GS soln $U_{ij}^{(k+1),GS}$.

→ We note that from ①,

(i) $\omega < 1 \Rightarrow$ interpolation, or <u>under-relaxation</u>,

(ii) $\omega > 1 \Rightarrow$ extrapolation, or <u>over-relaxation</u>.

(iii) $\omega = 1 \Rightarrow$ GS method.

(iv) $\omega = 2 \Rightarrow$ Not converge!

→ We always have $0 < \omega < 2$ for convergence.

→ Chossing a specific value of ω to attain the best possible convergence rate is a difficult problem in general.

→ One also can write SOR in the matrix form using

(i) $\underset{\sim}{M} = \frac{1}{\omega}\underset{\sim}{D} + \underset{\sim}{L}$,

(ii) $\underset{\sim}{N} = \left(\frac{1}{\omega} - 1\right)\underset{\sim}{D} - \underset{\sim}{U}$,

resulting in the corresponding form of:

$$\underset{\sim}{v}^{(k+1)} = (\underset{\sim}{D} + \omega\underset{\sim}{L})^{-1}\left[(1-\omega)\underset{\sim}{D} - \omega\underset{\sim}{U}\right]\underset{\sim}{v}^{(k)} + \omega(\underset{\sim}{D} + \omega\underset{\sim}{L})^{-1}\underset{\sim}{f}.$$