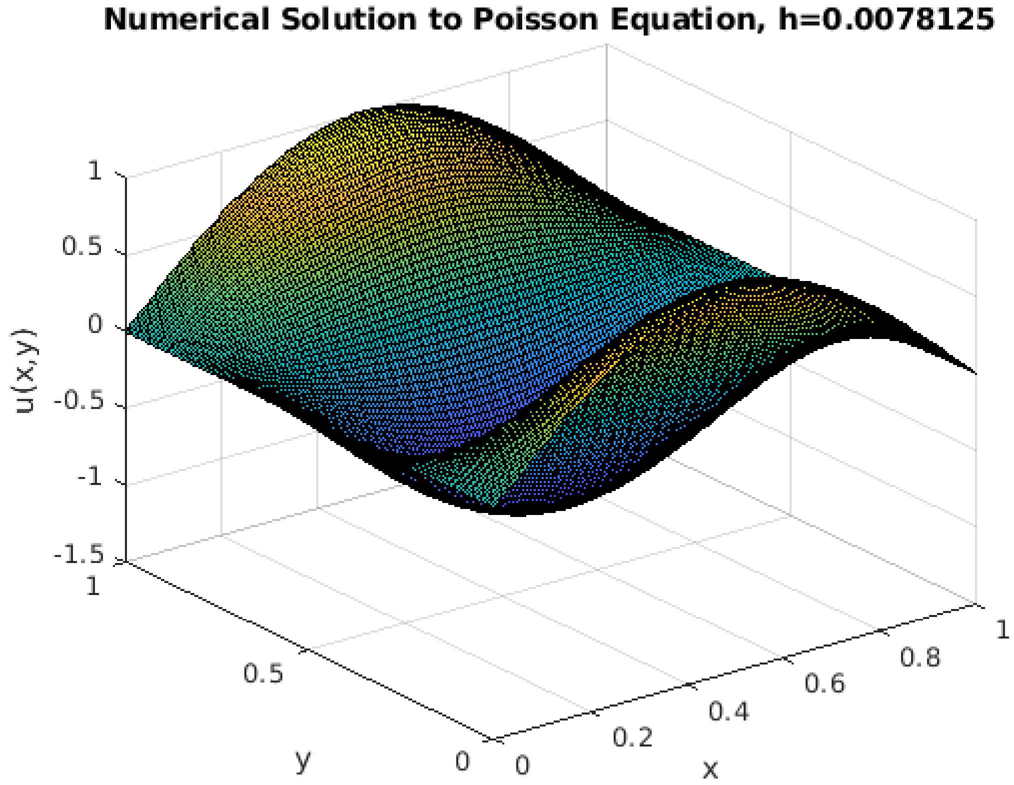
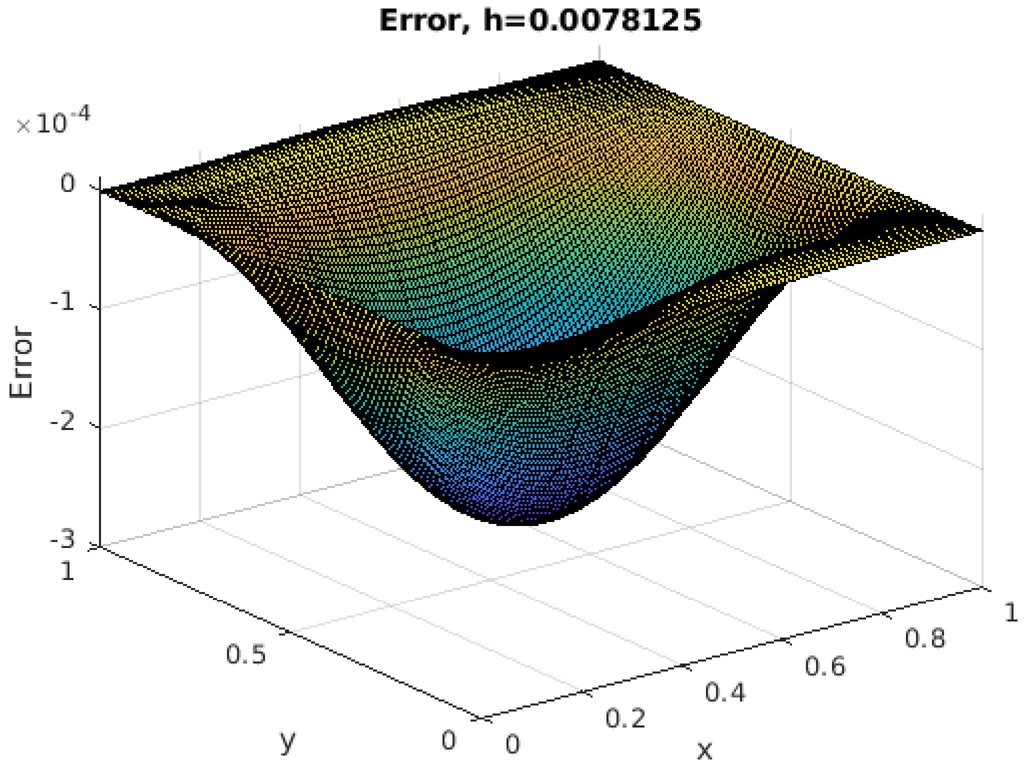
```
% Numerical approximation to Poisson's equation over the square [a,b]x[a,b] with
% Dirichlet boundary conditions.  Uses a uniform mesh with (n+2)x(n+2) total
% points (i.e, n interior grid points).
% Input:
%     ffun : the RHS of poisson equation (i.e. the Laplacian of u).
%     gfun : the boundary function representing the Dirichlet B.C.
%      a,b : the interval defining the square
%        m : m+2 is the number of points in either direction of the mesh.
% Ouput:
%        u : the numerical solution of Poisson equation at the mesh points.
%      x,y : the uniform mesh.

function [u,x,y] = fd2poissonsor(ffun,gfun,a,b,m,w)

h = (b-a)/(m+1); %mesh spacing

tol = 10^(-8);   %relative residual

maxiter = 1000;  %maximum value of k

[x,y] = meshgrid(a:h:b); %Uniform mesh, including boundary points.

idx = 2:m+1;
idy = 2:m+1;
dx = 1:m+2;
dy = 1:m+2;

u = zeros(m+2,m+2);

% Compute boundary terms, south, north, east, west
u(1,:)    = feval(gfun,x(1,:),y(1,:));        % Include corners
u(m+2, :) = feval(gfun,x(m+2,:),y(m+2,:)); % Include corners
u(idy,m+2)    = feval(gfun,x(idy,m+2),y(idy,m+2));        % No corners
u(idy,1)      = feval(gfun,x(idy,1),y(idy,1));              % No corners

% Evaluate the RHS of Poisson's equation at the interior points.
f = feval(ffun,x(dy,dx),y(dy,dx));

for k = 0:maxiter
    %Iterate
    for j = 2:(m+1)
        for i = 2:(m+1)
            u(i,j) = (1-w)*u(i,j)+(w/4)*(u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1)-(h^2)*f(i,j));
        end
    end

    %Compute the residual
    residual = zeros(m+2,m+2);

    for j = 2:(m+1)
        for i = 2:(m+1)
            residual(i,j) = -4*u(i,j)+(u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1)-(h^2)*f(i,j));
        end
    end

    %Determine if convergence has been reached
        if norm(residual(:),2)<tol*norm(f(:),2)
                break
    end
end

end
```

```
% USing fd2poissonsor function to solve the Poisson equation from the
% FD2-Poisson Handout.

m = (2^7) - 1;
a=0; b=1;
h = (b-a)/(m+1); %mesh spacing

w = 2/(1+sin(pi*h)); %optimal relaxation parameter

f = @(x,y) -5*pi^2*sin(pi*x).*cos(2*pi*y);
g = @(x,y) sin(pi*x).*cos(2*pi*y);

uexact = @(x,y) g(x,y);
% Laplacian(u) = f
% u = g on Boundary
% Exact solution is g.
% Compute and time the solution
tic
[u,x,y] = fd2poissonsor(f,g,a,b,m,w);
gedirect = toc;
fprintf('SOR take %d s\n',gedirect);

% Plot solution
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,u), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution to Poisson Equation, h=',num2str(h)));

% Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,u-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Error, h=',num2str(h)));
```

```
SOR take 1.730786e+00 s
```





Published with MATLAB® R2020a

```
% Numerical approximation to Poisson's equation over the square [a,b]x[a,b] with
% Dirichlet boundary conditions.  Uses a uniform mesh with (n+2)x(n+2) total
% points (i.e, n interior grid points).
% Input:
%     ffun : the RHS of poisson equation (i.e. the Laplacian of u).
%     gfun : the boundary function representing the Dirichlet B.C.
%      a,b : the interval defining the square
%        m : m+2 is the number of points in either direction of the mesh.
% Ouput:
%        u : the numerical solution of Poisson equation at the mesh points.
%      x,y : the uniform mesh.

function [u,x,y] = fd2poissonsp(ffun,gfun,a,b,m)

h = (b-a)/(m+1);   % Mesh spacing

[x,y] = meshgrid(a:h:b);   % Uniform mesh, including boundary points.

idx = 2:m+1;
idy = 2:m+1;

% Compute boundary terms, south, north, east, west
ubs = feval(gfun,x(1,1:m+2),y(1,1:m+2));     % Include corners
ubn = feval(gfun,x(m+2,1:m+2),y(m+2,1:m+2)); % Include corners
ube = feval(gfun,x(idy,m+2),y(idy,m+2));     % No corners
ubw = feval(gfun,x(idy,1),y(idy,1));         % No corners

% Evaluate the RHS of Poisson's equation at the interior points.
f = feval(ffun,x(idy,idx),y(idy,idx));

% Adjust f for boundary terms
f(:,1) = f(:,1) - ubw/h^2;             % West
f(:,m) = f(:,m) - ube/h^2;             % East
f(1,1:m) = f(1,1:m) - ubs(idx)/h^2;    % South
f(m,1:m) = f(m,1:m) - ubn(idx)/h^2;    % North

f = reshape(f,m*m,1);

%Using sparse matrix capabilities to form D2x and D2y matrices
I = eye(m);
e = ones(m,1);
e1 = zeros(m,1);
%D2x
T = spdiags([e1 -2*e1 e1],[-1 0 1],m,m);
S = spdiags([e e],[-1 1],m,m);
D2x = (1/h^2)*(kron(I, T) + kron(S,I));
%D2y
Ty = spdiags([e -2*e e],[-1 0 1],m,m);
Sy = spdiags([e1 e1],[-1 1],m,m);
D2y = (1/h^2)*(kron(I, Ty) + kron(Sy,I));

% Solve the system
u = (D2x + D2y)\f;

% Convert u from a column vector to a matrix to make it easier to work with
% for plotting.
u = reshape(u,m,m);

% Append on to u the boundary values from the Dirichlet condition.
u = [ubs;[ubw,u,ube];ubn];

end
```

Contents


```
% Script for testing fd2poisson over the square [a,b]x[a,b]
a = 0; b = 1;

% Laplacian(u) = f
f = @(x,y) 10*pi^2*(1+cos(4*pi*(x+2*y))-2*sin(2*pi*(x+2*y))).*exp(sin(2*pi*(x+2*y)));
% u = g on Boundary
g = @(x,y) exp(sin(2*pi*(x+2*y)));

% Exact solution is g.
uexact = @(x,y) g(x,y);

% Compute and time the solution
k1    = zeros(1,3);
h1    = zeros(1,3);
m1    = zeros(1,3);
t     = zeros(1,3);
t_sor = zeros(1,3);
t_sp  = zeros(1,3);
t_dst = zeros(1,3);
t_mg  = zeros(1,3);

t1   = [];
tsor = [];
tsp  = [];
tdst = [];
tmg  = [];
for ii = 1:3
    for k=4:6
        k1(k-3) = k;
        m1(k-3) = 2^k-1;
        m = 2^k-1;
        h1(k-3) = (b-a)/(m+1);
        h = (b-a)/(m+1);
        w = 2/(1+sin(pi*h)); %optimal relaxation parameter

        tic
        [u,x,y] = fd2poisson(f,g,a,b,m);
        gedirect = toc;
        t(k-3) = gedirect;

        tic
        [usor,x,y] = fd2poissonsor(f,g,a,b,m,w);
        gedirect = toc;
        t_sor(k-3) = gedirect;

        tic
        [usp,x,y] = fd2poissonsp(f,g,a,b,m);
        gedirect = toc;
        t_sp(k-3) = gedirect;

        tic
        [udst,x,y] = fd2poissondst(f,g,a,b,m);
        gedirect = toc;
        t_dst(k-3) = gedirect;

        tic
        [umg,x,y] = fd2poissonmg(f,g,a,b,m);
        gedirect = toc;
        t_mg(k-3) = gedirect;
    end

    t1   = [t1,t];
    tsor = [tsor,t_sor];
    tsp  = [tsp, t_sp];
    tdst = [tdst, t_dst];
    tmg  = [tmg,t_mg];
end

%k=4
c4=[t1(1);t1(4);t1(7)]';
d4=[tsor(1);tsor(4);tsor(7)]';
e4=[tsp(1);tsp(4);tsp(7)]';
fd4=[tdst(1);tdst(4);tdst(7)]';
h4=[tmg(1);tmg(4);tmg(7)]';

%k=5
c5=[t1(2);t1(5);t1(8)]';
d5=[tsor(2);tsor(5);tsor(8)]';
e5=[tsp(2);tsp(5);tsp(8)]';
fd5=[tdst(2);tdst(5);tdst(8)]';
h5=[tmg(2);tmg(5);tmg(8)]';

%k=6
c6=[t1(3);t1(6);t1(9)]';
d6=[tsor(3);tsor(6);tsor(9)]';
e6=[tsp(3);tsp(6);tsp(9)]';
```

```
fd6=[tdst(3);tdst(6);tdst(9)]';
h6=[tmg(3);tmg(6);tmg(9)]';

k4 = [k1(1);k1(1);k1(1)];
m4 = [m1(1);m1(1);m1(1)];
h4 = [h1(1);h1(1);h1(1)];
%Table showing timing results of each method and for each value of m.
Table4 = table(k4,m4,h4,c4(:),d4(:),e4(:),fd4(:),h4(:), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time_dst','time_mg'});

k5 = [k1(2);k1(2);k1(2)];
m5 = [m1(2);m1(2);m1(2)];
h5 = [h1(2);h1(2);h1(2)];
%Table showing timing results of each method and for each value of m.
Table5 = table(k5,m5,h5,c5(:),d5(:),e5(:),fd5(:),h5(:), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time_dst','time_mg'});

k6 = [k1(3);k1(3);k1(3)];
m6 = [m1(3);m1(3);m1(3)];
h6 = [h1(3);h1(3);h1(3)];
%Table showing timing results of each method and for each value of m.
Table6 = table(k6,m6,h6,c6(:),d6(:),e6(:),fd6(:),h6(:), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time_dst','time_mg'});

Table = [Table4; Table5; Table6]

%mean
Tablem4 = table(k1(1),m1(1),h1(1),mean(c4),mean(d4),mean(e4),mean(fd4),mean(h4), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time
Tablem5 = table(k1(2),m1(2),h1(2),mean(c5),mean(d5),mean(e5),mean(fd5),mean(h5), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time
Tablem6 = table(k1(3),m1(3),h1(3),mean(c6),mean(d6),mean(e6),mean(fd6),mean(h6), 'VariableNames',{'k','m','h','t_stan','time_sor','time_sp','time

Table_mean = [Tablem4; Tablem5; Tablem6]

fprintf(' Make: Ilife Zed AIR plus \n Processor type: Intel Celeron CPU N3350\n Speed: @ 1.10 GHz x2 \n Memory: 6GB DDR III RAM\n');

fprintf(' (d). According to the computed mean wall clock time from Table_mean, fd2poissondst \n appears to be the best since it has the lowest co

fprintf(' Note: I used only k values from 4 to 5, because when i tried to run for k = 7 and above \n the MATLAB on my computer terminated, so i w
```

```
Table =

  9×8 table
```

k	m	h	t_stan	time_sor	time_sp	time_dst	time_mg
4	15	0.0625	0.32918	0.019242	0.185	0.17037	0.0625
4	15	0.0625	0.009003	0.003236	0.025851	0.002601	0.0625
4	15	0.0625	0.011332	0.002087	0.002174	0.000754	0.0625
5	31	0.03125	0.085766	0.025161	0.023151	0.01136	0.03125
5	31	0.03125	0.093103	0.008237	0.008022	0.020678	0.03125
5	31	0.03125	0.104	0.006533	0.007171	0.001901	0.03125
6	63	0.015625	2.9396	0.076385	0.03003	0.013109	0.015625
6	63	0.015625	2.7507	0.037827	0.027925	0.006052	0.015625
6	63	0.015625	2.8239	0.039376	0.027933	0.005864	0.015625

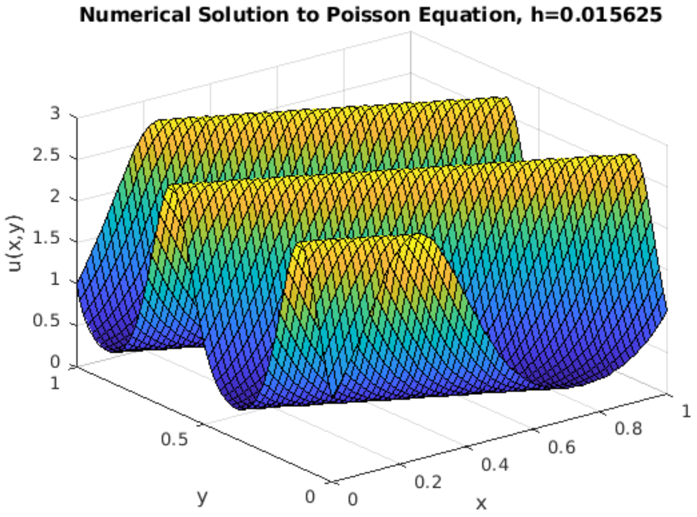
```
Table_mean =

  3×8 table
```

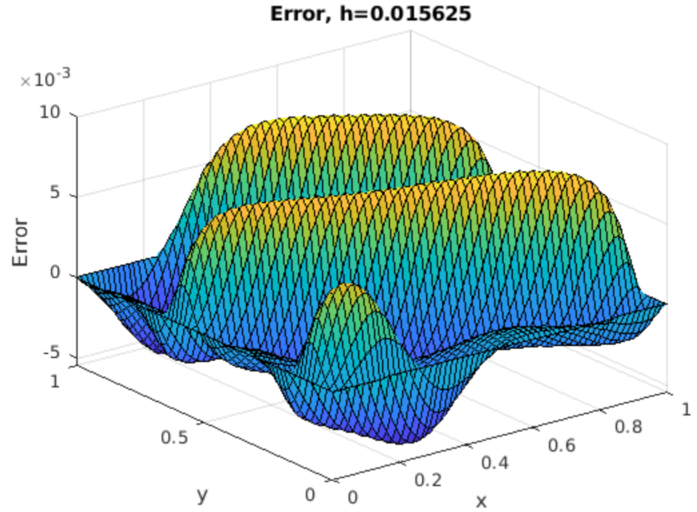
k	m	h	t_stan	time_sor	time_sp	time_dst	time_mg
4	15	0.0625	0.1165	0.0081883	0.071008	0.057909	0.0625
5	31	0.03125	0.094288	0.01331	0.012781	0.011313	0.03125
6	63	0.015625	2.8381	0.051196	0.028629	0.0083417	0.015625

```
 Make: Ilife Zed AIR plus
 Processor type: Intel Celeron CPU N3350
 Speed: @ 1.10 GHz x2
 Memory: 6GB DDR III RAM
 (d). According to the computed mean wall clock time from Table_mean, fd2poissondst
 appears to be the best since it has the lowest computation time  amongest all other method as m increases.
 Note: I used only k values from 4 to 5, because when i tried to run for k = 7 and above
 the MATLAB on my computer terminated, so i wouldnot perform any further simulations beyond k=6.
```

Plot solution

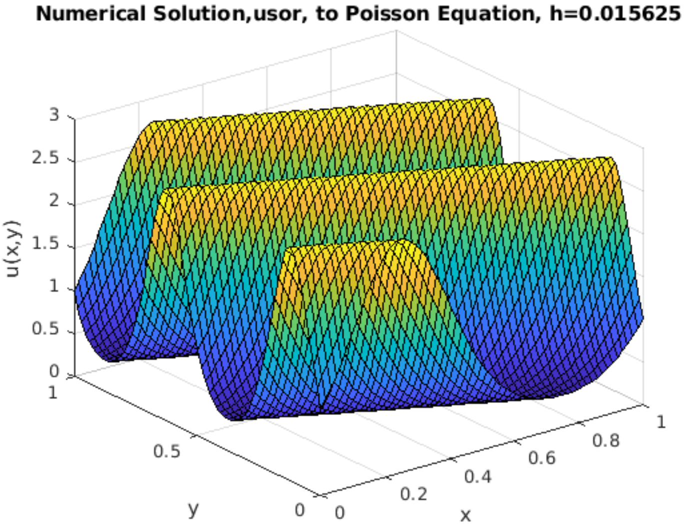
```
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,u), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution to Poisson Equation, h=',num2str(h)));

 %Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,u-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Error, h=',num2str(h)));
```

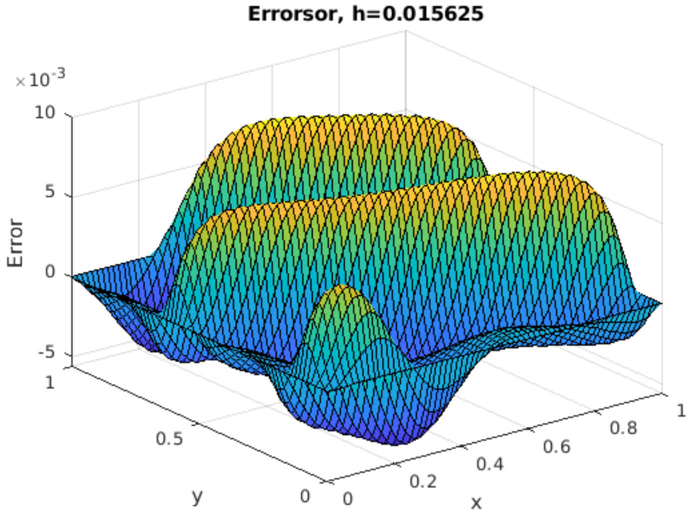




Plot solution

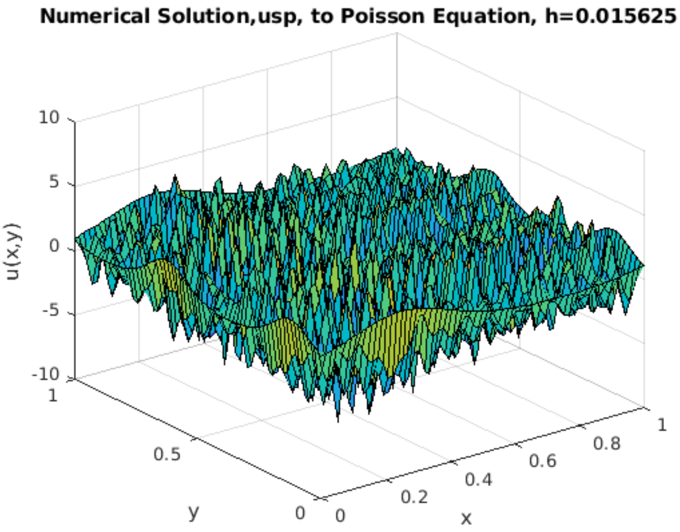
```
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,usor), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution,usor, to Poisson Equation, h=',num2str(h)));

% Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,usor-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Errorsor, h=',num2str(h)));
```

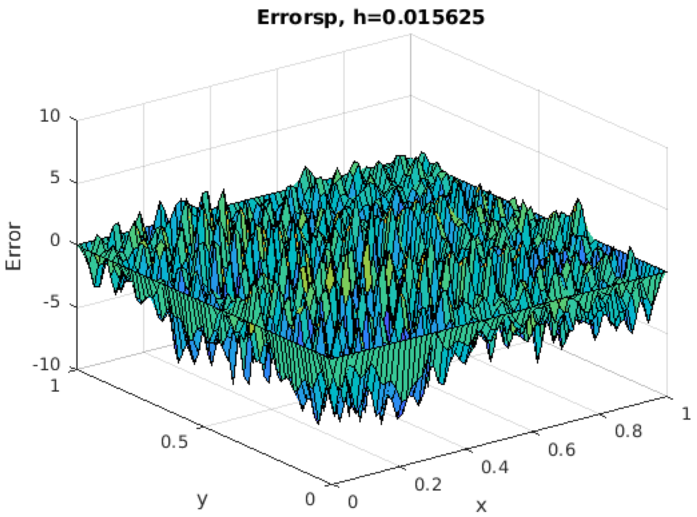




Plot solution

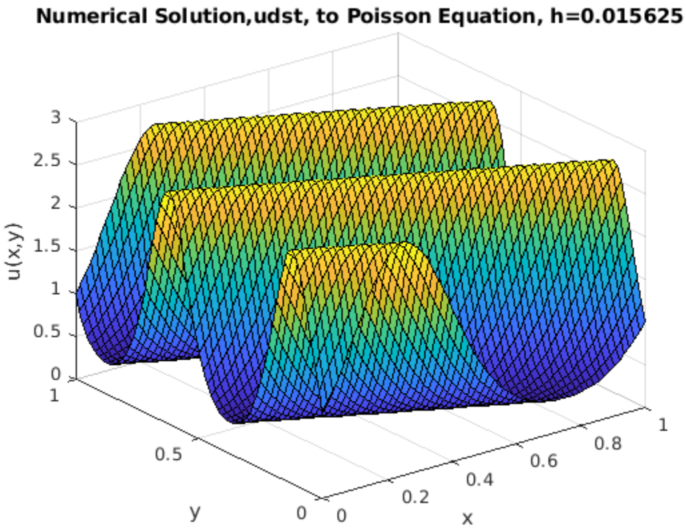
```
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,usp), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution,usp, to Poisson Equation, h=',num2str(h)));

% Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,usp-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Errorsp, h=',num2str(h)));
```

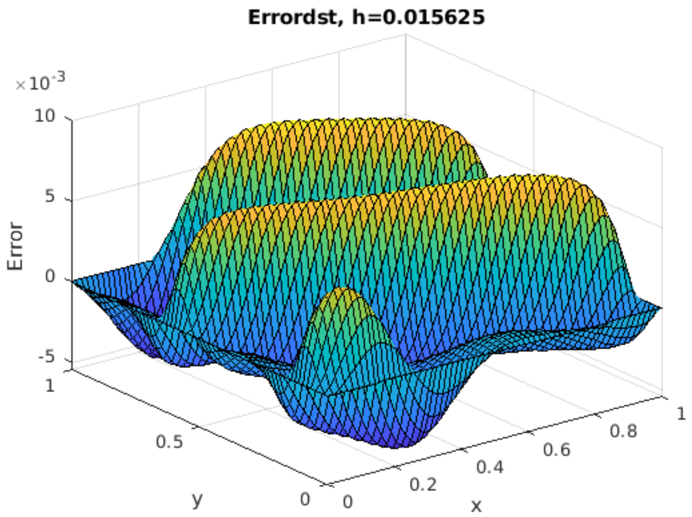




Plot solution

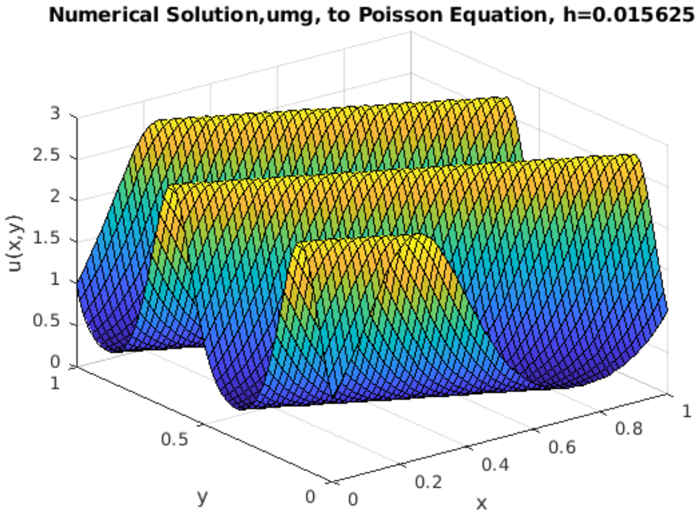
```
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,udst), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution,udst, to Poisson Equation, h=',num2str(h)));

% Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,udst-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Errordst, h=',num2str(h)));
```

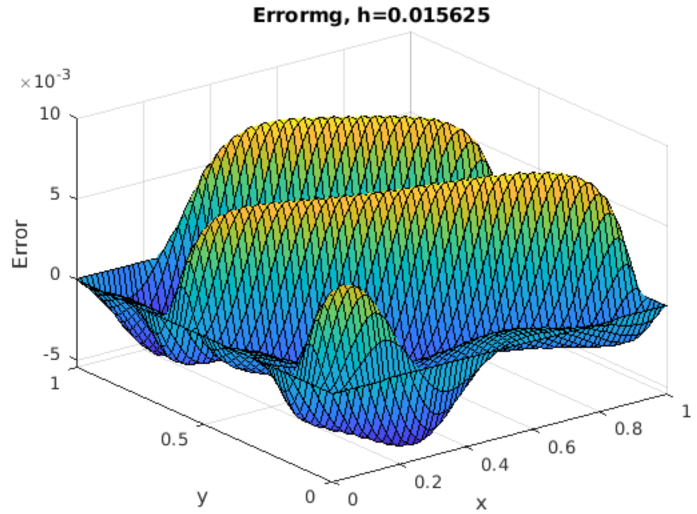




Plot solution

```
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,umg), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
title(strcat('Numerical Solution,umg, to Poisson Equation, h=',num2str(h)));

% Plot error
figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
surf(x,y,umg-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
title(strcat('Errormg, h=',num2str(h)));
```





Published with MATLAB® R2020a

```
% Numerical approximation to poisson's equation over the square [a,b] x
% [a,b] with zero Neumann boundary conditions. Uses a unifoorm mesh with
% (n+2) x (n+2) total points.

% Solves with the DCT

% Input
%   pfun : the RHS of poisson equation (i.e. the Laplacian of u). (f(x,y))
%    a,b : the interval defining the square
%    m : m+2 is the number of points in either direction of the mesh.
%

%Output
%   u : the numerical solution of poisson equation at the mesh points.
% x,y : the uniform mesh

function [u,x,y] = fd2poissondct(p,a,b,m)

h=1/(m+1);

% idx and idy need to include all the grid points:
idx = 1:m+2;
idy = 1:m+2;

[x,y] = meshgrid(a:h:b); %uniform mesh, including boundary points.

% Evaluate the RHS of Poisson's equation at the interior points.
fr = feval(p,x(idy,idx),y(idy,idx));

% Computation of fhat=(S*f)*S^(-1), where S is the DCT
fhat=idct(dct(fr,1),2);

% Denominator for the computation of uhat:
denom = [bsxfun(@plus,cos(pi*(idx-1)./(m+1)).',cos(pi*(idx-1)./(m+1)))-2];

uhat = h^2/2*(fhat./denom);

%Dealing with the zero eigenvalue.
uhat(1)=0;

% Computation of u = (S^(-1)*uhat)*S
u = dct(idct(uhat,1),2);

end
```

```
Not enough input arguments.

Error in fd2poissondct (line 19)
h=1/(m+1);
```

Published with MATLAB® R2020a

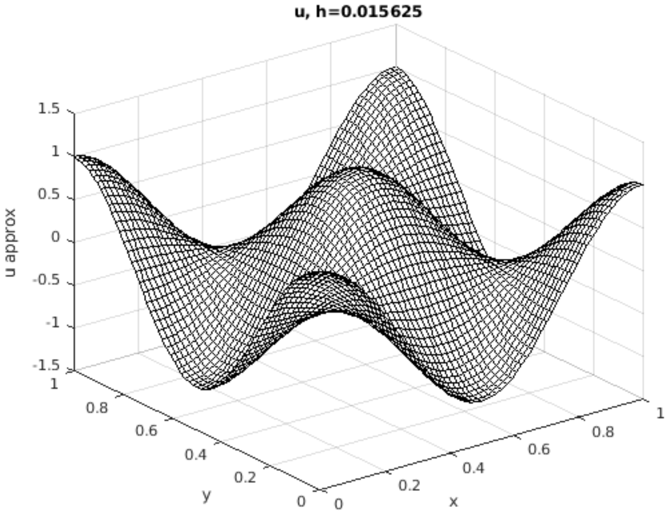
No. 3(b)

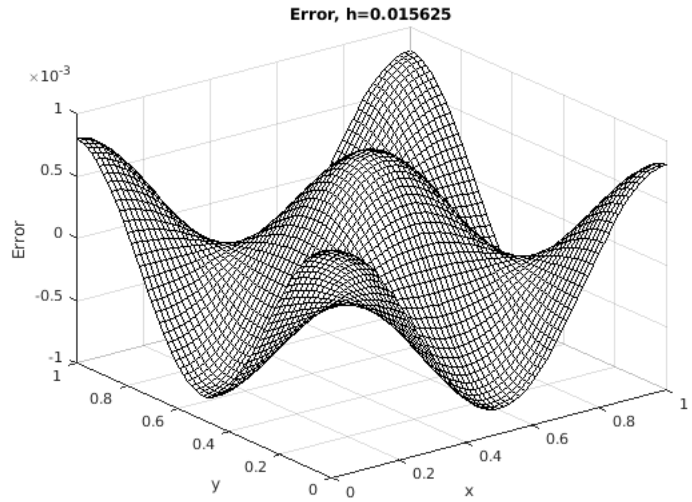
```
%Using the code from part(a) to solve the Poisson equation with f(x,y) =  -8*(pi^2)*(cos(2*pi*x)).*(cos(2*pi*y))

m=(2^6)-1;
a=0;b=1;
h=(b-a)/(m+1);

%fuction f(x,y)
pfun=@(x,y) -8*(pi^2)*(cos(2*pi*x)).*(cos(2*pi*y));

%Approximated
[u,x,y]=fd2poissondct(pfun,a,b,m);

%Numerical solution to the poisson equation
figure, set(gcf,'DefaultAxesFontSize',8,'PaperPosition', [0 0 3.5 3.5]),
mesh(x,y,u), colormap([0 0 0]),xlabel('x'),ylabel('y'),
zlabel('u_ approx'), title(strcat('u, h=',num2str(h)));

%Exact function
uex=@(x,y) (cos(2*pi*x)).*(cos(2*pi*y));
ue=uex(x,y);
error = (u-ue);

%Plot error
figure, set(gcf,'DefaultAxesFontSize',8,'PaperPosition', [0 0 3.5 3.5]),
mesh(x,y,error), colormap([0 0 0]),xlabel('x'),ylabel('y'),
zlabel('Error'), title(strcat('Error, h=',num2str(h)));

%Table showing the convergence of the solution to the true solution.
k1 = zeros(7,1);
h1=zeros(7,1);
L2=zeros(7,1);
m1=zeros(7,1);

for k = 4:10
    k1(k-3) = k;
    m1(k-3) = (2^k) - 1;
    m = (2^k) - 1;
    h1(k-3) = (b-a)/(m+1);
    h = (b-a)/(m+1);

    [x1,y1] = meshgrid(a:h:b);

    [u,x1,y1] = fd2poissondct(pfun,a,b,m);
    ue = uex(x1,y1);

    error = u - uex(x1,y1);

    L2(k-3) = R2Norm(error,ue);
end

%table
T = table(k1(:),m1(:),h1(:),L2(:), 'VariableNames',{'k','m','h','R2-norm'})
fprintf('Its clear from the table that as m increases due to increasing k, \n h decreases, and the value of the relative 2-norm significantly dec

%polyfit
p=polyfit(log(h1),log(L2),1);
p
fprintf('Since the order of convergence,p, is 2.0014, which is approximately 2, \n hence the method is second order accurate.\n')

function L2 = R2Norm(error, uexact)
    R = error .^2;
    u_ex = uexact.^2;
    L2 = sqrt(sum(R,'all')/sum(u_ex,'all'));
end
```

```
T =

  7×4 table
```

k	m	h	R2-norm
4	15	0.0625	0.012951
5	31	0.03125	0.003219
6	63	0.015625	0.00080358
7	127	0.0078125	0.00020082
8	255	0.0039062	5.0201e-05
9	511	0.0019531	1.255e-05
10	1023	0.00097656	3.1375e-06

```
Its clear from the table that as m increases due to increasing k,
 h decreases, and the value of the relative 2-norm significantly decreases as m grows big.
 Hence the big the m, the faster the solution converges to the true solution.

p =

    2.0014    1.1992

Since the order of convergence,p, is 2.0014, which is approximately 2,
 hence the method is second order accurate.
```





Published with MATLAB® R2020a

No. 4(b)

```
% Numerical approximation to Poisson's equation over the square [a,b]x[a,b] with
% Dirichlet boundary conditions.  Uses a uniform mesh with (n+2)x(n+2) total
% points (i.e, n interior grid points) on nine node.
% Input:
%     ffun : the RHS of poisson equation (i.e. the Laplacian of u).
%     gfun : the boundary function representing the Dirichlet B.C.
%      a,b : the interval defining the square
%        m : m+2 is the number of points in either direction of the mesh.
% Ouput:
%        u : the numerical solution of Poisson equation at the mesh points.
%      x,y : the uniform mesh.

function [u,x,y] = SOR(ffun,gfun,a,b,m,w)

h = (b-a)/(m+1); %mesh spacing

tol = 1e-16;   %relative residual

maxiter = 10000;  %maximum value of k

[x,y] = meshgrid(a:h:b); %Uniform mesh, including boundary points.

idx = 2:m+1;
idy = 2:m+1;
dx = 1:m+2;
dy = 1:m+2;

u = zeros(m+2);

% Compute boundary terms, south, north, east, west
u(1,1:m+2)    = feval(gfun,x(1,1:m+2),y(1,1:m+2));        % Include corners
u(m+2, 1:m+2) = feval(gfun,x(m+2,1:m+2),y(m+2,1:m+2)); % Include corners
u(idy,m+2)    = feval(gfun,x(idx,m+2),y(idy,m+2));        % No corners
u(idy,1)      = feval(gfun,x(idy,1),y(idy,1));              % No corners

% Evaluate the RHS of Poisson's equation at the interior points.
f = feval(ffun,x(dy,dx),y(dy,dx));

for k = 0:maxiter
    %Iterate
    for j = 2:m+1
        for i = 2:m+1
            u(i,j) = (1-w)*u(i,j)+(w/5)*(u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1))...
                +(w/20)*(u(i-1,j-1)+u(i+1,j-1)+u(i+1,j+1)+u(i-1,j+1))...
            -(h^2/20)*w*(4*f(i,j)+0.5*(f(i-1,j)+f(i+1,j)+f(i,j-1)+f(i,j+1)));
        end
    end

    %Compute the residual
    residual = zeros(m+2);

    for j = 2:m+1
        for i = 2:m+1
            residual(i,j) = -20*u(i,j)+4*(u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1))...
            +(u(i-1,j-1)+u(i+1,j-1)+u(i+1,j+1)+u(i-1,j+1))...
            -(h^2)*(4*f(i,j)+0.5*(f(i-1,j)+f(i+1,j)+f(i,j-1)+f(i,j+1)));
        end
    end

    %Determine if convergence has been reached
        if norm(residual(:),2) < tol*norm(f(:),2)
                break
    end
```

```
end
end
```

```
Not enough input arguments.

Error in SOR (line 15)
h = (b-a)/(m+1); %mesh spacing
```

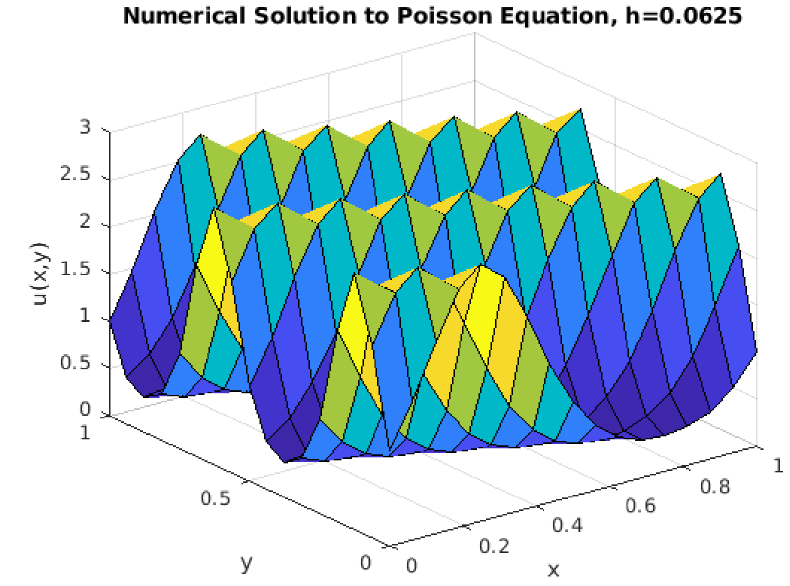
Published with MATLAB® R2020a

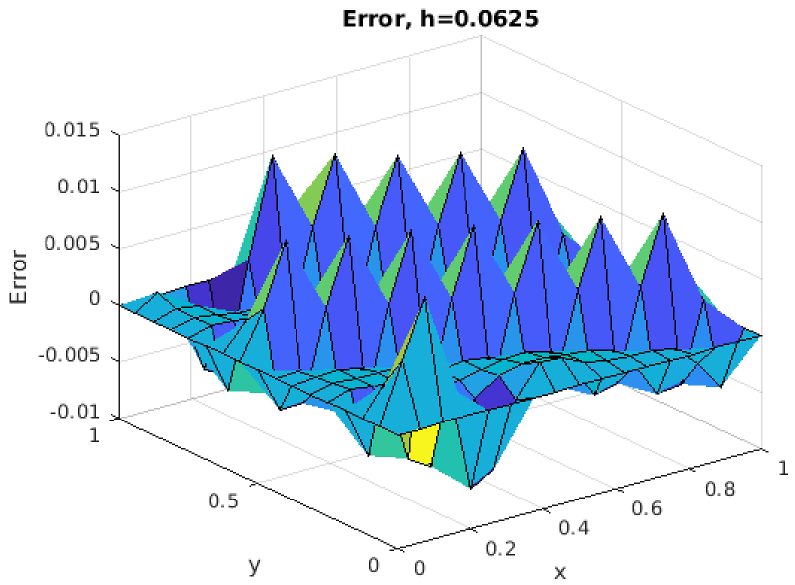
```
% Uses SOR function to to solve the poisson eqaution from problem 2 for
% various values of m and produce plots and tables that clearly show the
% forth order accuracy of the method.

a=0; b=1;

% Laplacian(u) = f
f = @(x,y) 10*pi^2*(1+cos(4*pi*(x+2*y))-2*sin(2*pi*(x+2*y))).*exp(sin(2*pi*(x+2*y)));
% u = g on Boundary
g = @(x,y) exp(sin(2*pi*(x+2*y)));

%Table showing the forth order acuracy of the method.
k1 = zeros(4,1);
h1=zeros(4,1);
L2=zeros(4,1);
m1=zeros(4,1);

for k = 4:7
    k1(k-3) = k;
    m1(k-3) = (2^k) - 1;
    m = (2^k) - 1;
    h1(k-3) = (b-a)/(m+1);
    h = (b-a)/(m+1);

    w = 2/(1+sin(pi*h)); %optimal relaxation parameter

    [x,y] = meshgrid(a:h:b);

    %Numerical solution
    [u,x,y] = SOR(f,g,a,b,m,w);

    % Exact solution is g.
    uexact = @(x,y) g(x,y);

    %Error
    error = u -uexact(x,y);

    %Relative 2-norm
    L2(k-3) = R2Norm(error,uexact(x,y));

    % Plot solution
    figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
    surf(x,y,u), xlabel('x'), ylabel('y'), zlabel('u(x,y)'),
    title(strcat('Numerical Solution to Poisson Equation, h=',num2str(h)));

    % Plot error
    figure, set(gcf,'DefaultAxesFontSize',10,'PaperPosition', [0 0 3.5 3.5]),
    surf(x,y,u-uexact(x,y)),xlabel('x'),ylabel('y'), zlabel('Error'),
    title(strcat('Error, h=',num2str(h)));

end

%table
T = table(k1(:),m1(:),h1(:),L2(:), 'VariableNames',{'k','m','h','R2-norm'})

%polyfit
p=polyfit(log(circshift(h1,size(h1))),log(L2),1);
p
fprintf('Since the order of convergence,p, is 4.1172, which is approximately 4, \n hence the method is fourth order accurate.\n')

plot(h1,L2);
xlabel('h');
ylabel('R 2-norm');
title('A graph of h against R 2-norm');

function L2 = R2Norm(error, uexact)
    R = error .^2;
    u_ex = uexact.^2;
    L2 = sqrt(sum(R,'all')/sum(u_ex,'all'));
end
```

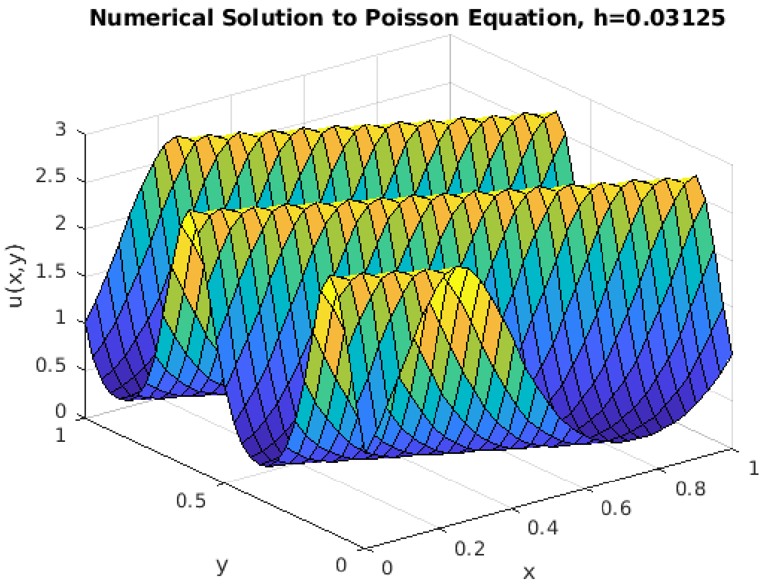
```
T =

  4×4 table
```

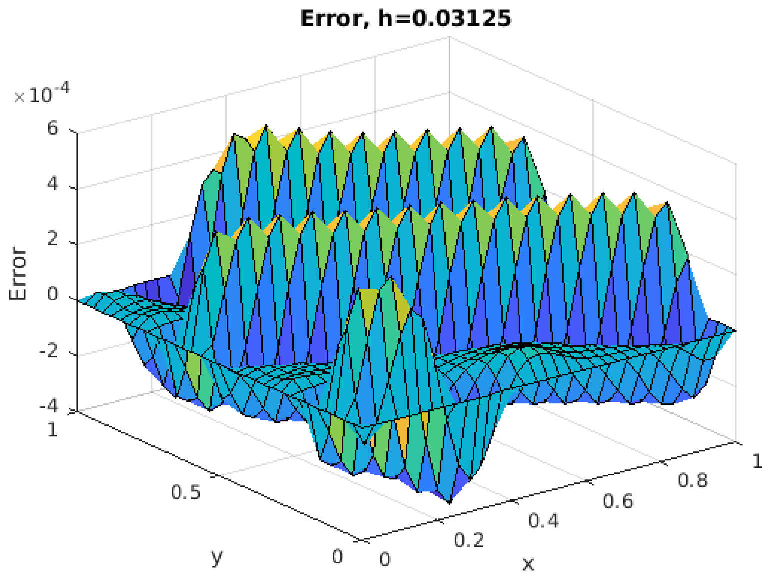
k	m	h	R2-norm
4	15	0.0625	0.0021715
5	31	0.03125	0.0001109
6	63	0.015625	6.6201e-06
7	127	0.0078125	4.1065e-07

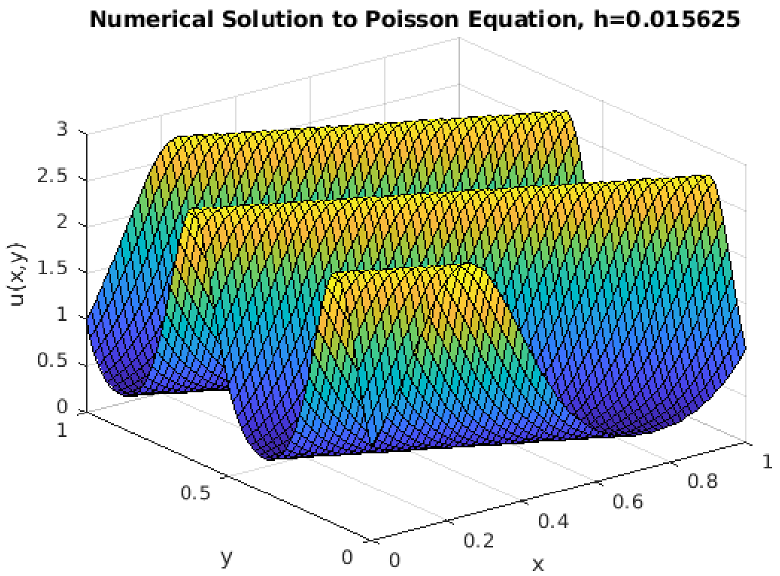
```
p =

    4.1172    5.2284

Since the order of convergence,p, is 4.1172, which is approximately 4,
 hence the method is fourth order accurate.
```

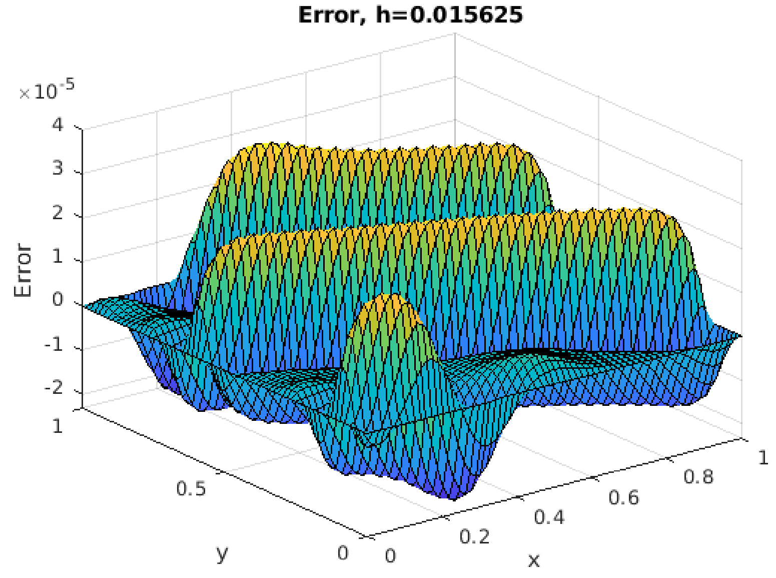


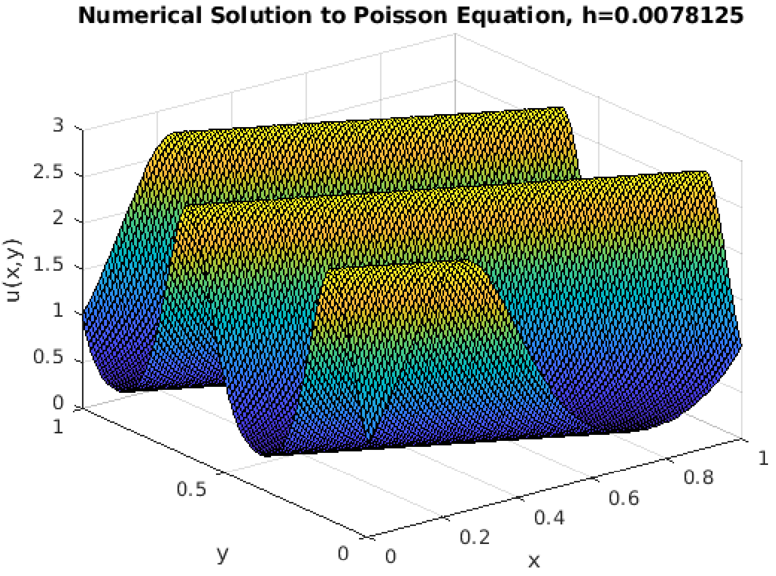


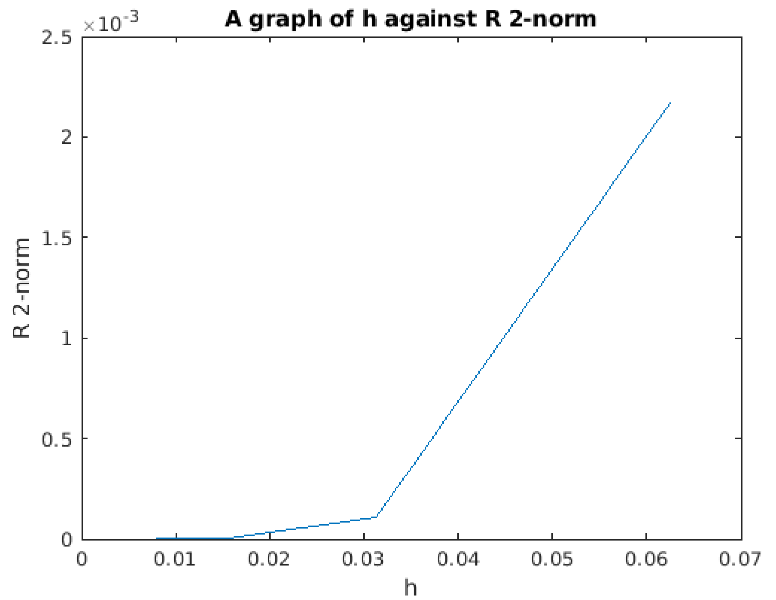

Numerical Solution to Poisson Equation, h=0.03125
u(x,y)
x
y


Error, h=0.03125
$\times 10^{-4}$
Error
x
y


Numerical Solution to Poisson Equation, h=0.015625
u(x,y)
x
y


Error, h=0.015625
×10^-5
4
3
2
1
0
-1
-2
Error
1
0.5
y
0
0
0.2
0.4
0.6
0.8
1
x


Numerical Solution to Poisson Equation, h=0.0078125
3
2.5
2
1.5
1
0.5
0
u(x,y)
1
0.5
y
0
0
0.2
0.4
0.6
0.8
1
x


A graph of h against R 2-norm
×10^-3
2.5
2
1.5
1
0.5
0
R 2-norm
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
h

3(a) Fast Poisson Solver with Neumann B.C

$$\nabla^2 u = f(x,y) \qquad (x,y) \in \Omega = (a,b) \times (a,b)$$

$$n \cdot \nabla u(x,y) = 0 \qquad (x,y) \in \partial\Omega$$

For interior points.

$$\nabla^2 u = f(x,y)$$

$$\nabla^2 u = u_{xx} + u_{yy} = f_{ij}$$

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

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

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

$$u_{ij} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{ij}\right)$$

For Boundary points.

$$n \cdot \nabla u(x,y) = 0 \Rightarrow \frac{\partial u}{\partial x} = 0, \quad \frac{\partial u}{\partial y} = 0$$

Using the Centered Differnce Formula and the fictitious point method, we have

$$\frac{\partial u}{\partial x} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} = 0 \Rightarrow u_{i+1,j} = u_{i-1,j}$$

at $i=0 \Rightarrow u_{1,j} = u_{-1,j}$

at $i=m+2 \Rightarrow u_{m+2,j} = u_{m+1,j}$

Boundary Points along y

$$\frac{\partial u}{\partial y} = \frac{u_{i,j+1} - u_{i,j-1}}{2h} = 0 \Rightarrow u_{i,j+1} = u_{i,j-1}$$

for $j = 0 \Rightarrow u_{i,1} = u_{i,-1}$

for $j = m+2 \Rightarrow u_{i,m+3} = u_{i,m+1}$

therefore the second-order accurate FD method formed is

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{i,j}\right)$$

with

$$u_{i,m+3} = u_{i,m+1}$$
$$u_{i,1} = u_{i,-1}$$
$$u_{1,j} = u_{-1,j}$$
$$u_{m+2,j} = u_{m+1,j}$$

a) Using the technique from problem 4 of homework 2, derive the following implicit fourth-order accurate approximation to the 2-D Poisson equation $u_{xx}+u_{yy}=f$:

$$\frac{1}{6h^2}\begin{bmatrix}1 & 4 & 1\\ 4 & -20 & 4\\ 1 & 4 & 1\end{bmatrix}u = \frac{1}{12}\begin{bmatrix} & 1 & \\ 1 & 8 & 1\\ & 1 & \end{bmatrix}f + O(h^4)$$

Solution:

From Homework 2, for 1D we have:

$$\frac{1}{h^2}\left[\frac{-1}{12} \quad \frac{4}{3} \quad -\frac{5}{2} \quad \frac{4}{3} \quad -\frac{1}{12}\right]u = f + O(h^4)$$

Now in 2D: $\nabla^2 u(x,y) = f(x,y)$ — (1), it will become

$$\frac{1}{h^2}\Big[-\frac{1}{12}u_{i-2,j} + \frac{4}{3}u_{i-1,j} \; \frac{5}{2}u_{i,j} + \frac{4}{3}u_{i+1,j} - \frac{1}{12}u_{i+2,j} - \frac{1}{12}u_{i,j-2}$$
$$+ \frac{4}{3}u_{i,j-1} - \frac{5}{2}u_{i,j} + \frac{4}{3}u_{i,j+1} - \frac{1}{12}u_{i,j+2}\Big]$$
$$= f_{i,j} + O(h^4) \text{ —— } (2)$$

Using a technique from Homework (2), differentiate equation (1) twice with respect to x and y.

$$\nabla^2(\nabla^2 u(x,y)) = \nabla^2 f(x,y)$$

$$\nabla^2 f(x,y) = \frac{1}{h^2}\left[f_{i-1,j} + f_{i,j+1} + f_{i+1,j} + f_{i,j-1} - 4f_{i,j}\right] + O(h^2)$$

$$\nabla^2 u(x,y) = u_{xx} + u_{yy}$$

$$\nabla^2 u(x,y) = \frac{1}{h^2}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\right] + O(h^4)$$

So

$$\nabla^2(\nabla^2 u(x,y)) = \frac{1}{h^2}\left[\nabla^2 u_{i+1,j} + \nabla^2 u_{i-1,j} + \nabla^2 u_{i,j+1} + \nabla^2 u_{i,j-1} - 4\nabla^2 u_{i,j}\right] + O(h^4)$$

$$\nabla^2 u_{i-1,j} = \frac{1}{h^2}\left[u_{i-2,j} + u_{i,j} + u_{i-1,j-1} - 4u_{i-1,j} + u_{i-1,j+1}\right] + O(h^4) \quad \text{--- (3)}$$

$$\nabla^2 u_{i,j-1} = \frac{1}{h^2}\left[u_{i,j} + u_{i,j-2} + u_{i+1,j-1} + u_{i-1,j-1} - 4u_{i,j-1}\right] + O(h^4) \quad \text{--- (4)}$$

$$\nabla^2 u_{i+1,j} = \frac{1}{h^2}\left[u_{i,j} + u_{i+2,j} + u_{i+1,j-1} + u_{i+1,j+1} - 4u_{i+1,j}\right] + O(h^4) \quad \text{--- (5)}$$

$$\nabla^2 u_{i,j+1} = \frac{1}{h^2}\left[u_{i,j} + u_{i,j+2} + u_{i+1,j+1} + u_{i-1,j+1} - 4u_{i,j+1}\right] + O(h^4) \quad \text{--- (6)}$$

$$\nabla^2 u_{i,j} = \frac{1}{h^2}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right] + O(h^4)$$

$$-4\nabla^2 u_{i,j} = \frac{1}{h^2}\left[-4u_{i+1,j} - 4u_{i-1,j} - 4u_{i,j-1} - 4u_{i,j+1} + 16u_{i,j}\right] + O(h^4) \quad \text{--- (7)}$$

Adding Equations (3), (4), (5), (6), and (7) and then divide by h^2, we obtain:

$$\nabla^2(\nabla^2 U(x,y)) = \frac{1}{h^4}\Big[20U_{i,j} + U_{i-2,j} + 2U_{i-1,j-1} - 8U_{i-1,j} + 2U_{i-1,j+1} + U_{i,j-2} + 2U_{i+1,j-1} - 8U_{i,j-1} + U_{i+2,j} + 2U_{i+1,j+1} - 8U_{i+1,j} + U_{i,j+2} - 8U_{i,j+1}\Big] + O(h^4)$$

therefore:

$$\nabla^2(\nabla^2 u(x,y)) = \frac{1}{h^2}\Big[f_{i-1,j} + f_{i,j-1} + f_{i+1,j} + f_{i,j+1} - 4f_{i,j}\Big] \qquad \text{---- (8)}$$

Using a technique from Homework (2),

$$(2) + \frac{h^2}{12}(8)$$

$$\frac{1}{h^2}\begin{bmatrix} & & -\frac{1}{12} & & \\ & & \frac{4}{3} & & \\ -\frac{1}{12} & \frac{4}{3} & -5 & \frac{4}{3} & -\frac{1}{12} \\ & & \frac{4}{3} & & \\ & & -\frac{1}{12} & & \end{bmatrix} u + \frac{1}{h^2}\begin{bmatrix} & & \frac{1}{12} & & \\ & \frac{1}{6} & -\frac{2}{3} & \frac{1}{6} & \\ \frac{1}{12} & -\frac{2}{3} & \frac{5}{3} & -\frac{2}{3} & \frac{1}{12} \\ & \frac{1}{6} & -\frac{2}{3} & \frac{1}{6} & \\ & & \frac{1}{12} & & \end{bmatrix} u =$$

$$f_{i,j} + \begin{bmatrix} & \frac{1}{12} & \\ \frac{1}{12} & \frac{2}{3} & \frac{1}{12} \\ & \frac{1}{12} & \end{bmatrix} f + O(h^4)$$

Which reduces to:

$$\frac{1}{6h^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} u = \frac{1}{12}\begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} f + O(h^4)$$