MA/CS 615 Spring 2022 Homework #5

Due on April 19th. Late homework will not be given any credit. Collaboration is OK but not encouraged. Indicate on your report whether you have collaborated with others and whom you have collaborated with.

1. (60 points) *General iterative methods and conjugate gradient method.* In MATLAB, we can use the functions *numgrid* and *delsq* to generate the classical second order discrete Laplacian with homogeneous Dirichlet b.c. on a few more general domains:

```
Length = 10;      % size of geometry

% 'numgrid' generates uniform grid points for the interior
%  of the specified shape.
% other possible shapes include L,S,N,C,D,A,H,B
Domain_grid = numgrid('L',Length); % L shape domain

% 'delsq' generate the second order discrete Laplacian
%  defined on the grid points
A = delsq(Domain_grid);
n = size(A,1);
b = rand(n,1);

figure;
% visualize the domain and grid points
subplot(1,2,1), spy(Domain_grid), title('domain shape')
% visualize the sparsity pattern of the matrix A
subplot(1,2,2), spy(A), title('system matrix')

%% Generate a linear system Ax=b
n = size(A,1);
b = rand(n,1);
x_true = A\b;
```

For example, if we specify the domain as the square, then the obtained matrix above is exactly $K \otimes Id + Id \otimes K$. Use the script above to generate a linear system $Ax = b$ where A is $n \times n$ and b is generated randomly. You can use whatever shape you like and use a proper length such that n is at least around one thousand. You can verify in MATLAB that the matrix A above is indeed symmetric and positive definite:

```
norm(A-A','fro') % If A is real symmetric, then this should be zero
min(eig(A))      % The smallest eigenvalue should be positive
```

The following can be used to compute the decomposition $A = D - L - U$:

```
D=diag(A);
L=-tril(A)+diag(D);
U=diag(D)-L-A;
```

Implement the following methods:

- Jacobi iteration: $Dx^{k+1} = (D - A)x^k + b$.
- Weighted Jacobi iteration: $D/wx^{k+1} = (D/w - A)x^k + b$, with $w = \frac{2}{3}$.
- Gauss-Seidel iteration: $(D - L)x^{k+1} = Ux^k + b$.
- Successive overrelaxation (SOR):

$$\frac{1}{w}(D - wL)x^{k+1} = \frac{1}{w}[(1 - w)D + wU]x^k + b,$$

  where w can be chose as a number close to 2. You can try different w, e.g., use $w = 1.9$.
- Steepest Descent for minimizing the cost function $f(x) = \frac{1}{2}x^T Ax - x^T b$.
- Conjuage Gradient (CG) method for solving $Ax = b$.
- Preconditioned Conjuage Gradient (PCG): use CG to solve $PAP^T y = Pb$. Use the incomplete Cholesky factorization as the preconditioner, i.e., if $A \approx \tilde{L}\tilde{L}^T$ then set $P = \tilde{L}^{-1}$. Use the function *ichol* for the incomplete Cholesky factorization. You can check how good the approximation $\tilde{L}\tilde{L}^T$ is and the sparsity pattern of $\tilde{L}$:

```
Lt=ichol(A);
spy(Lt);
norm(A-Lt*Lt','fro')/norm(A)
```

Store the following quantities for each iteration:

1. Relative error $\|x^k - x\|/\|x\|$.
2. Relative residue $\|b - Ax^k\|/\|b\|$.
3. For CG and Steepest Descent only: cost function $f(x_k)$ where $f(x) = \frac{1}{2}x^T Ax - x^T b$.

Use zero initial guesss thus the initial relative error and the initial relative residue are both one for each method. For CG and PCG, stop the iteration if the relative residue is smaller than some parameter, e.g., 10^{-14} or 10^{-15}. For all other methods, set the iteration number as $n/2$ where n is the size of the vector b.

Explain what parameter you have used for SOR and the size of your matrix A. Show the following four figures:

1. One figure of the domain grid points.
2. One figure of semilogy plot of relative residue for seven methods. Add legend in the figure by the following

```
legend('Steepest Descent','Jacobi','Weighted ...
   Jacobi','Gauss-Seidel','SOR','CG','PCG');
```

3. One figure of semilogy plot of relative error for seven methods.

4. One figure of the loglog plot of the cost function for CG and SD only. Notice that in CG and SD, the cost function is supposed to decrease as iteration number increases, while the error and residue are not necessarily monotonically descreasing. bf If your cost function has a minimum value $f(x^*) < 0$ with x^* being minimizer, then simply plot the modified cost function value $\bar{f}(x) = f(x) - f(x^*)$.

2. (40 points) For a rectangular domain Ω, consider a 2D variable coefficient problem

$$-\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega$$

with homogeneous Dirichlet boundary condition, where $a(\mathbf{x}) > 0$ is a scalar coefficient. Consider a uniform rectangular mesh and using Q^1 finite element method with trapezoidal quadrature for both x and y variables. The finite element method is to seek $u_h \in V_0^h$ satisfying

$$\mathcal{A}_h(u_h, v_h) = \langle f, v_h \rangle.$$

Using notation in Chapter 2, the scheme can be written as

$$\left[\frac{1}{\Delta x^2}(D_x^T \otimes I_y)A_1(D_x \otimes I_y) + \frac{1}{\Delta y^2}(I_x \otimes D_y^T)A_2(I_x \otimes D_y)\right] vec(U) = vec(F), \qquad (0.1)$$

where A_1 and A_2 are two diagonal matrices defined as follows.

Let a_1 be a 2D array of size $Ny \times (Nx+1)$ satisfying $a_2(j,i) = \frac{1}{2}a(x_i, y_j) + \frac{1}{2}a(x_{i-1}, y_j)$ and a_2 be a 2D array of size $(Ny+1) \times Nx$ satisfying $a_1(j,i) = \frac{1}{2}a(x_i, y_j) + \frac{1}{2}a(x_i, y_{j-1})$. Then A_1 and A_2 can be easily generated in MATLAB as sparse diagonal matrices:

```
A1=sparse(diag(a1(:)));
A2=sparse(diag(a2(:)));
```

Notice that (0.1) is equivalent to

$$\frac{1}{\Delta x^2}((UD_x^T).*a_1)D_x + \frac{1}{\Delta y^2}D_y^T(a_2.*(D_yU)) = F. \qquad (0.2)$$

Solve the linear system (0.1) by the Preconditioned Conjuage Gradient Method using the constant coefficient Laplacian (see the code *Poisson2D_Dirichlet.m*) as the preconditioner. Use eigenvector method to invert the constant coefficient Laplacian.

Let the linear system in (0.1) be $Ax = b$, when computing a matrix vector multiplication Ap for a matrix A and a vector p, you will have to use the compact form (0.2) for large grids. For example, you can do it as following:

```
% To compute a matrix-vector multiplication Ap = A*p  where A is:
%  A=1/dx^2*kron(Dx', Iy)*sparse(diag(A1(:)))*kron(Dx, Iy)+1/dy^2*kron(Ix, ...
   Dy')*sparse(diag(A2(:)))*kron(Ix, Dy);

   P=reshape(p, Ny, Nx);  % converting p to a 2D array P
   AP=((P*Dx').*A1)*Dx/dx^2+Dy'*((Dy*P).*A2)/dy^2;
   Ap=AP(:); % converting an array to a vector
```

Run your PCG for at most 20 iterations, then plot the semilogy of relative residue v.s. iteration numbers for two grids $1024 * 1024$ and $2048 * 2048$ for two different setup:

1.
```
% a uniform N+2 by N+2 grid for the domain (0,1)*(0,1)
% xi and yi is the N by N interior grid
dx=1/(N+1); x=[0:dx:1]'; xi=x(2:end-1);

dy=dx; y=x; yi=xi;

% a smooth coefficient
a=@(x,y) cos(y)*cos(x');
f=@(x,y) 2*pi*cos(y).*sin(4*pi*y)*(sin(x').*cos(2*pi*x')...
         +2*pi*cos(x').*sin(2*pi*x'))+...
          4*pi*(sin(y).*cos(4*pi*y)...
         +4*pi*cos(y).*sin(4*pi*y))*(cos(x').*sin(2*pi*x'));
```

2.
```
% a uniform N+2 by N+2 grid for the domain (0,1)*(0,1)
% xi and yi is the N by N interior grid
dx=1/(N+1); x=[0:dx:1]'; xi=x(2:end-1);

dy=dx; y=x; yi=xi;

% a nonsmooth coefficient
a=@(x,y) 10000*ones(size(y))*(x'>0.5)+ones(size(y))*ones(size(x'));
f=@(x,y) 2*pi*cos(y).*sin(4*pi*y)*(sin(x').*cos(2*pi*x')...
         +2*pi*cos(x').*sin(2*pi*x'))+...
          4*pi*(sin(y).*cos(4*pi*y)...
         +4*pi*cos(y).*sin(4*pi*y))*(cos(x').*sin(2*pi*x'));
```