INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

APPLIED COMPUTATIONAL METHODS LABORATORY

LAB – 6 Report

Rajiv Harlalka

20MA20073

Department of Mathematics

```matlab
% Rajiv Harlalka 20MA20073 9th March
% Program Poisson2D_SOR.m
% This is the main program for the solution of
% Poisson's equation
% -a△u = f with Dirichlet b.c. u = 0
% in 2D using iterative SOR method.

close all
clc
clear
clf
%Define input parameters
n=20; % number of inner nodes in one direction.
a_amp = 12; % amplitude for the function a(x_1,x_2)
f_amp = 1; % we can choose f=1, 50, 100
x_0=0.5;
y_0=0.5;
c_x=1;
c_y=1;
h = 1/(n+1); % define step length
% ----------------------------------------
% Computing all matrices and vectors
% ----------------------------------------
% Generate a n*n by n*n stiffness matrix
S = Discrete_Poisson_2D(n);
%% generate coefficient matrix of a((x_1)_i,(x_2)_j) = a(i*h,j*h)
C = zeros(n,n);
for i=1:n
for j=1:n
C(i,j) = 1 + a_amp*exp(-((i*h-x_0)^2/(2*c_x^2)...
+(j*h-y_0)^2/(2*c_y^2)));
end
end
% create diagonal matrix from C
D = zeros(n^2,n^2);
for i=1:n
for j=1:n
D(j+n*(i-1),j+n*(i-1)) = C(i,j);
end
end
% If f is constant.
% f = f_amp*ones(n^2,1);
% If f is Gaussian function.
f=zeros(n^2,1);
for i=1:n
for j=1:n
f(n*(i-1)+j)=f_amp*exp(-((i*h-x_0)^2/(2*c_x^2)...
+(j*h-y_0)^2/(2*c_y^2)));
end
end
% Compute vector of right hand side
% b = D^(-1)*f computed as b(i,j)=f(i,j)/a(i,j)
b=zeros(n^2,1);
for i=1:n
for j=1:n
b(n*(i-1)+j)=f(n*(i-1)+j)/C(i,j); % Use coefficient matrix C or
% diagonal matrix D to get a(i,j)
end
end
% ----------------------------------------
% Solution of 1/h^2 S u = b using SOR method
% with red-black ordering, version I
% ----------------------------------------
err = 1; k=0; sch = 0; tol=10^(-9);
V = zeros(n,n);
V_old = zeros(n,n);
F=vec2mat(b,n)';
X=diag(ones(1,n-1),-1);
X=X+X';
%arrange red-black indexing
blackindex = invhilb(n) < 0;
redindex = fliplr(blackindex);
B=V;
V(redindex)=0;
R=V;
V(blackindex)=0;
redF = F; redF(blackindex)=0;
blackF = F; blackF(redindex)=0;
% extract matrices L and U for matrix RSOR
L=tril(S,-1);
U=L';
Dinv=diag(diag(S).^(-1));
L = Dinv*(-L);
U = Dinv*(-U);
D=diag(ones(1,n*n));
omegas = 1.05:0.05:1.95;
for omega = omegas
k=0;
err =1;
B=V;
V(redindex)=0;
R=V;
V(blackindex)=0;
```

```matlab
% counter for omega
sch = sch+1;
while(err>tol)
R = (1 - omega)*R + omega*(X*B + B*X + h^2*redF)/4;
B = (1- omega)*B + omega*(X*R + R*X + h^2*blackF)/4;
k=k+1;
V_new =R+B;
err = norm(V_new - V_old);
V_old = V_new;
end
% the matrix RSOR in the method SOR: x_m+1 = RSOR*x_m + c_SOR
RSOR = inv(D - omega*L)*((1-omega)*D + omega*U);
lambda = max(abs(eig(RSOR)));
mu = (lambda + omega -1)/(sqrt(lambda)*omega);
disp('-- Relaxation parameter in SOR method ----------')
omega;
disp('-- Computed optimal relaxation parameter ----------')
omega_opt = 2/(1 + sqrt(1 - mu^2))
if (omega <= 2.0 && omega >=omega_opt )
disp('-- omega_opt < omega < 2.0 ----------')
radius = omega -1;
elseif(omega <= omega_opt && omega > 0)
disp('-- omega < omega_opt ----------')
omega_tail = -omega +0.5*omega^2*mu^2; ...
+ omega*mu*sqrt(1 - omega + 0.25*omega^2*mu^2)
radius = 1 + omega_tail;
end
disp('-- Number of iterations in SOR method ----------')
k;
iterations(sch) = k;
spectral_radius(sch)= radius;
omega_optimal(sch) = omega_opt;
end
% apply zero boundary conditions
V_new = [zeros(1,n+2); zeros(n,1) V_new zeros(n,1);zeros(1,n+2)];
% ----------------------------------------
% Plots and figures for SOR method, version I
% ----------------------------------------
figure(1)
%% plotting
x1=0:h:1;
y1=0:h:1;
subplot (2,2,1)
surf(x1,y1,V_new) % same plot as above, (x1, y1 are vectors)
view(2)
colorbar
xlabel('x_1')
ylabel('x_2')
zlabel('u(x_1,x_2)')
title( ['solution u(x_1,x_2) in SOR method ',...
', N = ',num2str(n),', iter. = ',num2str(k)])
subplot (2,2,2)
surf(x1,y1,V_new) % same plot as above
colorbar
xlabel('x_1')
ylabel('x_2')
zlabel('u(x_1,x_2)')
title( ['solution u(x_1,x_2) in SOR method',...
', N = ',num2str(n),', iter. = ',num2str(k)])
% Plotting a(x,y)
Z_a= zeros(n+2);
for i=1:(n+2)
for j=1:(n+2)
Z_a(i,j)= 1 + a_amp*exp(-((i*h-x_0)^2/(2*c_x^2)...
+(j*h-y_0)^2/(2*c_y^2)));
end
end
subplot (2,2,3)
surf(x1,y1,Z_a)
xlabel('x_1')
ylabel('x_2')
zlabel('a(x_1,x_2)')
title( ['coefficient a(x_1,x_2) with A = ',num2str(a_amp)])
% plot the function f(x,y)
Z_f= zeros(n+2);
for i=1:(n+2)
for j=1:(n+2)
Z_f(i,j)=f_amp*exp(-((x1(i)-x_0)^2/(2*c_x^2)...
+(y1(j)-y_0)^2/(2*c_y^2)));
end
end
subplot (2,2,4)
surf(x1,y1,Z_f)
xlabel('x_1')
ylabel('x_2')
zlabel('f(x_1,x_2)')
title( ['f(x_1,x_2) with A_f = ',num2str(f_amp)])
% plot convergence of SOR depending on omega
figure(2)
plot(omegas, iterations,'b o-', 'LineWidth',2)
hold on
plot(omega_optimal, iterations,'r o ', 'LineWidth',2)
xlabel('Relaxation parameter \omega')
ylabel('Number of iterations in SOR')
legend('SOR(\omega)','Computed optimal \omega')
title(['Mesh: ',num2str(n),' by ',num2str(n),' points'])
% plot convergence of SOR depending on omega
figure(3)
plot(omegas, spectral_radius,'b o-', 'LineWidth',2)
xlabel('Relaxation parameter \omega')
ylabel(' Spectral radius \rho(R_{SOR(\omega)})')
legend('\rho(R_{SOR(\omega)})')
title(['Mesh: ',num2str(n),' by ',num2str(n),' points'])
```

Output:

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.4864

-- Number of iterations in SOR method ----------

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

0.4813

-- Number of iterations in SOR method ----------

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

1.7406

-- omega < omega_opt ----------

ans =

0.4734

-- Number of iterations in SOR method ----------

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

```
    1.7406

-- omega < omega_opt ----------

ans =

    0.4626

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.4490

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
```

```
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.4323

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =
```

```
    0.4125

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.3894

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406
```

```
-- omega < omega_opt ----------

ans =

    0.3626

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.3317

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------
```

```
omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.2959

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.2540
```

```
-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------

ans =

    0.2032

-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7406

-- omega < omega_opt ----------
```

ans =

    0.1350

-- Number of iterations in SOR method ----------

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

    1.7500

-- omega_opt < omega < 2.0 ----------

-- Number of iterations in SOR method ----------

-- Relaxation parameter in SOR method ----------

-- Computed optimal relaxation parameter ----------

omega_opt =

    1.8000

-- omega_opt < omega < 2.0 ----------

-- Number of iterations in SOR method ----------

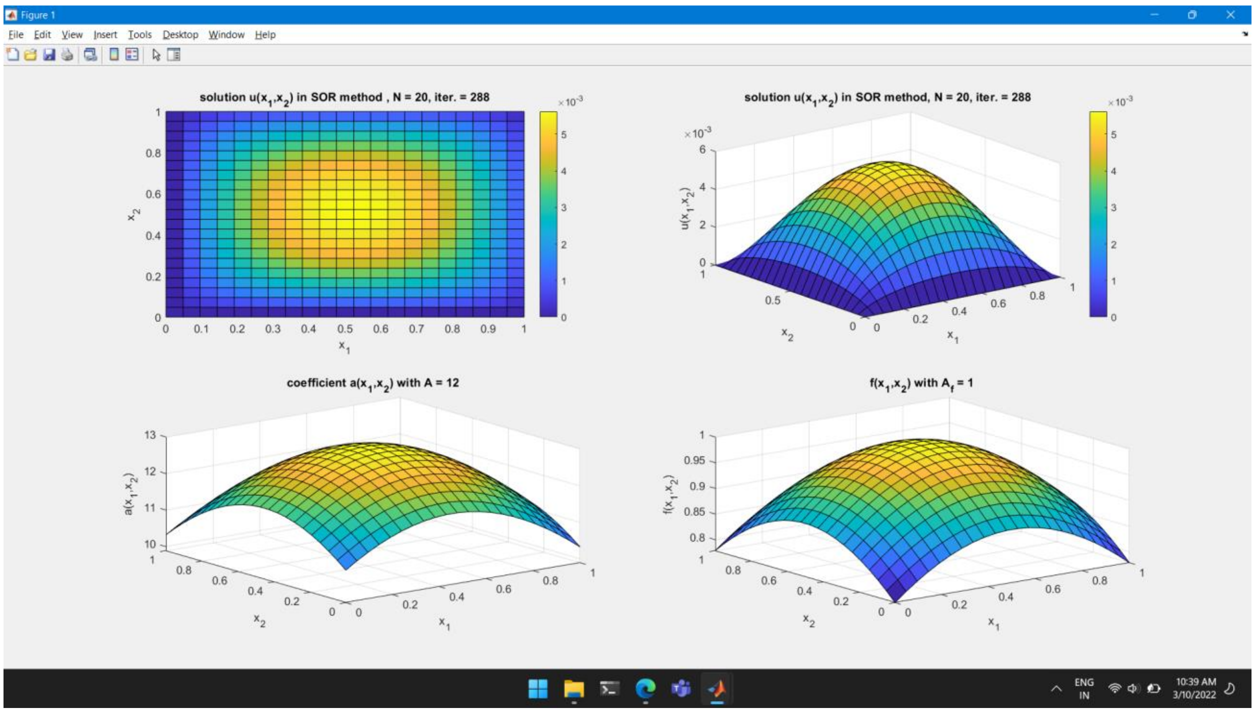
-- Relaxation parameter in SOR method ----------

```
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.8500

-- omega_opt < omega < 2.0 ----------
-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.9000

-- omega_opt < omega < 2.0 ----------
-- Number of iterations in SOR method ----------
-- Relaxation parameter in SOR method ----------
-- Computed optimal relaxation parameter ----------

omega_opt =

    1.9500
```

-- omega_opt < omega < 2.0 ----------

-- Number of iterations in SOR method ----------

Plot SOR Solution



Plot SOR Iterations and Spectral Radius

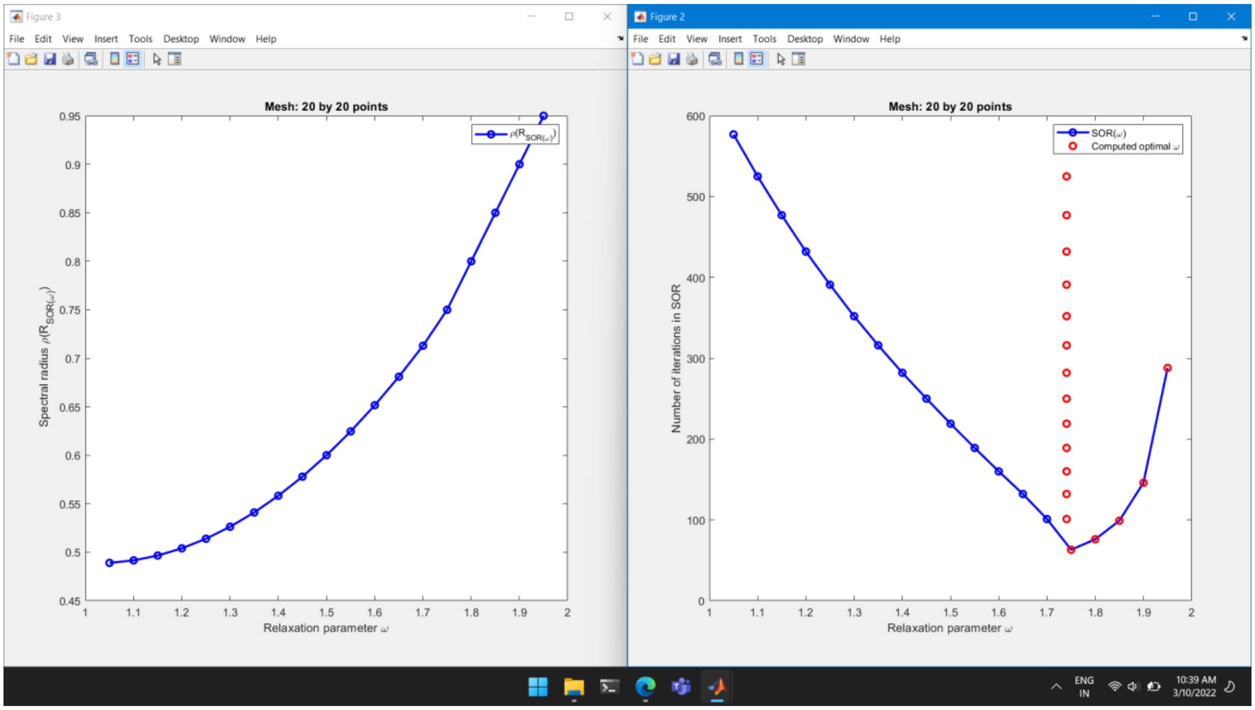


Figure 3
File Edit View Insert Tools Desktop Window Help
Mesh: 20 by 20 points
Spectral radius $\rho(R_{SOR(\omega)})$
Relaxation parameter ω
Figure 2
Mesh: 20 by 20 points
SOR(ω)
Computed optimal ω
Number of iterations in SOR
Relaxation parameter ω
ENG
IN
10:39 AM
3/10/2022