Project of Numerical Analysis

February 18, 2014

Consider the Dirichlet boundary-value problem:

$$
\begin{aligned}
&-\Delta u \equiv -u_{xx} - u_{yy} = 2\pi^2 \sin \pi x \sin \pi y, \text{ for } (x, y) \in \Omega, \qquad (1)\\
&u(x, y) = 0 \ \ (x, y) \in \partial\Omega,
\end{aligned}
$$

for $\Omega := \{x, y | 0 < x, y < 1\} \subseteq \mathbb{R}^2$ with boundary $\partial\Omega$, which has the exact solution

$$u(x, y) = \sin \pi x \sin \pi y,$$

and is shown in Figure 1.

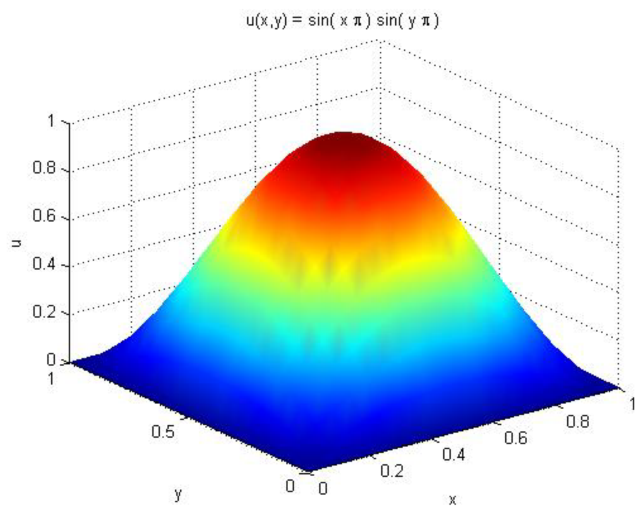

Figure 1: Exact solution.

1 Center difference discretization

To solve (1) by means of a difference methods, one replaces the differential operator by a difference operator. Let

$$\begin{aligned}
\Omega_h &:= \{(x_i, y_i) | i, j = 1, \ldots, n\}, \\
\partial\Omega_h &:= \{(x_i, 0), (x_i, 1), (0, y_j), (1, y_j) | i, j = 0, 1, \ldots, n+1\},
\end{aligned}$$

where $x_i = ih,\ y_j = jh,\ i, j = 0, 1, \ldots, n+1,\ h := \frac{1}{n+1},\ n \geq 1$, is an integer. From the Taylor's theorem, we have

$$\begin{aligned}
u(x_i + h) &= u(x_i) + u'(x_i)h + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u'''(x_i) + \frac{h^4}{24}u^{(4)}(\xi_1) \\
u(x_i - h) &= u(x_i) - u'(x_i)h + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u'''(x_i) + \frac{h^4}{24}u^{(4)}(\xi_2),
\end{aligned}$$

where ξ_1 is between x_i and $x_i + h$ and ξ_2 is between x_i and $x_i - h$. Hence

$$\begin{aligned}
u''(x_i) &= \frac{u(x_i + h) - 2u(x_i) + u(x_i - h)}{h^2} - \frac{h^2}{12}u^{(4)}(\xi) \\
&= \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - \frac{h^2}{12}u^{(4)}(\xi),
\end{aligned}$$

where ξ is between $x_i - h$ and $x_i + h$. Similarly,

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2}(x_i, y_j) &= \frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j), \\
\frac{\partial^2 u}{\partial y^2}(x_i, y_j) &= \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(x_i, \eta_j),
\end{aligned}$$

where $\xi_i \in (x_{i-1}, x_{i+1})$ and $\eta_j \in (y_{j-1}, y_{j+1})$. It implies that

$$\begin{aligned}
&\frac{\partial^2 u}{\partial x^2}(x_i, y_j) + \frac{\partial^2 u}{\partial y^2}(x_i, y_j) \\
=&\frac{u(x_i, y_{j-1}) + u(x_{i-1}, y_j) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1})}{h^2} \\
&- \frac{h^2}{12}\left[\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{\partial^4 u}{\partial x^4}(x_i, \eta_j)\right].
\end{aligned}$$

Let u_{ij} denote an approximated value of function u at the grid point (x_i, y_j) for $i, j = 1, \ldots, n+1$. Then

$$-u_{xx}(x_i, y_j) - u_{yy}(x_i, y_j) \approx \frac{-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}}{h^2}$$

with an error $O(h^2)$ and the equation

$$-u_{xx}(x_i, y_j) - u_{yy}(x_i, y_j) = 2\pi^2 \sin \pi x_i \sin \pi y_j \equiv f_{ij}$$

can be replaced by the following equation

$$\frac{-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}}{h^2} = f_{ij} \tag{2}$$

for $i, j = 1, \ldots, n$.

For $j = 1$, we have

$$
\begin{aligned}
-u_{1,0} - u_{0,1} + 4u_{1,1} - u_{2,1} - u_{1,2} =& h^2 f_{1,1}, &\text{(3a)}\\
-u_{2,0} - u_{1,1} + 4u_{2,1} - u_{3,1} - u_{2,2} =& h^2 f_{2,1}, &\text{(3b)}\\
&\vdots\\
-u_{n-1,0} - u_{n-2,1} + 4u_{n-1,1} - u_{n,1} - u_{n-1,2} =& h^2 f_{n-1,1}, &\text{(3c)}\\
-u_{n,0} - u_{n-1,1} + 4u_{n,1} - u_{n+1,1} - u_{n,2} =& h^2 f_{n,1}. &\text{(3d)}
\end{aligned}
$$

By the boundary condition, it holds that

$$
\begin{aligned}
u_{1,0} =& u_{2,0} = \cdots = u_{n,0} = 0, &\text{(4a)}\\
u_{0,1} =& u_{n+1,1} = 0. &\text{(4b)}
\end{aligned}
$$

Substituting (4) into (3), we get

$$
\begin{aligned}
4u_{1,1} - u_{2,1} - u_{1,2} &= h^2 f_{1,1}, &\text{(5a)}\\
-u_{1,1} + 4u_{2,1} - u_{3,1} - u_{2,2} &= h^2 f_{2,1}, &\text{(5b)}\\
&\vdots\\
-u_{n-2,1} + 4u_{n-1,1} - u_{n,1} - u_{n-1,2} &= h^2 f_{n-1,1}, &\text{(5c)}\\
-u_{n-1,1} + 4u_{n,1} - u_{n,2} &= h^2 f_{n,1}. &\text{(5d)}
\end{aligned}
$$

Let, for $j = 1, \ldots, n$,

$$u_{:,j} = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix}, f_{:,j} = \begin{bmatrix} f_{1,j} \\ f_{2,j} \\ \vdots \\ f_{n,j} \end{bmatrix}, A_1 = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

Then (5) can be rewritten as following matrix form:

$$\begin{bmatrix} A_1 & -I_n \end{bmatrix} \begin{bmatrix} u_{:,1} \\ u_{:,2} \end{bmatrix} = h^2 f_{:,1}.$$

For $j = 2, \ldots, n-1$, using $u_{0,j} = u_{n+1,j} = 0$, we have

$$
\begin{aligned}
-u_{1,j-1} + 4u_{1,j} - u_{2,j} - u_{1,j+1} &= h^2 f_{1,j},\\
-u_{2,j-1} - u_{1,j} + 4u_{2,j} - u_{3,j} - u_{2,j+1} &= h^2 f_{2,j},\\
&\vdots\\
-u_{n-1,j-1} - u_{n-2,j} + 4u_{n-1,j} - u_{n,j} - u_{n-1,j+1} &= h^2 f_{n-1,j},\\
-u_{n,j-1} - u_{n-1,j} + 4u_{n,j} - u_{n,j+1} &= h^2 f_{n,j}.
\end{aligned}
$$

Above equations can be represented as following matrix form:

$$\begin{bmatrix} -I_n & A_1 & -I_n \end{bmatrix} \begin{bmatrix} u_{:,j-1} \\ u_{:,j} \\ u_{:,j+1} \end{bmatrix} = h^2 f_{:,j}.$$

For $j = n$, using $u_{1,n+1} = u_{2,n+1} = u_{n,n+1} = 0$, we have

$$\begin{aligned} -u_{1,n-1} + 4u_{1,n} - u_{2,n} &= h^2 f_{1,n}, \\ -u_{2,n-1} - u_{1,n} + 4u_{2,n} - u_{3,n} &= h^2 f_{2,n}, \\ &\vdots \\ -u_{n-1,n-1} - u_{n-2,n} + 4u_{n-1,n} - u_{n,n} &= h^2 f_{n-1,n}, \\ -u_{n,n-1} - u_{n-1,n} + 4u_{n,n} &= h^2 f_{n,n}. \end{aligned}$$

Above equations can be represented as following matrix form:

$$\begin{bmatrix} -I_n & A_1 \end{bmatrix} \begin{bmatrix} u_{:,n-1} \\ u_{:,n} \end{bmatrix} = h^2 f_{:,n}.$$

Therefore, (2) with boundary conditions is equivalent to a linear system

$$Au = h^2 f \tag{6}$$

with

$$A = \begin{bmatrix} A_1 & -I_n & & \\ -I_n & A_1 & \ddots & \\ & \ddots & \ddots & -I_n \\ & & -I_n & A_1 \end{bmatrix} \in \mathbb{R}^{n^2 \times n^2}, \tag{7}$$

and

$$A_1 = \begin{bmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{bmatrix}, \ u = \begin{bmatrix} u_{:,1} \\ u_{:,2} \\ \vdots \\ u_{:,n} \end{bmatrix}, \ f = \begin{bmatrix} f_{:,1} \\ f_{:,2} \\ \vdots \\ f_{:,n} \end{bmatrix}.$$

2 Project for direct method

(a) Use Algorithms 1, 2 and 3 (Gaussian elimination) to reduce A in (7) to an upper triangular matrix and modify the entries of b accordingly. Compare and plot the CPU times for reducing A to upper triangular with various n by using these three algorithms. (Use “tic” and “toc” functions in MATLAB to estimate the CPU times.)

Require: Nonsingular matrix A and right hand side vector b.
Ensure: This algorithm implements the Gaussian elimination procedure to reduce A to upper triangular and modify the entries of b accordingly.

for $k = 1, \ldots, n-1$ **do**
  Let p be the smallest integer with $k \leq p \leq n$ and $a_{pk} \neq 0$.
  If $\nexists\ p$, then stop.
  If $p \neq k$, then perform $(E_p) \leftrightarrow (E_k)$.
  for $i = k+1, \ldots, n$ **do**
    Compute $t = A(i,k)/A(k,k)$;
    Set $A(i,k) = 0$;
    Update $b(i) = b(i) - t \times b(k)$;
    for $j = k+1, \ldots, n$ **do**
      Update $A(i,j) = A(i,j) - t \times A(k,j)$;
    end for
  end for
end for

Algorithm 1: Gaussian elimination

Require: Nonsingular matrix A and right hand side vector b.
Ensure: This algorithm implements the Gaussian elimination procedure to reduce A to upper triangular and modify the entries of b accordingly.

for $k = 1, \ldots, n-1$ **do**
  Let p be the smallest integer with $k \leq p \leq n$ and $a_{pk} \neq 0$.
  If $\nexists\ p$, then stop.
  If $p \neq k$, then perform $(E_p) \leftrightarrow (E_k)$.
  for $i = k+1, \ldots, n$ **do**
    Compute $t = A(i,k)/A(k,k)$;
    Set $A(i,k) = 0$;
    Update $b(i) = b(i) - t \times b(k)$;
    Update $A(i, k+1:n) = A(i, k+1:n) - t \times A(k, k+1:n)$;
  end for
end for

Algorithm 2: Vector version of Gaussian elimination

Require: Nonsingular matrix A and right hand side vector b.
Ensure: This algorithm implements the Gaussian elimination procedure to reduce A to upper triangular and modify the entries of b accordingly.

for $k = 1, \ldots, n-1$ **do**
  Let p be the smallest integer with $k \le p \le n$ and $a_{pk} \neq 0$.
  If $\nexists\, p$, then stop.
  If $p \neq k$, then perform $(E_p) \leftrightarrow (E_k)$.
  Compute $t = A(k+1:n, k)/A(k,k)$;
  Set $A(k+1:n, k) = 0$;
  Update $A(k+1:n, k+1:n) = A(k+1:n, k+1:n) - t \times A(k, k+1:n)$;
  Update $b(k+1:n) = b(k+1:n) - b(k) \times t$.
end for

Algorithm 3: Matrix version of Gaussian elimination

(b) Use backward substitution to solve the upper triangular linear system in (a). Plot the CPU times for solving such linear system with various n.

(c) Compare the CPU times for using left matrix divide "$A \setminus b$" in MATLAB with that in (a) and (b).

(d) Store the matrix A with sparse format. Plot the CPU times for generating matrix A and solving the associated linear systems by left matrix divide "$A \setminus b$" with various n.

3 Project for iterative method

(e) Use Jacobi method to solve linear system (6).
Given an initial vector $x^{(0)}$, rewrite the linear system as:

$$\begin{aligned}
a_{11}x_1^{(k)} + a_{12}x_2^{(k-1)} + a_{13}x_3^{(k-1)} + \cdots + a_{1n}x_n^{(k-1)} &= b_1 \\
a_{21}x_1^{(k-1)} + a_{22}x_2^{(k)} + a_{23}x_3^{(k-1)} + \cdots + a_{2n}x_n^{(k-1)} &= b_2 \\
&\vdots \\
a_{n1}x_1^{(k-1)} + a_{n2}x_2^{(k-1)} + a_{n3}x_3^{(k-1)} + \cdots + a_{nn}x_n^{(k)} &= b_n.
\end{aligned}$$

If we decompose the coefficient matrix A as

$$A = L + D + U,$$

where D is the diagonal part, L is the strictly lower triangular part, and U is the strictly upper triangular part, of A, then we derive the iterative formulation for Jacobi method:

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

- Use Algorithm 4 with initial vector $x^{(0)} = [1, \cdots, 1]^\top$ to solve linear system (6). Plot the CPU times and iteration numbers k for solving such linear system with various n.

Require: Given $x^{(0)}$, tolerance TOL, maximum number of iteration M.
Ensure: The solution x.

Set $k = 1$.
Compute $x = -D^{-1}(L+U)x^{(0)} + D^{-1}b$.
while $k \le M$ and $\|x - x^{(0)}\|_2 \ge TOL$ **do**
  Set $k = k+1$, $x^{(0)} = x$;
  Compute $x = -D^{-1}(L+U)x^{(0)} + D^{-1}b$;
end while

Algorithm 4: Jacobi method

(f) Use Gauss-Seidel method to solve linear system (6).
Given an initial vector $x^{(0)}$, rewrite the linear system as:

$$\begin{array}{lcl}
\color{red}{a_{11}x_1^{(k)}} + a_{12}x_2^{(k-1)} + a_{13}x_3^{(k-1)} + \cdots + a_{1n}x_n^{(k-1)} & = & b_1 \\
\color{blue}{a_{21}x_1^{(k)}} + \color{red}{a_{22}x_2^{(k)}} + a_{23}x_3^{(k-1)} + \cdots + a_{2n}x_n^{(k-1)} & = & b_2 \\
\color{blue}{a_{31}x_1^{(k)}} + \color{blue}{a_{32}x_2^{(k)}} + \color{red}{a_{33}x_3^{(k)}} + \cdots + a_{3n}x_n^{(k-1)} & = & b_3 \\
 & \vdots & \\
\color{blue}{a_{n1}x_1^{(k)}} + \color{blue}{a_{n2}x_2^{(k)}} + \color{blue}{a_{n3}x_3^{(k)}} + \cdots + \color{red}{a_{nn}x_n^{(k)}} & = & b_n.
\end{array}$$

This improvement induce the Gauss-Seidel method. The iteration of the Gauss-Seidel method is defined as follows:

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

Require: Given $x^{(0)}$, tolerance TOL, maximum number of iteration M.
Ensure: The solution x.

Set $k = 1$.
Compute $x = -(D+L)^{-1}Ux^{(0)} + (D+L)^{-1}b$.
while $k \le M$ and $\|x - x^{(0)}\|_2 \ge TOL$ **do**
  Set $k = k+1$, $x^{(0)} = x$;
  Compute $x = -(D+L)^{-1}Ux^{(0)} + (D+L)^{-1}b$;
end while

Algorithm 5: Gauss-Seidel method

1. Use MATLAB functions “triu(A,1)” and “tril(A,-1)” to extract the strictly upper and lower triangular parts of A, respectively.
2. Use Algorithm 5 with initial vector $x^{(0)} = [1, \cdots, 1]^\top$ to solve linear system (6). Plot the CPU times and iteration numbers k for solving such linear system with various n.
3. Compare the results produced by Jacobi and Gauss-Seidel methods.

(g) Use SSOR method to solve linear system (6).

Given an initial vector $x^{(0)}$, rewrite the linear system as:

$$\begin{array}{lll}
a_{11}x_1^{(k)} + a_{12}x_2^{(k-1)} + a_{13}x_3^{(k-1)} + \cdots + a_{1n}x_n^{(k-1)} & = & b_1 \\
a_{21}x_1^{(k)} + a_{22}x_2^{(k)} + a_{23}x_3^{(k-1)} + \cdots + a_{2n}x_n^{(k-1)} & = & b_2 \\
a_{31}x_1^{(k)} + a_{32}x_2^{(k)} + a_{33}x_3^{(k)} + \cdots + a_{3n}x_n^{(k-1)} & = & b_3 \\
 & \vdots & \\
a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + a_{n3}x_3^{(k)} + \cdots + a_{nn}x_n^{(k)} & = & b_n.
\end{array}$$

Let the approximate solution $\mathbf{x}^{(k,i)}$ produced by Gauss-Seidel method be defined by

$$\mathbf{x}^{(k,i)} = \left[x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)}\right]^T$$

and

$$r_i^{(k)} = \left[r_{1i}^{(k)}, r_{2i}^{(k)}, \ldots, r_{ni}^{(k)}\right]^T = b - A\mathbf{x}^{(k,i)}$$

be the corresponding residual vector. Then the ith component of $r_i^{(k)}$ is

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

so

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

Consequently, the Gauss-Seidel method can be characterized as choosing $x_i^{(k)}$ to satisfy

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}}.$$

Relaxation method is modified the Gauss-Seidel procedure to

$$\begin{aligned}
x_i^{(k)} &= x_i^{(k-1)} + \omega\frac{r_{ii}^{(k)}}{a_{ii}} \\
&= x_i^{(k-1)} + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)} - a_{ii}x_i^{(k-1)}\right] \\
&= (1-\omega)x_i^{(k-1)} + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right] \qquad (8)
\end{aligned}$$

for certain choices of positive ω. These methods are called for

$\omega < 1$: under relaxation,
$\omega = 1$: Gauss-Seidel method,
$\omega > 1$: over relaxation.

Over-relaxation methods are called SOR (Successive over-relaxation). To determine the matrix of the SOR method, we rewrite (8) as

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

so that if $A = L + D + U$, then we have

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

Theorem 1 (Ostrowski-Reich) *If A is positive definite and the relaxation parameter ω satisfying $0 < \omega < 2$, then the SOR iteration converges for any initial vector $x^{(0)}$.*

Let A be symmetric and $A = D + L + L^T$. The idea is in fact to implement the SOR formulation twice, one forward and one backward, at each iteration. That is, SSOR method defines

$$\begin{aligned} (D + \omega L)x^{(k-\frac{1}{2})} &= \left[(1-\omega)D - \omega L^T\right] x^{(k-1)} + \omega b, & (9) \\ (D + \omega L^T)x^{(k)} &= \left[(1-\omega)D - \omega L\right] x^{(k-\frac{1}{2})} + \omega b. & (10) \end{aligned}$$

Define

$$\begin{cases} M_\omega : = D + \omega L, \\ N_\omega : = (1-\omega)D - \omega L^T. \end{cases}$$

Then from the iterations (9) and (10), it follows that

$$\begin{aligned} x^{(k)} &= \left(M_\omega^{-T} N_\omega^T M_\omega^{-1} N_\omega\right) x^{(k-1)} + \omega \left(M_\omega^{-T} N_\omega^T M_\omega^{-1} + M_\omega^{-T}\right) b \\ &\equiv T(\omega)x^{(k-1)} + M(\omega)^{-1} b, \end{aligned}$$

where

$$M(\omega) = \frac{1}{\omega(2-\omega)}(D + \omega L)D^{-1}\left(D + \omega L^T\right).$$

1. Take $x^{(0)} = [1, \cdots, 1]^\top$ as an initial vector.
2. Use MATLAB functions "triu(A,1)" and "tril(A,-1)" to extract the strictly upper and lower triangular parts of A, respectively.
3. Fixed $n = 100$ and uniformly took 40 values for the parameter ω in the interval $(0, 2)$, show the iteration numbers and CPU times of SSOR iterative method for each ω. Find the optimal value ω^* of the parameter ω.

4. Compare the iteration numbers and CPU times for Jacobi, Gauss-Seidel and SSOR(ω^*) iterative methods with various n.

(h) Use conjugate gradients method to solve linear system (6).

1. Use MATLAB function `pcg` without any preconditioner:

   [x, flag, relres, iter] = pcg(A, b, tol, maxit)

2. Use MATLAB function `pcg` with a given preconditioner:

   [x, flag, relres, iter] = pcg(A, b, tol, maxit, M),
   [x, flag, relres, iter] = pcg(A, b, tol, maxit, M1, M2),
   [x, flag, relres, iter] = pcg(A, b, tol, maxit, [], M2),
   [x, flag, relres, iter] = pcg(A, b, tol, maxit, MFUN).

   (i) **Jacobi method:** $A = D + (L + U), \quad M = D$

   $$x_{k+1} = -D^{-1}(L + U)x_k + D^{-1}b$$

   (ii) **Gauss-Seidel:** $A = (D + L) + U, \quad M = D + L$

   $$x_{k+1} = -(D + L)^{-1}Ux_k + (D + L)^{-1}b.$$

   (iii) **SSOR:** $A = D + L + L^T, \quad M = M(\omega)$

   $$x^{(k)} = \left(M_\omega^{-T} N_\omega^T M_\omega^{-1} N_\omega\right) x^{(k-1)} + M(\omega)^{-1}b,$$

   where

   $$M(\omega) = \frac{1}{\omega(2-\omega)}(D + \omega L)D^{-1}\left(D + \omega L^T\right).$$

   (iv) M may be a function handle MFUN returning $M^{-1}x$

   [x, flag, relres, iter] = pcg(A, b, tol, maxit, ...
   @(x)precSSOR(x,omega,mtxLower,mtxdiag)

3. Compare the iteration numbers and CPU times for `pcg` by using different preconditioner with various n.