Problem P18: (a) Consider the following BFP with Dirichlet boundary conditions:

$$u''(x) + u(x) = 0 \qquad \text{for,} \qquad a < x < b, \qquad u(a) = \alpha, \qquad u(b) = \beta$$

Write a Matlab, finite difference code to solve this problem.

Solution:
With a small modification to the code we wrote for P15 we get a three point centered finite difference scheme to solve this problem.
Code:

```
function [x,U] = lchdbvp(m, a, b, alpha, beta)
% This function uses a centered finite difference scheme to solve
% the following linear, constant coefficient, homogeneous,
% dirichlet boundary value problem:
%
%   u''(x) + u(x) = 0, u(a) = \alpha, u(b) = \beta
%
% Using m + 2 point grid and the following finite difference approx.
%
%    D^2U_j = 1/h^2(U_{j - 1} -2U_j +U_{j + 1})
%


x = linspace(a, b, m+2); % Generating grid
h = (b - a)/(m + 1); % Grid spacing
% Generating A
A = (1/h^2).*(
    diag(((h^2 - 2)*ones(m, 1))) +
    diag(ones(m - 1, 1), 1) +
    diag(ones(m - 1, 1), -1)
    );

% Generating F
F = zeros(m, 1);
F(1) = F(1) - alpha/h^2;
F(m) = F(m) - beta/h^2;

% Solving U
U = A\F;
U = [alpha; U; beta];
```

(b) Determine the exact solution to the problem when $a = 0$, $b = 1$, $\alpha = 2$, and $\beta = 3$. test your code from part (a) using this solution, including a demonstration of convergence at the optimal rate (which is what?) as $h \to 0$. Generate a convergence figure.

Solution:
Solving this problem by hand, first we form the characteristic equation,

$$r^2 + 1 = 0$$

Which gives roots $r = \pm i$. So we get the following general solution,

$$u(x) = c_1 e^{ix} + c_2 e^{-ix}.$$

Applying Euler's Formula we get

$$u(x) = c_1(\cos(x) + i\sin(x)) + c_2(\cos(x) - i\sin(x)).$$

Combining like terms and redefining our coefficient terms we get the general solution,

$$u(x) = c_1 \cos(x) + c_2 \sin(x).$$

Solving the IVP with $u(0) = 2$ and $u(1) = 3$, we get the following,

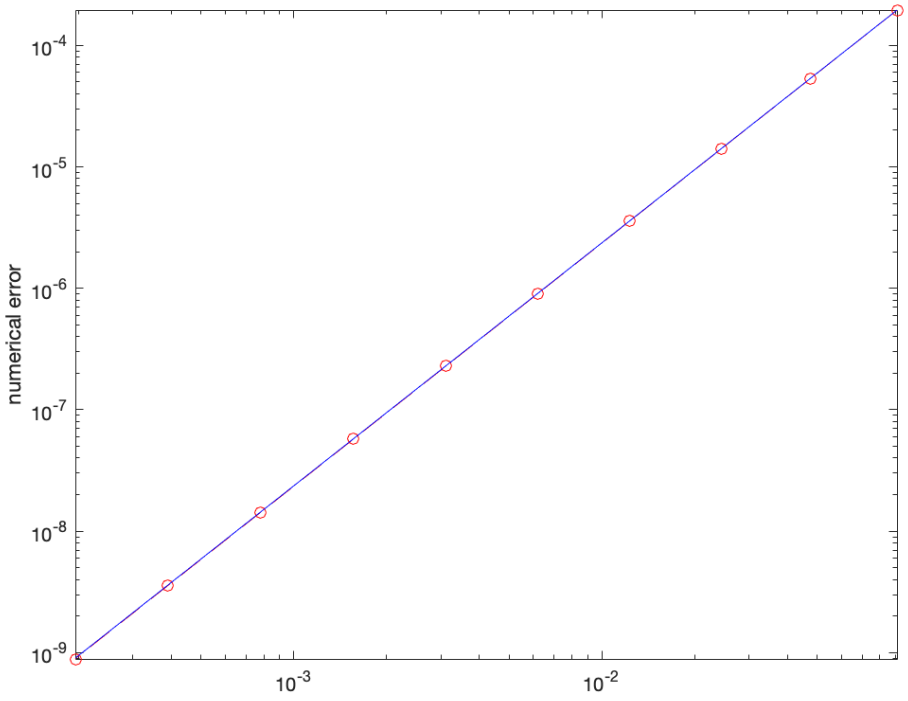
$$c_1 = 2,$$
$$c_2 = \frac{3 - 2\cos(1)}{\sin(1)}.$$

So our solution looks like,

$$u(x) = 2\cos(x) + \frac{3 - 2\cos(1)}{\sin(1)} \sin(x)$$

Testing our code using this solution we get a convergence rate of $O(h^p)$ where $p \approx 2.001056$. Which is as exprected with the optimal convergence rate of $O(h^2)$. Below is the convergence figure and code, we see the optimal convergence graphed in purple, and our experimental convergence graphed in with circles.

Figure 1: Convergence Plot via Discrete L^2 Norm



Console:

```
f = @(x) 2*cos(x) + ((3 - 2*cos(1))/sin(1))*sin(x);
M = [10 20 40 80 160 320 640 1280, 2560, 5120];
Error = zeros(1, 10);
Spacing = zeros(1, 10);
g = @(x) 0*x;

for i = 1:10
    Spacing(i) = 1/(M(i) + 1);
    [x,U] = lchdbvp(M(i), 0, 1, 2, 3);
    Error(i) = sqrt(Spacing(i)) * norm(U - f(x)',2);
end

p = polyfit(log(Spacing),log(Error),1);
fprintf('convergence at rate O(h^p) with p = %f\n',p(1))
loglog(Spacing,Error,'o', ...
    Spacing,exp(p(2) + p(1)*log(Spacing)),'r--', ...
    Spacing, exp(p(2) + 2*log(Spacing)),'b--') %Optimal
xlabel h,  ylabel('numerical error')

axis tight
```

(c) Let $a = 0$ and $b = \pi$. For what values of α and β does $BVP(1)$ have solutions? sketch a family of solutions in a case where there are infinitely-many solutions.

Solution:
Recall that the general solution to $BVP(1)$ is given by,

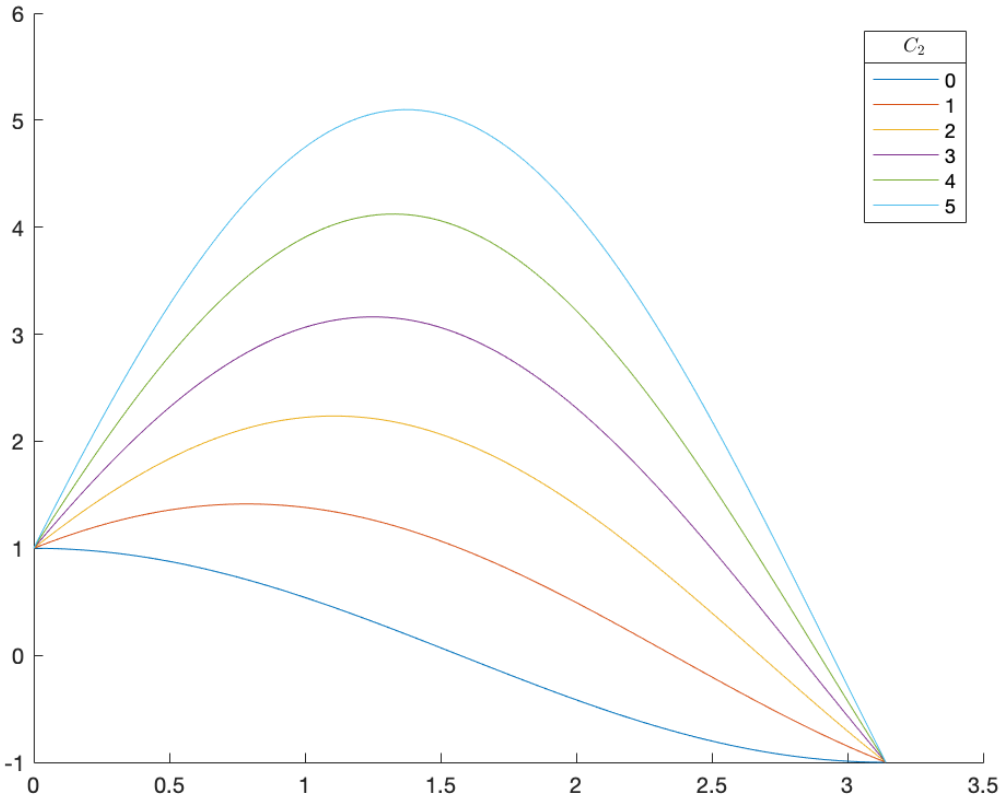
$$u(x) = c_1 \cos(x) + c_2 \sin(x).$$

Note that $a = 0$ and $b = \pi$ generates the following system,

$$\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

For this system to have a solution $\alpha = -\beta$, with c_2 as a free variable. The following is a plot of a family of solution when $\alpha = C_1 = 1$ and C_2 is allowed to vary.

Figure 2: Family of Solutions for BVP(1) with $a = 0$, $b = \pi$, and $\alpha = c_1 = 1$.



Console:

```
x = linspace(0, pi, 100);
hold on
for i = 0:5
    j = @(x) cos(x) + i*sin(x);
    plot(x, j(x));
end
lgd = legend('0', '1', '2', '3', '4', '5');
title(lgd, '$$C_2$$', 'interpreter','latex')
```

Problem P19: Write a program to solve the BVP for the nonlinear pendulum as discussed in the text, i.e. problem (2.77), using the Newton iteration strategy outlined in subsection 2.16.1. Reproduce Figures 2.4(b) and 2.5, for which $T = 2\pi$, and $\alpha = \beta = .7$.

Solution:
Recall that the nonlinear BVP described in the text is,

$$\theta''(t) = sin(\theta(t)) \qquad \text{for } 0 < t < T$$
$$\theta(0) = \alpha \qquad \theta(T) = \beta$$

The Newton iteration strategy described in the text involves using a centered finite difference approximation for the $\theta''(t)$ term to create a system of non-linear equations,

$$\frac{1}{h^2}(\theta_{i-1} - 2\theta_i + \theta_{i+1}) + \sin(\theta_i) = 0.$$

Note that this system is defined in m dimensions using the same discretization scheme with $m+2$ equally spaced points between 0 and T as we've used before. To formulate the newton step we need to generate the Jacobian,

$$J = \frac{1}{h^2}\begin{bmatrix} (-2 + h^2\cos(\theta_1)) & 1 & & & \\ 1 & (-2 + h^2\cos(\theta_2)) & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & (-2 + h^2\cos(\theta_m)) \end{bmatrix}.$$

The following code solves the double pendulum BVP problem via this Newton iteration strategy.

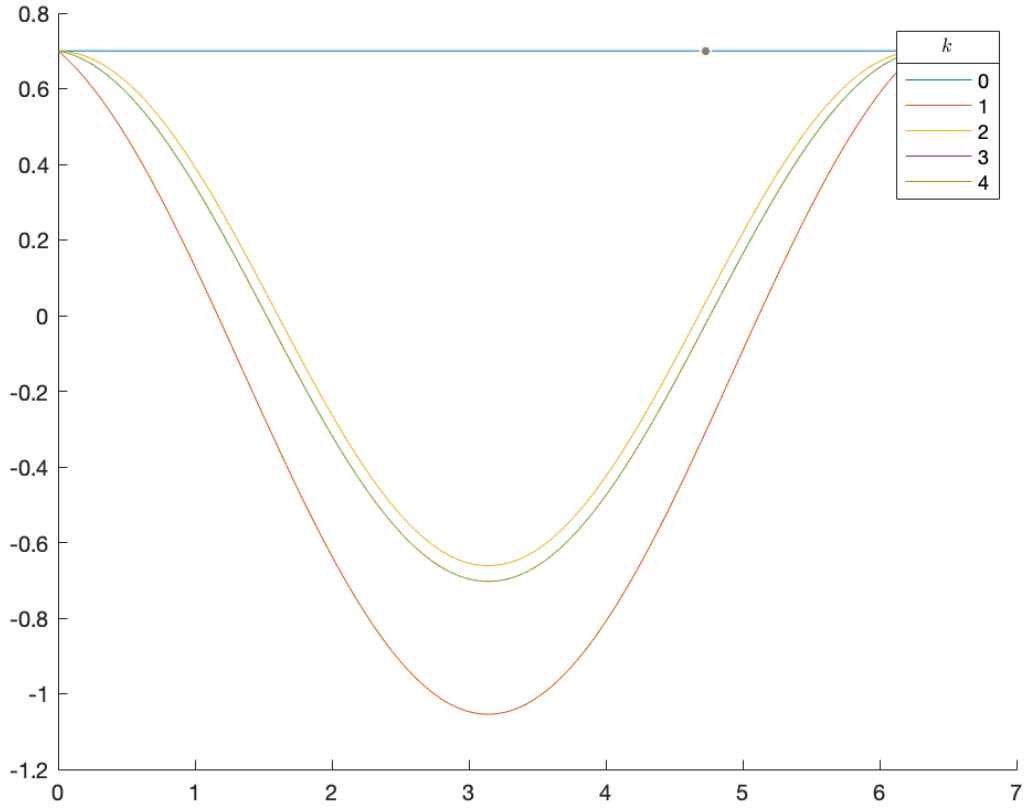
Code:

```
function [x,U, Hist] = NLhdbvp(m,f0,alpha, beta, T,N)
% This function uses a centered finite difference scheme to solve
% the following Non-linear boundary value problem:
%
%   u''(x) + sin(u(x)) = 0, u(0) = \alpha, u(T) = \beta
%
% Using m + 2 point grid and the following finite difference approx.
%
%    D^2U_j = 1/h^2(U_{j - 1} -2U_j +U_{j + 1})
%

x = linspace(0, T, m+2); % Generating grid
h = T/(m + 1); % Grid spacing
j = @(x) -2 + h^2.*cos(x); % Jacobian Function
theta = zeros(m+2, 1); %Generating Initial Iterate
theta(1) = alpha;
theta(2:m+1) = f0(x(2:m+1));
theta(m+2) = beta;
Hist = zeros(m+2, N+1);% Generate Hist
Hist(:, 1) = theta;
for i = 1:N
    % Computing g
    g = 1/h^2.* ...
        (theta(1:m) - 2.*theta(2:m+1) + theta(3:m+2)) + ...
```

```
        sin(theta(2:m+1));
    % Generating Jacobian
    J = 1/h^2.*( ...
        diag(j(theta(2:m+1))) + ...
        diag(ones(m - 1, 1), 1) + ...
        diag(ones(m - 1, 1), -1));
    % Newton Step
    dtheta = J\(-1.*g);
    % Newton Update
    theta(2:m+1) = theta(2:m+1) + dtheta;
    % Update Hist
    Hist(:, i+1) = theta;
end
U = theta;
```

Generating figure 2.4(b) from the text using initial iterate $\theta_i^{[0]} = .7$, $T = 2\pi$ and, $\alpha = \beta = .7$

Figure 3: Figure 2.4b from the text. Note k denotes iterate.



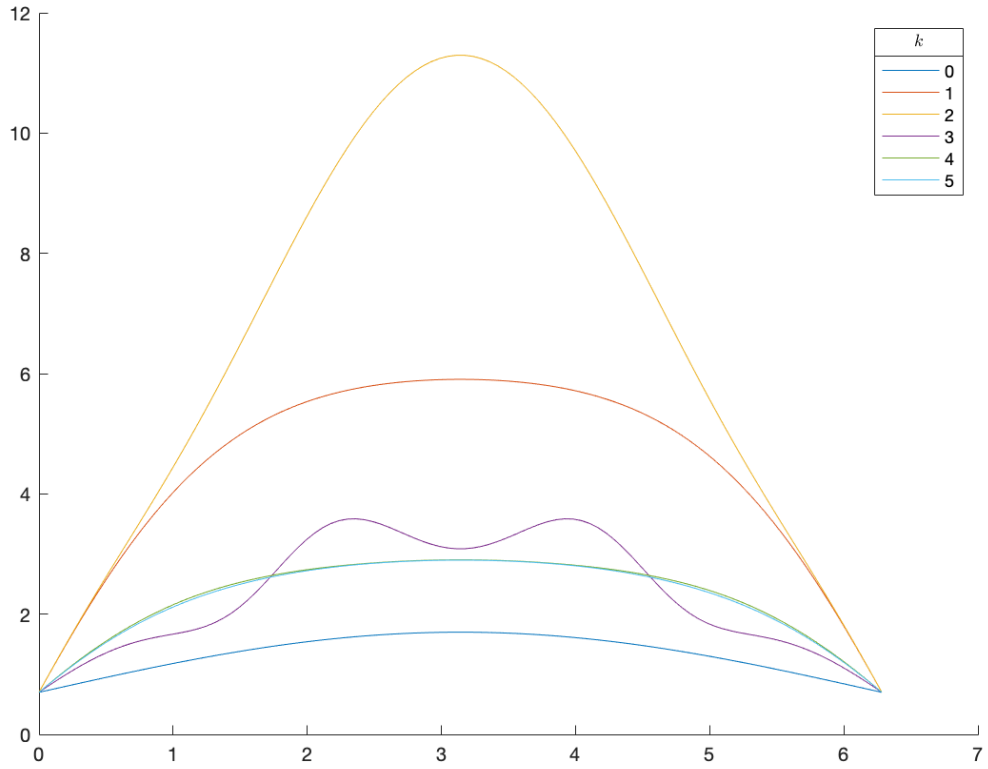
Console:

```
>> f0 = @(x) .7
>> [x,U, Hist] = NLhdbvp(100,f0,.7, .7, 2*pi,10);
>> hold on
```

```
>> for i = 1:5
     plot(x, Hist(:, i));
   end

>> lgd = legend('0', '1', '2', '3', '4');
title(lgd, '$$k$$', 'interpreter','latex')
```

Generating figure 2.5 from the text using initial iterate $\theta_i^{[0]} = .7 + \sin(t_i/2)$, $T = 2\pi$ and, $\alpha = \beta = .7$

Figure 4: Figure 2.5 from the text. Note k denotes iterate.



Console:

```
>> f0 = @(x) .7 + sin(x./2)
>> [x,U, Hist] = NLhdbvp(100,f0,.7, .7, 2*pi,10);
>> hold on
>> for i = 1:6
     plot(x, Hist(:, i));
   end

>> lgd = legend('0', '1', '2', '3', '4', '5');
title(lgd, '$$k$$', 'interpreter','latex')
```

Problem P20: Recall the ODE BVPs from P11 on Assignment #2;

$$u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \qquad u(x_L) = \alpha, \qquad u(x_R) = \beta.$$

(a) Consider the case $x_L = 0$, $x_R = 1$, $\alpha = 1$, $\beta = 0$, $p = -20$, $q = 0$, and $f(x) = 0$. Confirm that an exact solution to this problem is,

$$u(x) = 1 - \frac{1 - e^{20x}}{1 - e^{20}}.$$

Is it the only solution?

Solution:
By substitution our ODE BVP becomes,

$$u''(x) - 20u'(x) = 0, \qquad u(0) = 1, \qquad u(1) = 0.$$

Constructing the characteristic equation we get,

$$r^2 - 20r = 0$$

Which has roots $r = 0, 20$, so we get the general solution,

$$u(x) = c_1 + c_2 e^{20x}.$$

Our boundary values give the following system,

$$\begin{bmatrix} 1 & 1 \\ 1 & e^{20} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Which has a single solution $c_1 = 1 - \frac{1}{1-e^{20}}$, $c_2 = -\frac{1}{1-e^{20}}$. So by substitution we conclude that,

$$u(x) = 1 - \frac{1}{1 - e^{20}} - \frac{e^{20x}}{1 - e^{20}} = 1 - \frac{1 - e^{20x}}{1 - e^{20}}.$$

(b) Solve the problem in part (a) numerically using centered finite differences, $h = 1/(m+1)$ equal spacing, and $m = 3, 5, 10, 20, 50, 200, 1000$ interior points. Put all these numerical solutions, and the exact solution from (a), on one figure, with decent labeling.

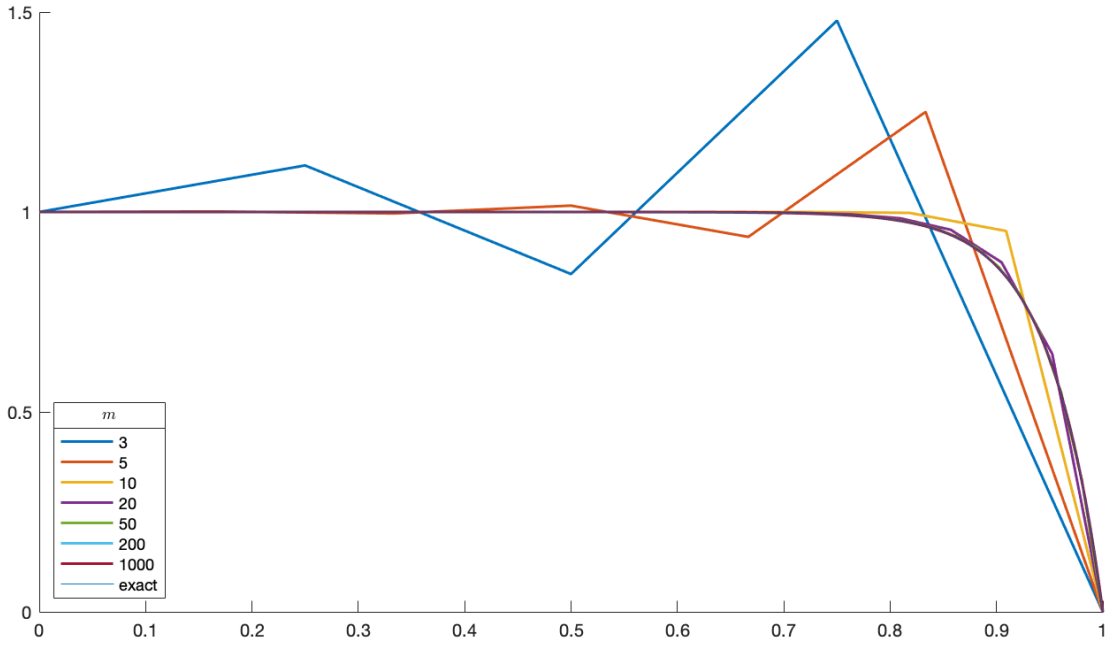
Solution:
Using our solution from P11 on Assignment #2 we can implement a code which solves this problem using centered finite difference approximations for u' and u''.

Code:

```
function [x,U] = secondorderbvp(m, xL, xR, p, q, f, alpha, beta)
% This function uses a centered finite difference scheme to solve
% the following boundary value problem:
%
%   u''(x) + p(x)u'(x) + q(x)u(x) = f(x), u(x_L) = \alpha, u(x_R) = \beta
%
% Using m + 2 point grid and the following finite difference approx.
%
%    D^2U_j = 1/h^2(U_{j - 1} -2U_j +U_{j + 1})
%
%    D_0U_j = 1/2h(U_{j + 1} -U_{j - 1})
%

x = linspace(xL, xR, m+2); % Generating grid
h = (xR - xL)/(m + 1); % Grid spacing

% Generating A
A = (1/h^2).*( ...
    diag(q(x(2:m+1)).*h^2 - 2) + ...
    diag((1 - (p(x(3:m+1)).*h)./2), -1) + ...
    diag((1 + (p(x(2:m)).*h)./2), 1) ...
    );
% Generating F
F = f(x(2:m+1))';
F(1) = F(1) - ((1/h^2).*(1 - (p(x(2))*h)/2)*alpha);
F(m) = F(m) - ((1/h^2).*(1 + (p(x(m+1))*h)/2)*beta);

% Solving U
U = A\F;
U = [alpha; U; beta];



end
```

Running the code for the given boundary values, source functions, and grid refinements we get the following figure,

Figure 5: Centered FD with Grid Refinements m.



Console:

```
m = [3, 5, 10, 20, 50, 200, 1000];

xL = 0;
xR = 1;
p = @(x) -20.*ones(size(x));
q = @(x) 0.*ones(size(x));
f = @(x) 0.*ones(size(x));
alpha = 1;
beta = 0;
exact = @(x)  1 - (1 - exp(20.*x))./(1 - exp(20));
hold on
for i = 1:7
    [x,U] = secondorderbvp(m(i), xL, xR, p, q, f, alpha, beta);
    plot(x, U, 'LineWidth',1.5)
end
plot(x, exact(x));
lgd = legend('3', '5', '10', '20', '50', ...
    '200', '1000', 'exact', ...
    'Location','southwest');

title(lgd, '$$m$$', 'interpreter','latex')
```

(c) Observe that the solutions for small values of m are poor but the high m solutions all basically agree. Why do you think that small m values are problematic here, though they are not for the problem solved in section 2.4? Write a few sentences, perhaps based on a bit of research into Section 2.17.

Solution:
This behavior where small m values are a problem seems to be cause by a rapidly changing solution over a very small interval. Naturally increasing the number of sample points in order to capture that change seems to fix the problem. Our text describes this as a singular perturbation problem. In this second order case we have a problem of the form $-\kappa u''(x) + au'(x) = f(x)$, noting it is equivalent to $\epsilon u''(x) - u'(x) = f(x)$ where $\epsilon = \kappa/a$. When a is sufficiently large relative to κ, ϵ approaches 0 and the problem can be thought of as a small perturbation to the first order $-u'(x) = f(x)$. In our problem this happens near the $u(1) = 0$ boundary. Here the derivatives of $u(x)$ are very large and since our centered FD schemes for u' and u'' are proportional to $h^2u'''(x)$ and $h^2u''''(x)$ respectively. Without sufficiently small h we can expect the error to be very large.

Proeblem P21: (a) Based on the ideas in sections 3.1-3.3, namely centered finite differences, write a matlab code that solves the Poisson equation on the unit square $0 \leq x \leq 1$, $0 \leq 1$, with zero Dirichlet boundary conditions. Allow the user to set the function $f = f(x, y)$.

Solution:
Slightly modifying heat.m we solve this problem with Dirichlet conditions instead.

Code:

```
function [xx,yy,UU] = heat2d(m,fsource)
% HEAT2D  Solve Poisson equation on the unit square [0,1]^2:
%   u_xx + u_yy = f(x,y)
% with Dirichlet conditions u=0 on the x=0, x=1, and y=1 sides, but
% u_y=0 on the y=0 side.  The grid has local (left fig.) indices,
% while the unknowns U_k = * have global (right fig.) indices as
% follows in the m = 3 case.  Note the Dirichlet boundary is
% indicated by a zero:
%
%   y=1  j=4  0   0   0   0   0          0   0   0   0   0
%        j=3  0   *   *   *   0          0   7   8   9   0
%        j=2  0   *   *   *   0          0   4   5   6   0
%        j=1  0   *   *   *   0          0   1   2   3   0
%   y=0  j=0  0   0   0   0   0          0   0   0   0   0
%            i=0 i=1 i=2 i=3 i=4
%            x=0             x=1            k values
%
% Note h = 1/(m+1) and (x_i,y_j) = (i h,j h) is grid location.
% Usage:
%    >> [xx,yy,UU] = heat2d(m,fsource)
```

```
% Example:  See TESTHEAT2D.

h = 1 / (m+1);
n = m * m;                          % number of unknowns
A = sparse(n,n);
F = zeros(n,1);
%KK = zeros(size(xx));                % for debugging: store global indices
for j = 1:m
    for i = 1:m
        k = LG(m,i,j);               % at (x_i,y_j) --> U_k is unknown here
        %KK(i+1,j+1) = k;            % for debugging
        F(k) = fsource(i*h,j*h);
        % fill kth row of A
        A(k,k) = -4;
        if i > 1
            A(k,LG(m,i-1,j)) = 1;    % west neighbor
        end
        if i < m
            A(k,LG(m,i+1,j)) = 1;    % east neighbor
        end
        if j > 1
            A(k,LG(m,i,j-1)) = 1;    % south neighbor
        end
        if j < m
            A(k,LG(m,i,j+1)) = 1;    % north neighbor (general)
        end
    end
end
A = (1/h^2) * A;
U = A \ F;                           % solution as a column vector

% for debugging: to check matrix structure
spy(A)
%full(A)
% for debugging: to check grid ordering, uncomment *all* "KK" lines
%KK, xx, yy

% insert solution into grid positions, in a form suitable
% for plotting
[xx,yy] = ndgrid(0:h:1,0:h:1);       % grid locations
UU = zeros(size(xx));                % includes Dirichlet values
UU(2:m+1,2:m+1) = reshape(U,m,m);  % insert
end % heat2d

    function k = LG(m,i,j)
    % LG  Local-to-Global index function.  Compute global
    % index k for unknown U_k, from local indices of (x_i,y_j).
    k = (m) * (j-1) + i;
    end % LG
```

(b) Find an appropriate nonzero exact solution that allows you to verify that the code converges at the theoretically-exprected rate. That is, show that $\|E^h\|_2 \to 0$, as $h \to 0$, at the rate $O(h^2)$ if $\Delta x = \Delta y = h = 1/(m+1)$.

Solution:
Consider the following function which satisfies the Dirichlet boundary conditions,

$$u(x) = x(1-x)sin(\pi x).$$

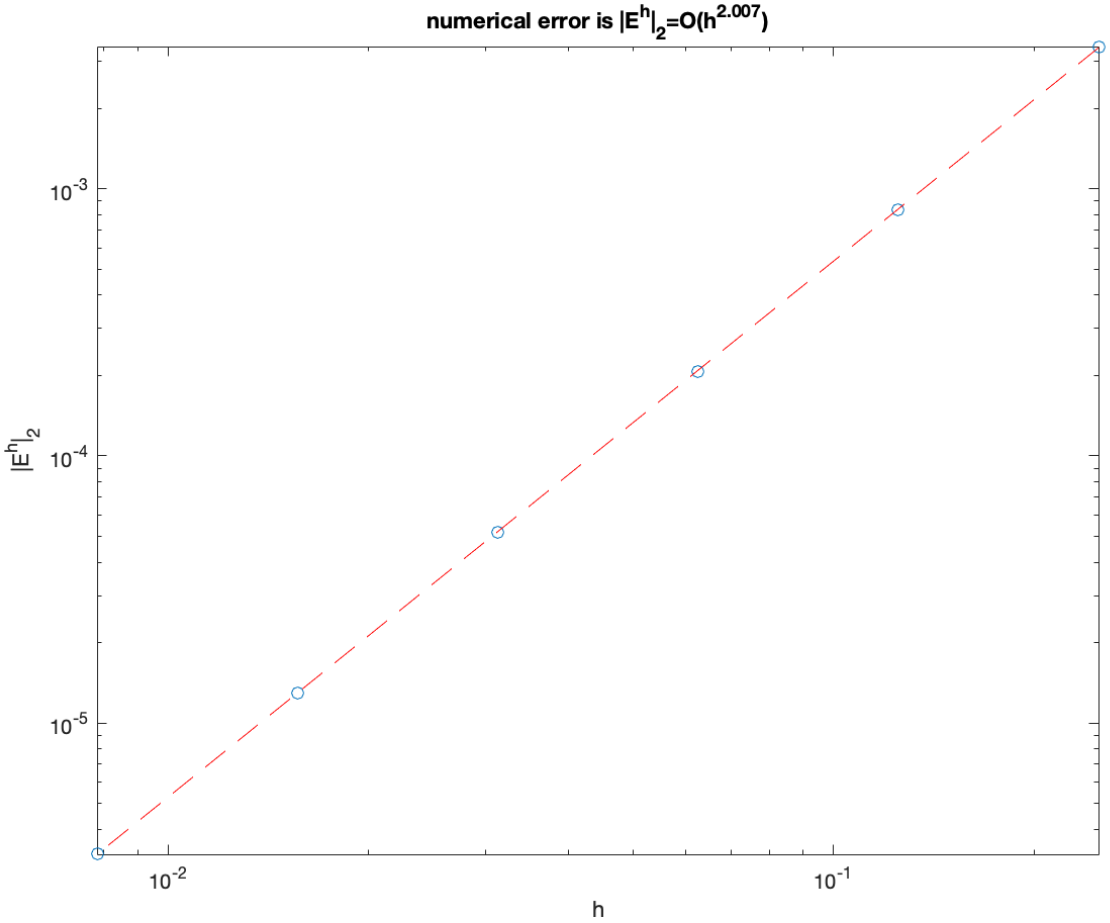
Solving for the second partial derivatives found in the poisson equation we get,

$$u_{xx} = -2\sin(\pi y),$$

$$u_{yy} = -\pi^2 x \sin(\pi y)(1-x).$$

Using the testheat2d.m code we can quickly use grid refinements to check convergence via the discrete L^2 Norm. Doing so we find that the code converges converges to the theoretical rate of $O(h^2)$.

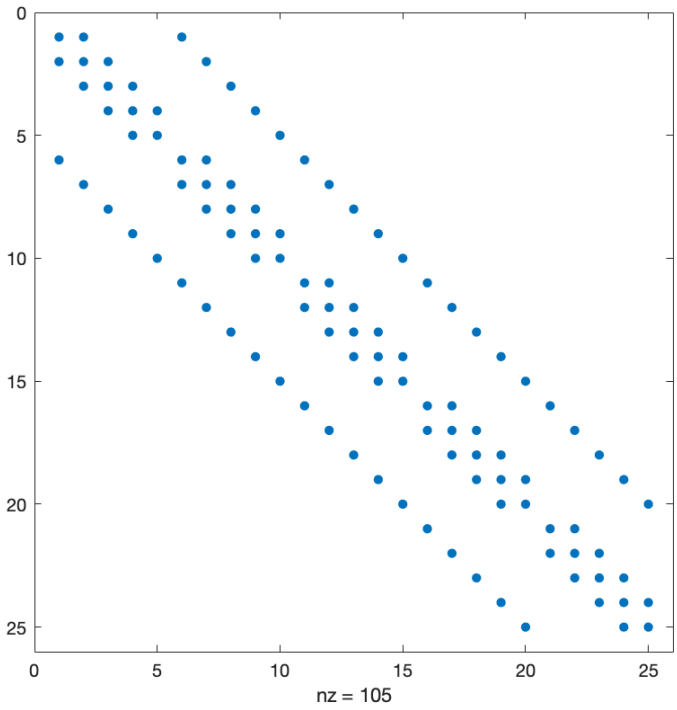
Figure 6: Convergence Plot via Discrete L^2 Norm



(c) Use Matlab's spy, or similar, to show the sparsity pattern of the matrix A^h for $m = 5$. Also confirm in that case that the matrix has the form shown by equation (3.12).

Solution:
Running our modified heat.m with poisson equation defined in the previous part with $m = 5$ we get the following spy plot,

Figure 7: Spy Plot $m = 5$.



This is as expected, and follows the tridiagonal block structure described in equation 3.12 of the text.