3.1 a. Test this script by performing a grid refinement study to verify that it is second order accurate.
b. Modify the script so that it works on a rectangular domain $[a_x, b_x] \times [a_y, b_y]$, but still with $\Delta x = \Delta y = h$. Test your modified script on a non-square domain.
c. Further modify the code to allow $\Delta x \neq \Delta y$ and test the modified script.

*Solution.*
Here, the code was edited to be in a for loop so that the $m$ values would increase in size from 200 to 2000. We see that the order of accuracy converges to $\mathcal{O}(h^2)$.

The code here was editied to be a rectangular domain of $[0, 1] \times [0, 2]$. The code was also editied such that $h$ was the same with $h = 0.2$.

The code here was only slightly edited such that $h_x = 0.2$ and $h_y = 0.2222$.

```
% poisson2.m  -- solve the Poisson problem u_{xx} + u_{yy} = f(x,y)
% on [a,b] x [a,b].
%
% The 5-point Laplacian is used at interior grid points.
% This system of equations is then solved using backslash.
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/chapter3  (2007)

clear;
close all;
clc;

for i=1:10

a = 0;
b = 1;
m = 20*10*i;
h = (b-a)/(m+1);
x = linspace(a,b,m+2);   % grid points x including boundaries
y = linspace(a,b,m+2);   % grid points y including boundaries


[X,Y] = meshgrid(x,y);      % 2d arrays of x,y values
X = X';                     % transpose so that X(i,j),Y(i,j) are
Y = Y';                     % coordinates of (i,j) point

Iint = 2:m+1;              % indices of interior points in x
Jint = 2:m+1;              % indices of interior points in y
Xint = X(Iint,Jint);       % interior points
Yint = Y(Iint,Jint);

f = @(x,y) 1.25*exp(x+y/2);         % f(x,y) function

rhs = f(Xint,Yint);        % evaluate f at interior points for ...
    right hand side
                           % rhs is modified below for boundary ...
                              conditions.

utrue = exp(X+Y/2);        % true solution for test problem

% set boundary conditions around edges of usoln array:

usoln = utrue;              % use true solution for this test problem
                            % This sets full array, but only ...
                               boundary values
                            % are used below.  For a problem where ...
                               utrue
                            % is not known, would have to set each ...
                               edge of
                            % usoln to the desired Dirichlet ...
                               boundary values.


```

```
% adjust the rhs to include boundary terms:
rhs(:,1) = rhs(:,1) - usoln(Iint,1)/h^2;
rhs(:,m) = rhs(:,m) - usoln(Iint,m+2)/h^2;
rhs(1,:) = rhs(1,:) - usoln(1,Jint)/h^2;
rhs(m,:) = rhs(m,:) - usoln(m+2,Jint)/h^2;


% convert the 2d grid function rhs into a column vector for rhs of ...
    system:
F = reshape(rhs,m*m,1);

% form matrix A:
I = speye(m);
e = ones(m,1);
T = spdiags([e -4*e e],[-1 0 1],m,m);
S = spdiags([e e],[-1 1],m,m);
A = (kron(I,T) + kron(S,I)) / h^2;


% Solve the linear system:
uvec = A\F;

% reshape vector solution uvec as a grid function and
% insert this interior solution into usoln for plotting purposes:
% (recall boundary conditions in usoln are already set)

usoln(Iint,Jint) = reshape(uvec,m,m);

% assuming true solution is known and stored in utrue:
E=usoln-utrue;
err = max(max(abs(usoln-utrue)));
fprintf('Error relative to true solution of PDE = %10.3e \n',err)

% plot results:

clf
hold on

% plot grid:
% plot(X,Y,'g');  plot(X',Y','g');

% plot solution:
contour(X,Y,usoln,30,'k')

axis([a b a b])
daspect([1 1 1])
title('Contour plot of computed solution')
hold on
end
error_loglog(h, err);
hold on
```

```

% poisson2.m  -- solve the Poisson problem u_{xx} + u_{yy} = f(x,y)
% on [a,b] x [a,b].
%
% The 5-point Laplacian is used at interior grid points.
% This system of equations is then solved using backslash.
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/chapter3  (2007)

clear;
close all;
clc;

a = 0;
b = 1;
c = 0;
d = 2;
mx = 4;
my=8;
hx = (b-a)/(mx+1);
hy=(d-c)/(my+2);
x = linspace(a,b,mx+2);   % grid points x including boundaries
y = linspace(c,d,my+2);   % grid points y including boundaries


[X,Y] = meshgrid(x,y);      % 2d arrays of x,y values
X = X';                     % transpose so that X(i,j),Y(i,j) are
Y = Y';                     % coordinates of (i,j) point

Iint = 2:mx+1;              % indices of interior points in x
Jint = 2:my+1;              % indices of interior points in y
Xint = X(Iint,Jint);       % interior points
Yint = Y(Iint,Jint);

f = @(x,y) 1.25*exp(x+y/2);         % f(x,y) function

rhs = f(Xint,Yint);        % evaluate f at interior points for ...
    right hand side
                           % rhs is modified below for boundary ...
                              conditions.

utrue = exp(X+Y/2);        % true solution for test problem

% set boundary conditions around edges of usoln array:

usoln = utrue;              % use true solution for this test problem
                            % This sets full array, but only ...
                               boundary values
                            % are used below.  For a problem where ...
                               utrue
                            % is not known, would have to set each ...
                               edge of
                            % usoln to the desired Dirichlet ...
```

```
                                  boundary values.


% adjust the rhs to include boundary terms:
rhs(:,1) = rhs(:,1) - usoln(Iint,1)/hy^2;
rhs(:,my) = rhs(:,my) - usoln(Iint,my+2)/hy^2;
rhs(1,:) = rhs(1,:) - usoln(1,Jint)/hx^2;
rhs(mx,:) = rhs(mx,:) - usoln(mx+2,Jint)/hx^2;


% convert the 2d grid function rhs into a column vector for rhs of ...
    system:
F = reshape(rhs,mx*my,1);

% form matrix A:
Ix = speye(mx);
Iy=speye(my);
ex = ones(mx,1);
ey=ones(my,1);
T = spdiags([ex -4*ex ex],[-1 0 1],mx,mx);
S = spdiags([ey ey],[-1 1],my,my);
A = (kron(Iy,T) + kron(S,Ix)) / hx^2;


% Solve the linear system:
uvec = A\F;

% reshape vector solution uvec as a grid function and
% insert this interior solution into usoln for plotting purposes:
% (recall boundary conditions in usoln are already set)

usoln(Iint,Jint) = reshape(uvec,mx,my);

% assuming true solution is known and stored in utrue:
err = max(max(abs(usoln-utrue)));
fprintf('Error relative to true solution of PDE = %10.3e \n',err)

% plot results:

clf
hold on

%plot grid:
plot(X,Y,'g');  plot(X',Y','g')

% plot solution:
contour(X,Y,usoln,30,'k')

axis([a b c d])
daspect([1 1 1])
title('Contour plot of computed solution')
hold off
```

```

% poisson2.m  -- solve the Poisson problem u_{xx} + u_{yy} = f(x,y)
% on [a,b] x [a,b].
%
% The 5-point Laplacian is used at interior grid points.
% This system of equations is then solved using backslash.
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/chapter3  (2007)

clear;
close all;
clc;

a = 0;
b = 1;
c = 0;
d = 2;
mx = 4;
my=8;
hx = (b-a)/(mx+1);
hy=(d-c)/(my+1);
x = linspace(a,b,mx+2);   % grid points x including boundaries
y = linspace(c,d,my+2);   % grid points y including boundaries


[X,Y] = meshgrid(x,y);      % 2d arrays of x,y values
X = X';                     % transpose so that X(i,j),Y(i,j) are
Y = Y';                     % coordinates of (i,j) point

Iint = 2:mx+1;              % indices of interior points in x
Jint = 2:my+1;              % indices of interior points in y
Xint = X(Iint,Jint);       % interior points
Yint = Y(Iint,Jint);

f = @(x,y) 1.25*exp(x+y/2);         % f(x,y) function

rhs = f(Xint,Yint);        % evaluate f at interior points for ...
    right hand side
                           % rhs is modified below for boundary ...
                              conditions.

utrue = exp(X+Y/2);        % true solution for test problem

% set boundary conditions around edges of usoln array:

usoln = utrue;              % use true solution for this test problem
                            % This sets full array, but only ...
                               boundary values
                            % are used below.  For a problem where ...
                               utrue
                            % is not known, would have to set each ...
                               edge of
                            % usoln to the desired Dirichlet ...
```

```
                                  boundary values.


% adjust the rhs to include boundary terms:
rhs(:,1) = rhs(:,1) - usoln(Iint,1)/hy^2;
rhs(:,my) = rhs(:,my) - usoln(Iint,my+2)/hy^2;
rhs(1,:) = rhs(1,:) - usoln(1,Jint)/hx^2;
rhs(mx,:) = rhs(mx,:) - usoln(mx+2,Jint)/hx^2;


% convert the 2d grid function rhs into a column vector for rhs of ...
    system:
F = reshape(rhs,mx*my,1);

% form matrix A:
Ix = speye(mx);
Iy=speye(my);
ex = ones(mx,1);
ey=ones(my,1);
T = spdiags([ex -4*ex ex],[-1 0 1],mx,mx);
S = spdiags([ey ey],[-1 1],my,my);
A = (kron(Iy,T) + kron(S,Ix)) / (hx*hy);


% Solve the linear system:
uvec = A\F;

% reshape vector solution uvec as a grid function and
% insert this interior solution into usoln for plotting purposes:
% (recall boundary conditions in usoln are already set)

usoln(Iint,Jint) = reshape(uvec,mx,my);

% assuming true solution is known and stored in utrue:
err = max(max(abs(usoln-utrue)));
fprintf('Error relative to true solution of PDE = %10.3e \n',err)

% plot results:

clf
hold on

% plot grid:
% plot(X,Y,'g');  plot(X',Y','g')

% plot solution:
contour(X,Y,usoln,30,'k')

axis([a b c d])
daspect([1 1 1])
title('Contour plot of computed solution')
hold off
```

3.2 a. Show that the 9-point Laplacian (3.17) has the truncation error derived in Section 3.5.
b. Modify the matlab script poisson.m to use the 9-point Laplacian (3.17) instead of the 5-point Laplacian, and to solve the linear system (3.18) where $f_{ij}$ is given by (3.19). Perform a grid refinement study to verify that fourth order accuracy is achieved.

*Solution.*
We start with $\nabla_5^2 u(x_i, y_j) = \frac{1}{h^2}(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-4u_{i,j})$. Hence, $\nabla_9^2 u(x_i, y_j) = \frac{1}{6h^2}(4h^2\nabla_5^2 u(x_i, y_j)+u_{i-1,j-1}+u_{i+1,j-1}+u_{i-1,j-1}+u_{i+1,j+1}-4u_{i,j})$. We also know the local truncation error of the 5-point Laplacian is given by $\tau_{ij} = \frac{1}{12}h^2(u_{xxxx}+u_{yyyy})+\mathcal{O}(h^4)$. Now applying the Taylor series expansion of $u(x+h, y+h)$ centered at $u(x_i, y_j)$, we get

$$u(x+h,y+h) = u(x_i,y_j) + u(x_i,y_j)_x h + u(x_i,y_j)_y h + \frac{1}{2}u(x_i,y_j)_{xx}h^2 + u(x_i,y_j)_{xy}h^2$$
$$+\frac{1}{2}u(x_i,y_j)_{yy}h^2 + \frac{1}{6}u(x_i,y_j)_{xxx}h^3 + \frac{1}{2}u(x_i,y_j)_{xxy}h^3$$
$$+\frac{1}{2}u(x_i,y_j)_{xyy}h^3 + \frac{1}{6}u(x_i,y_j)_{yyy}h^3 + \frac{1}{24}u(x_i,y_j)_{xxxx}h^4$$
$$+\frac{1}{6}u(x_i,y_j)_{xxxy}h^4 + \frac{1}{4}u(x_i,y_j)_{xxyy}h^4 + \frac{1}{6}u(x_i,y_j)_{xyyy}h^4$$
$$+\frac{1}{24}u(x_i,y_j)_{yyyy}h^4 + \mathcal{O}(h^5)$$

Therefore

$$u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j-1} + u_{i+1,j+1} =$$
$$4u(x_i,y_j) + 2h^2\nabla^2 u(x_i,y_j) + h^4(\frac{1}{6}u(x_i,y_j)_{xxxx} + u(x_i,y_j)_{xxyy}$$
$$+\frac{1}{6}u(x_i,y_j)_{yyyy}) + \mathcal{O}(h^6)$$

Thus, we have that

$$\nabla_9^2 u(x_i,y_j) =$$
$$\frac{1}{6h^2}(\frac{1}{3}h^4(u(x_i,y_j)_{xxxx} + u(x_i,y_j)_{yyyy}) + 6h^2\nabla^2 u(x_i,y_j) +$$
$$h^4[\frac{1}{6}u(x_i,y_j)_{xxxx} + u(x_i,y_j)_{xxyy} + \frac{1}{6}u(x_i,y_j)_{yyyy}] + \mathcal{O}(h^6)) =$$
$$\nabla^2 u(x_i,y_j) + \frac{1}{12}h^2\nabla^2(\nabla^2 u(x_i,y_j)) + \mathcal{O}(h^4) =$$

$$\nabla^2 u(x_i, y_j) + \frac{1}{12}h^2\nabla^2 f(x_i, y_j) + \mathcal{O}(h^4)$$

Whence, the local truncation error for the 9-point Laplacian is given by $\tau_{ij} = \nabla_9^2 u(x_i, y_j) - \nabla^2 u(x_i, y_j) = \frac{1}{12}h^2\nabla^2 f(x_i, y_j) + \mathcal{O}(h^4)$ as derived in scetion 3.5.

The modified code here is of the 9 point Laplacian. Also, upon a grid refinement, we find that the order of accuracy indeed converges to fourth order.

```matlab
clear;
close all;
clc;

for i=1:10

a = 0;
b = 1;
m = 40*2*i;
h = (b-a)/(m+1);
x = linspace(a,b,m+2);    % grid points x including boundaries
y = linspace(a,b,m+2);    % grid points y including boundaries


[X,Y] = meshgrid(x,y);       % 2d arrays of x,y values
X = X';                      % transpose so that X(i,j),Y(i,j) are
Y = Y';                      % coordinates of (i,j) point

Iint = 2:m+1;               % indices of interior points in x
Jint = 2:m+1;               % indices of interior points in y
Xint = X(Iint,Jint);        % interior points
Yint = Y(Iint,Jint);

f = @(x,y) 1.25*exp(x+y/2);          % f(x,y) function

rhs = f(Xint,Yint)+(h^2/12)*1.25*f(Xint,Yint);        % evaluate f ...
    at interior points for right hand side
                            % rhs is modified below for boundary ...
                               conditions.

utrue = exp(X+Y/2);         % true solution for test problem

% set boundary conditions around edges of usoln array:

usoln = utrue;               % use true solution for this test problem
                             % This sets full array, but only ...
                                boundary values
                             % are used below.  For a problem where ...
                                utrue
                             % is not known, would have to set each ...
                                edge of
                             % usoln to the desired Dirichlet ...
                                boundary values.


% form matrix A:
I = speye(m);
e = ones(m,1);
T= spdiags([(4/h^2)*e (-10/h^2 - 10/h^2)*e (4/h^2)*e], [-1 0 1], ...
    m, m);
T2=spdiags([(1/h^2)*e (2/h^2 + 2/h^2)*e (1/h^2)*e], [-1 0 1], m, m);
K = spdiags([e 4*e e],[-1 0 1],m,m);
S = spdiags([e e],[-1 1],m,m);
```

```
A =(1/6)*(kron(I,T) + kron(S,T2));

% adjust the rhs to include boundary terms:
rhs(:,1) = rhs(:,1) - K*usoln(Iint,1)/(6*h^2);
rhs(:,m) = rhs(:,m) - K*usoln(Iint,m+2)/(6*h^2);
rhs(1,:) = rhs(1,:) - (usoln(1,Jint)*K)/(6*h^2);
rhs(m,:) = rhs(m,:) - (usoln(m+2,Jint)*K)/(6*h^2);
rhs(1,1) = rhs(1,1) - usoln(1, 1)/(6*h^2);
rhs(1,m) = rhs(1,m) - usoln(1, m+2)/(6*h^2);
rhs(m,1) = rhs(m,1) - usoln(m+2, 1)/(6*h^2);
rhs(m,m) = rhs(m,m) - usoln(m+2, m+2)/(6*h^2);


% convert the 2d grid function rhs into a column vector for rhs of ...
    system:
F = reshape(rhs,m*m,1);

% Solve the linear system:
uvec = A\F;

% reshape vector solution uvec as a grid function and
% insert this interior solution into usoln for plotting purposes:
% (recall boundary conditions in usoln are already set)

usoln(Iint,Jint) = reshape(uvec,m,m);

% assuming true solution is known and stored in utrue:
err = max(max(abs(usoln-utrue)));
fprintf('Error relative to true solution of PDE = %10.3e \n',err)

% plot results:

clf
hold on

% plot grid:
% plot(X,Y,'g');  plot(X',Y','g')

% plot solution:
contour(X,Y,usoln,30,'k')

axis([a b a b])
daspect([1 1 1])
title('Contour plot of computed solution')
hold off

end

error_loglog(h,err);
```