Table of Contents


```
clear all
addpath ../fdmbook
```

2.2: Green's Function with Neumann and Drichlet Boundary Conditions

Equation 2.54

```
h = .25;
A = 1/h^2*[ -h h 0 0 0; 1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 0
 h^2];

% Green's function to get A inverse
m = 3;
x = 0:1/(m+1):1;

for col = 0:m+1
    Ainv(:, col+1) = getG2_2(col, m, h, x);
end
Ainv
```

```
Ainv =

   -1.0000   -0.2500   -0.1875   -0.1250    1.0000
   -0.7500   -0.1875   -0.1875   -0.1250    1.0000
   -0.5000   -0.1250   -0.1250   -0.1250    1.0000
   -0.2500   -0.0625   -0.0625   -0.0625    1.0000
         0         0         0         0    1.0000
```

2.3: Solvability Condition for Nuemann Problem

```
clear all
% Equation 2.58
h = .25;
A =  1/h^2*[ -h h 0 0 0; 1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 h -
h];
x = null(A')
f = sum((A*x).*[h/2 h h h h/2]') %??
```

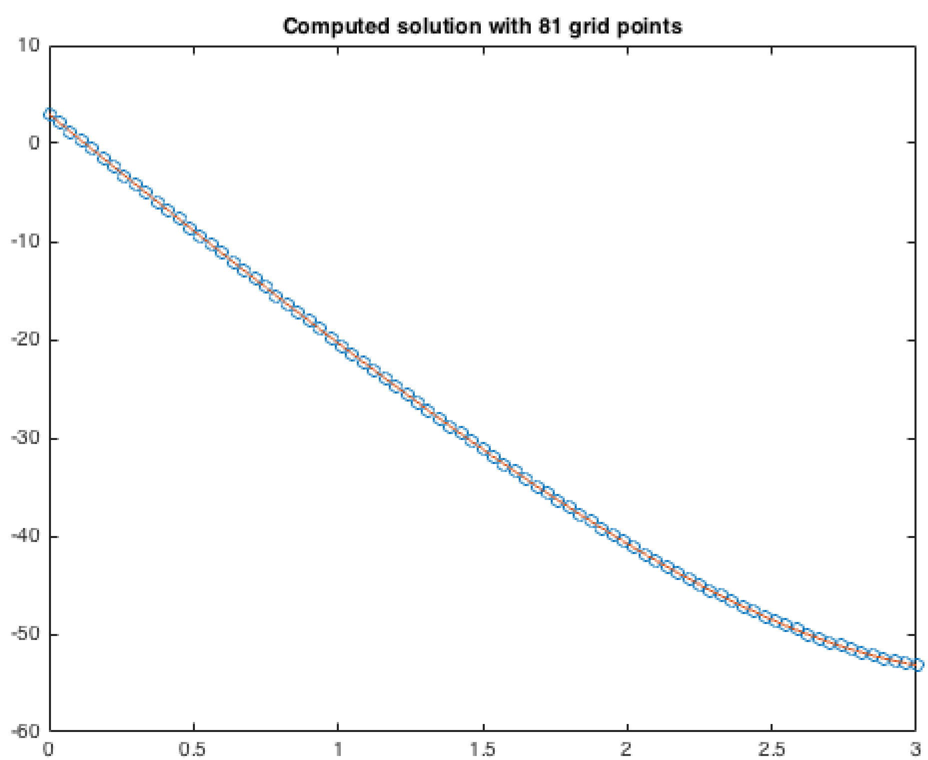
```
x =

    0.6761
    0.1690
    0.1690
    0.1690
    0.6761


f =

    3.5496
```

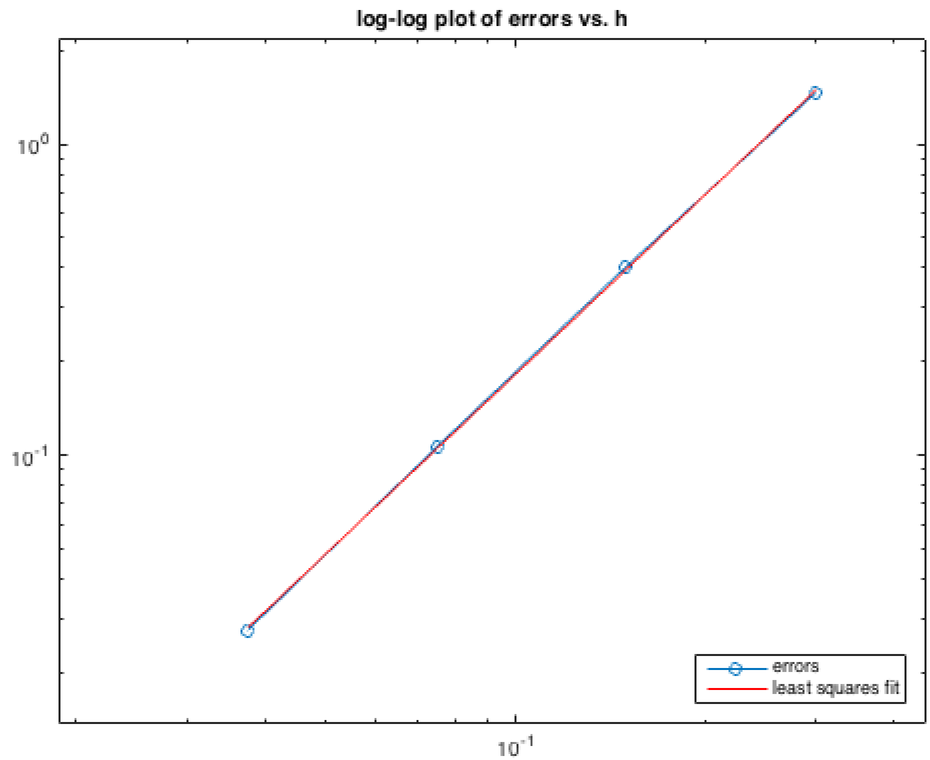
2.4: Modifying BVP code

```
clear all
% Part A
bvp_2_2_4
% Part B
bvp_4_2_4
```

```
Error with 11 points is 1.74886e+00

Error with 21 points is 4.80811e-01

Error with 41 points is 1.26086e-01

Error with 81 points is 3.22858e-02

       h         error        ratio       observed order
   0.30000    1.74886e+00       NaN              NaN
   0.15000    4.80811e-01    3.63732          1.86288
   0.07500    1.26086e-01    3.81336          1.93106
   0.03750    3.22858e-02    3.90531          1.96544


Least squares fit gives E(h) = 18.0049 * h^1.92092


Error with 11 points is 1.46048e+00

Error with 21 points is 4.04732e-01

Error with 41 points is 1.06847e-01

Error with 81 points is 2.74643e-02

       h         error        ratio       observed order
   0.30000    1.46048e+00       NaN              NaN
   0.15000    4.04732e-01    3.60850          1.85140
```

```
    0.07500   1.06847e-01   3.78798          1.92143
    0.03750   2.74643e-02   3.89038          1.95991

Least squares fit gives E(h) = 14.8888 * h^1.91196
```


Computed solution with 81 grid points
10
0
-10
-20
-30
-40
-50
-60
0
0.5
1
1.5
2
2.5
3



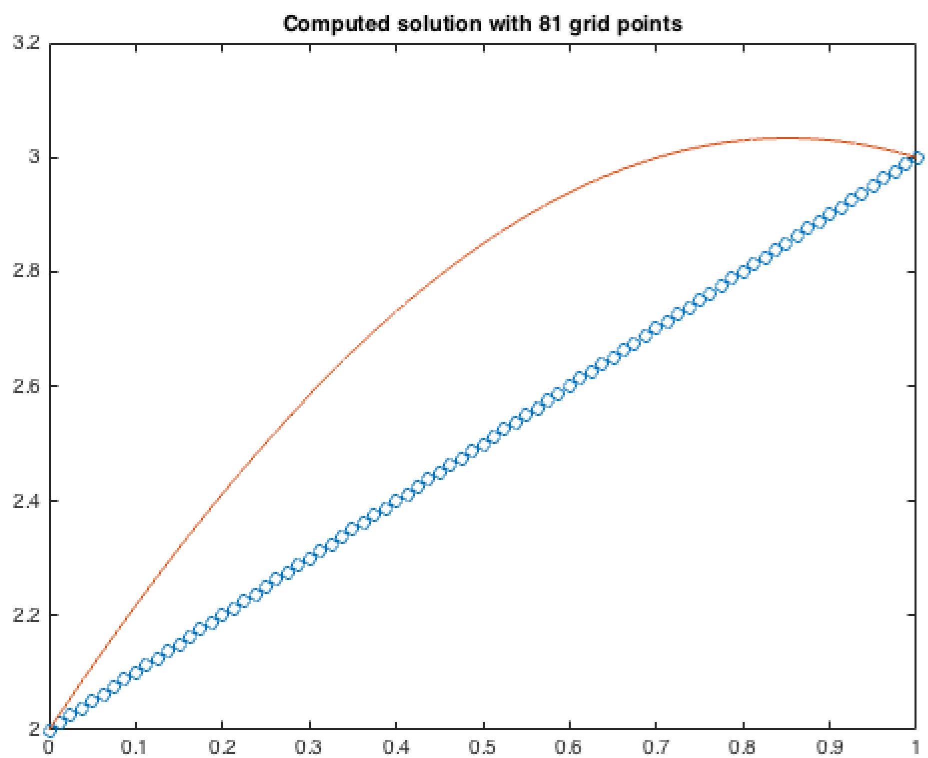
2.6 Ill Posed BVP

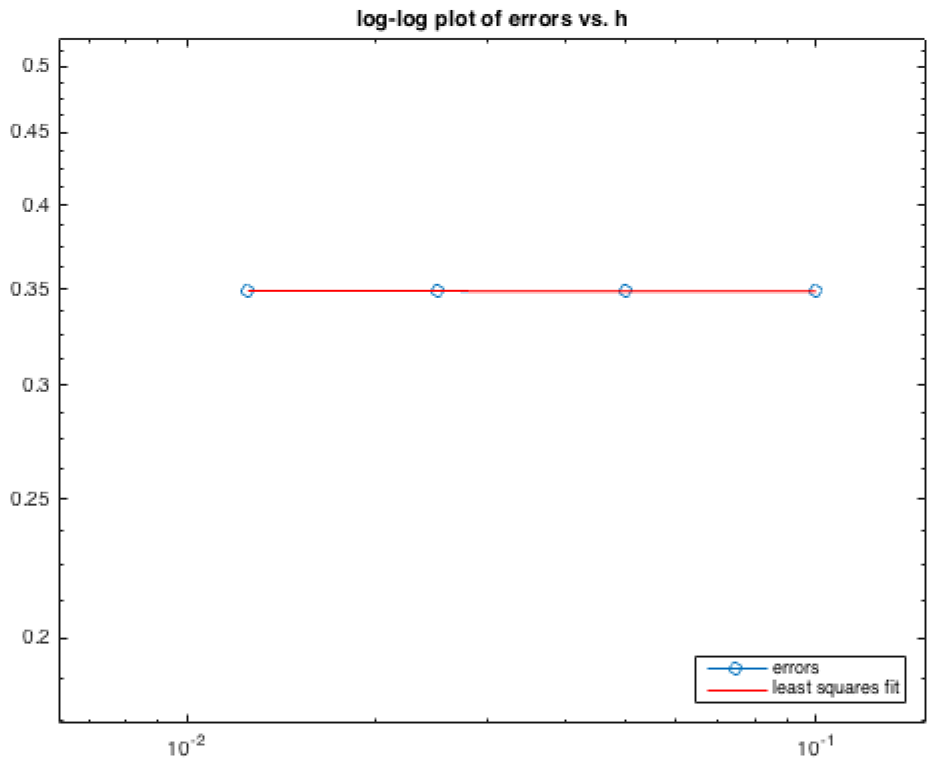
```
clear all
ax = 0;
bx = 1;
alpha = 2;    % Dirichlet boundary condition at ax
beta = 3;      % Dirichlet boundary condtion at bx
bvp_2_2_6
```

```
utrue =

    @(x)X(1)*cos(x)+X(2)*sin(x)

The eigen values of A are

eigensA =

 -390.2113
 -361.8034
 -317.5571
 -261.8034
 -200.0000
 -138.1966

The 2 norm of A inverse is
```

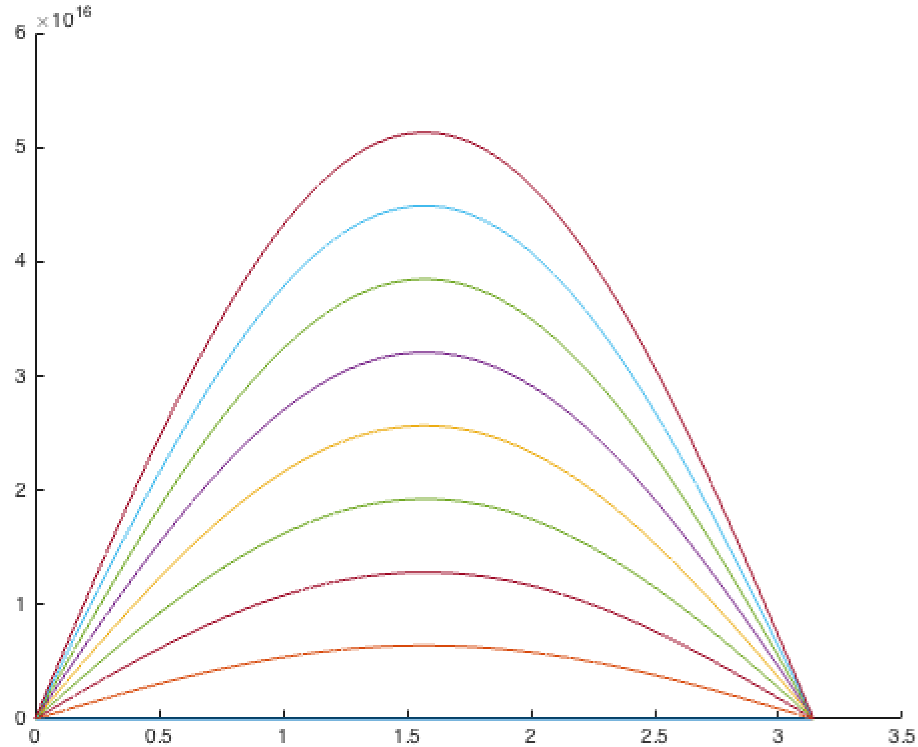
```
twoNormA =

    2.3468


Error with 11 points is 3.48735e-01
The eigen values of A are

eigensA =

   1.0e+03 *

   -1.5902
   -1.5608
   -1.5128
   -1.4472
   -1.3657
   -1.2702

The 2 norm of A inverse is

twoNormA =

    3.2416


Error with 21 points is 3.48735e-01
The eigen values of A are

eigensA =

   1.0e+03 *

   -6.3901
   -6.3606
   -6.3116
   -6.2434
   -6.1564
   -6.0512

The 2 norm of A inverse is

twoNormA =

    4.5286


Error with 41 points is 3.48915e-01
The eigen values of A are

eigensA =

   1.0e+04 *
```

```
   -2.5590
   -2.5561
   -2.5511
   -2.5442
   -2.5354
   -2.5246

The 2 norm of A inverse is

twoNormA =

    6.3646


Error with 81 points is 3.49048e-01

      h         error       ratio       observed order
   0.10000   3.48735e-01       NaN              NaN
   0.05000   3.48735e-01   1.00000          0.00000
   0.02500   3.48915e-01   0.99948         -0.00074
   0.01250   3.49048e-01   0.99962         -0.00055


Least squares fit gives E(h) = 0.348318 * h^-0.00046339
```


Computed solution with 81 grid points
3.2
3
2.8
2.6
2.4
2.2
2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



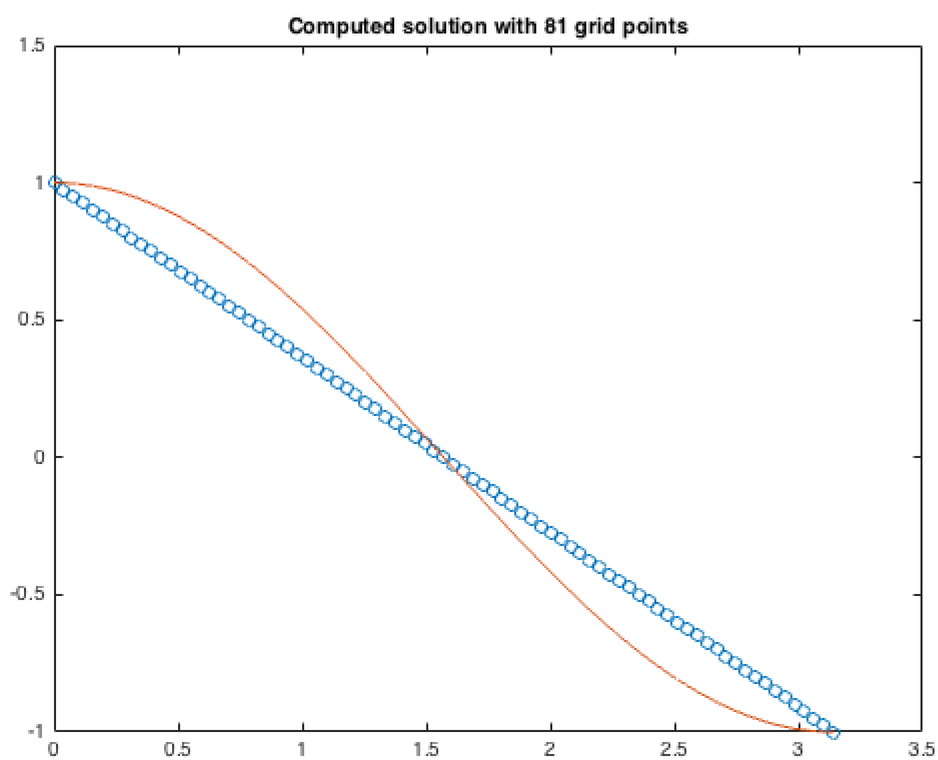

```
clear all
ax = 0;
bx = pi;
count = 1;
check = [0 pi/4 pi/2 3*pi/4 pi];
x = linspace(0, pi);
syms A B
for alpha = check
    for beta = check
        X = bvp_check(ax, bx, alpha, beta, A, B);
        if X(1)~= 0 && X(2) ~= 0
            keep(:, count) = [alpha; beta];
            count = count +1;
            sln(:, count) = X(1)*cos(x) + X(2)*sin(x);
        end
    end
end
figure (1)
hold on
for i = 1:count
    plot(x, sln(:, i))
end
```

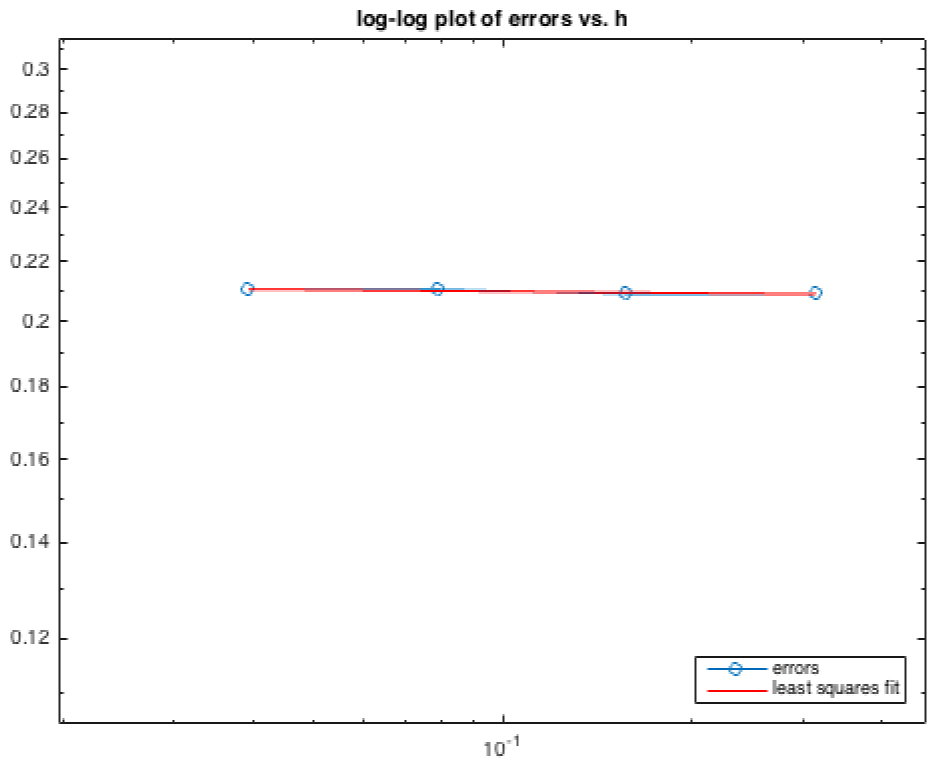



```
clear all
ax = 0;
bx = pi;
alpha = 1;    % Dirichlet boundary condition at ax
beta = -1;      % Dirichlet boundary condtion at bx
bvp_2_2_6


utrue =

    @(x)X(1)*cos(x)+X(2)*sin(x)

The eigen values of A are

eigensA =

  -39.5367
  -36.6583
  -32.1753
  -26.5262
  -20.2642
  -14.0022

The 2 norm of A inverse is

twoNormA =
```

```
    2.5044


Error with 11 points is 2.09017e-01
The eigen values of A are

eigensA =

 -161.1159
 -158.1467
 -153.2792
 -146.6334
 -138.3729
 -128.7010

The 2 norm of A inverse is

twoNormA =

    3.3599


Error with 21 points is 2.09017e-01
The eigen values of A are

eigensA =

 -647.4561
 -644.4638
 -639.4971
 -632.5867
 -623.7752
 -613.1169

The 2 norm of A inverse is

twoNormA =

    4.6152


Error with 41 points is 2.10406e-01
The eigen values of A are

eigensA =

   1.0e+03 *

   -2.5928
   -2.5898
   -2.5848
   -2.5779
   -2.5689
```

```
   -2.5580

The 2 norm of A inverse is

twoNormA =

    6.4269


Error with 81 points is 2.10406e-01

      h         error       ratio       observed order
   0.31416   2.09017e-01       NaN             NaN
   0.15708   2.09017e-01   1.00000         0.00000
   0.07854   2.10406e-01   0.99340        -0.00956
   0.03927   2.10406e-01   1.00000        -0.00000


Least squares fit gives E(h) = 0.207956 * h^-0.00382215
```


Computed solution with 81 grid points
1.5
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5

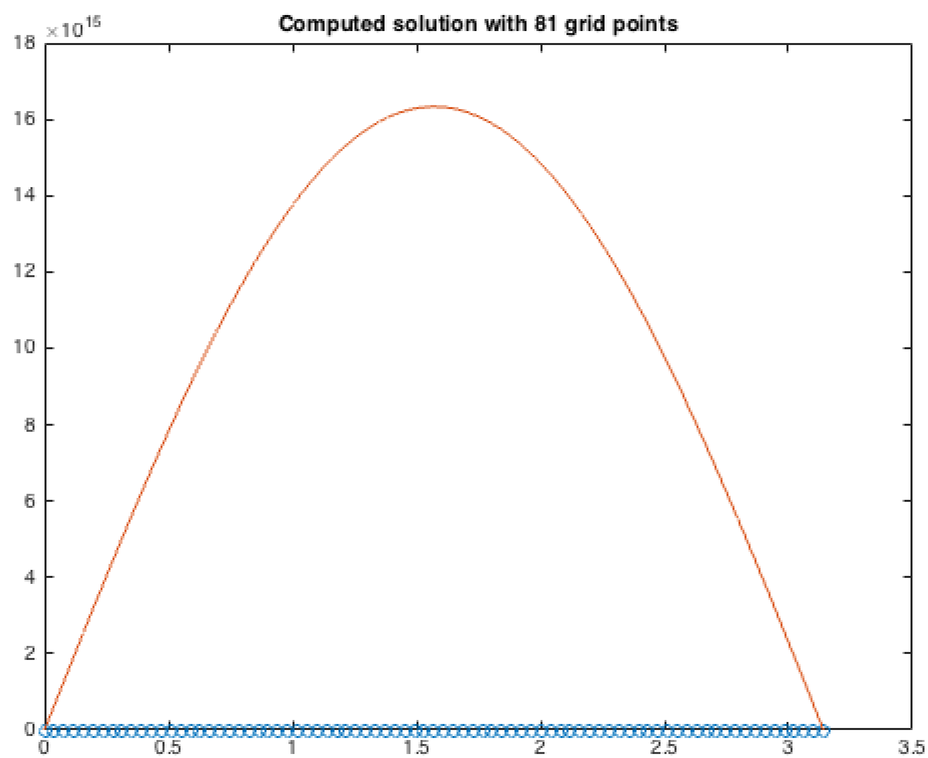


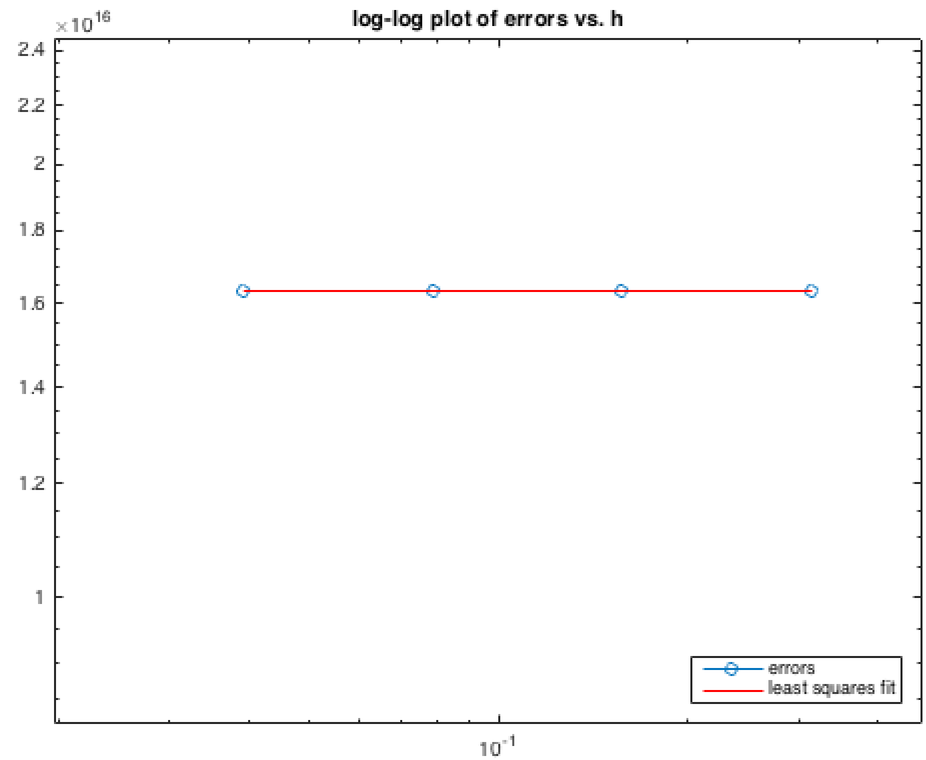

```
clear all
ax = 0;
bx = pi;
alpha = 1;    % Dirichlet boundary condition at ax
beta = 1;      % Dirichlet boundary condtion at bx
bvp_2_2_6
```

```
utrue =

    @(x)X(1)*cos(x)+X(2)*sin(x)

The eigen values of A are

eigensA =

  -39.5367
  -36.6583
  -32.1753
  -26.5262
  -20.2642
  -14.0022

The 2 norm of A inverse is

twoNormA =
```

```
    2.5044


Error with 11 points is 1.63312e+16
The eigen values of A are

eigensA =

 -161.1159
 -158.1467
 -153.2792
 -146.6334
 -138.3729
 -128.7010

The 2 norm of A inverse is

twoNormA =

    3.3599


Error with 21 points is 1.63312e+16
The eigen values of A are

eigensA =

 -647.4561
 -644.4638
 -639.4971
 -632.5867
 -623.7752
 -613.1169

The 2 norm of A inverse is

twoNormA =

    4.6152


Error with 41 points is 1.63312e+16
The eigen values of A are

eigensA =

   1.0e+03 *

   -2.5928
   -2.5898
   -2.5848
   -2.5779
   -2.5689
```

```
   -2.5580

The 2 norm of A inverse is

twoNormA =

    6.4269


Error with 81 points is 1.63312e+16

      h         error        ratio       observed order
   0.31416   1.63312e+16       NaN              NaN
   0.15708   1.63312e+16   1.00000          0.00000
   0.07854   1.63312e+16   1.00000          0.00000
   0.03927   1.63312e+16   1.00000          0.00000


Least squares fit gives E(h) = 1.63312e+16 * h^6.09851e-15
```


Computed solution with 81 grid points
$\times 10^{15}$
18
16
14
12
10
8
6
4
2
0
0
0.5
1
1.5
2
2.5
3
3.5



Published with MATLAB® R2015b

```
function [ cj ] = getG2_2( col, m, h, x)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

if col == 0
    cj = x-1;
elseif col == m+1
    cj = ones(length(x), 1);
else
    for i = 1:length(x)
        if i<=col
            cj(i) = h*(x(col)-1);
        else
            cj(i) = h*(x(i)-1);
        end
    end
end

end
```

```
Not enough input arguments.

Error in getG2_2 (line 5)
if col == 0
```

Published with MATLAB® R2015b

```
%
% bvp_2.m
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% Using 3-pt differences on an arbitrary nonuniform grid.
% Should be 2nd order accurate if grid points vary smoothly, but may
% degenerate to "first order" on random or nonsmooth grids.
%
% Different BCs can be specified by changing the first and/or last
 rows of
% A and F.
%
% From  http://www.amath.washington.edu/~rjl/fdmbook/  (2007)

% ** modified
close all
ax = 0;
bx = 3;
sigma = -5;   % Dirichlet boundary condition at ax
alpha = 3;     % Neumann boundary condtion at bx

f = @(x) exp(x);  % right hand side function *modified
utrue = @(x) exp(x) + (sigma-exp(bx))*(x) + alpha - exp(ax);  % true
 soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test
 convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                 % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for convergence
 tests

  % set grid points:
  gridchoice = 'uniform';          % see xgrid.m for other choices
  x = xgrid(ax,bx,m,gridchoice);

  % set up matrix A (using sparse matrix storage):
  A = spalloc(m2,m2,3*m2);   % initialize to zero matrix

  % first row for Dirichlet BC at ax: *modified
  A(1,1:3) = fdcoeffF(0, x(1), x(1:3));
```

```
  % interior rows:
  for i=2:m1
     A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1)));
     end

  % last row for Nuemann BC at bx: *modified
  A(m2,m:m2) = fdcoeffF(1,x(m2),x(m:m2));

  % Right hand side:
  F = f(x);
  F(1) = alpha; % **modified
  F(m2) = sigma;

  % solve linear system:
  U = A\F;


  % compute error at grid points:
  uhat = utrue(x);
  err = U - uhat;
  E(jtest) = max(abs(err));
  disp(' ')
  disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

  clf
  figure(i)
  plot(x,U,'o')  % plot computed solution
  title(sprintf('Computed solution with %i grid points',m2));
  hold on

  plot(xfine,ufine)  % plot true solution
  hold off

  % pause to see this plot:
  drawnow
  %input('Hit <return> to continue ');

  end

error_table(hvals, E);   % print tables of errors and ratios
figure(2)
error_loglog(hvals, E);  % produce log-log plot of errors and least
 squares fit
```

Error with 11 points is 1.74886e+00

Error with 21 points is 4.80811e-01

Error with 41 points is 1.26086e-01

Error with 81 points is 3.22858e-02

```
    h        error       ratio       observed order
  0.30000   1.74886e+00      NaN              NaN
  0.15000   4.80811e-01   3.63732          1.86288
  0.07500   1.26086e-01   3.81336          1.93106
  0.03750   3.22858e-02   3.90531          1.96544

Least squares fit gives E(h) = 18.0049 * h^1.92092
```

```
function [ X ] = bvp_check(ax, bx, alpha, beta, A, B)

%
% bvp_2.m
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% Using 3-pt differences on an arbitrary nonuniform grid.
% Should be 2nd order accurate if grid points vary smoothly, but may
% degenerate to "first order" on random or nonsmooth grids.
%
% Different BCs can be specified by changing the first and/or last
 rows of
% A and F.
%
% From  http://www.amath.washington.edu/~rjl/fdmbook/  (2007)
%addpath ../fdmbook


f = @(x) 0;  % right hand side function ?? what to make this?
eqn1 = A-(alpha-B*sin(ax))/cos(ax) == 0;
eqn2 = B-(beta-A*cos(bx))/sin(bx) == 0;
[C, D] = equationsToMatrix([eqn1, eqn2], [A, B]);
X= linsolve(C, D);
%{
utrue = @(x) X(1)*cos(x) +X(2)*sin(x)  % true soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test
 convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                 % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for convergence
 tests

  % set grid points:
  gridchoice = 'uniform';          % see xgrid.m for other choices
  x = xgrid(ax,bx,m,gridchoice);

  % set up matrix A (using sparse matrix storage):
  A = spalloc(m2,m2,3*m2);   % initialize to zero matrix

  % first row for Dirichlet BC at ax:
```

```
  A(1,1:3) = fdcoeffF(0, x(1), x(1:3));

  % interior rows:
  for i=2:m1
     A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1)));
     end

  % last row for Dirichlet BC at bx:
  A(m2,m:m2) = fdcoeffF(0,x(m2),x(m:m2));

  % Right hand side:
  F = f(x);
  F(1) = alpha;
  F(m2) = beta;

  % solve linear system:
  U = A\F;


  % compute error at grid points:
  uhat = utrue(x);
  err = U - uhat;
  E(jtest) = max(abs(err));
  disp(' ')
  disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

  clf
  plot(x,U,'o')  % plot computed solution
  title(sprintf('Computed solution with %i grid points',m2));
  hold on
  plot(xfine,ufine)  % plot true solution
  hold off

  % pause to see this plot:
  drawnow
  input('Hit <return> to continue ');

  end

error_table(hvals, E);   % print tables of errors and ratios
figure(2)
error_loglog(hvals, E);  % produce log-log plot of errors and least
 squares fit
%}
end
```

```
Not enough input arguments.

Error in bvp_check (line 19)
eqn1 = A-(alpha-B*sin(ax))/cos(ax) == 0;
```

Published with MATLAB® R2015b

```
%
% bvp4.m
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% fourth order finite difference method for the bvp
%   u'' = f,   u'(ax)=sigma,   u(bx)=beta
% Using 5-pt differences on an arbitrary grid.
% Should be 4th order accurate if grid points vary smoothly.
%
% Different BCs can be specified by changing the first and/or last
 rows of
% A and F.
%
% From  http://www.amath.washington.edu/~rjl/fdmbook/chapter2  (2007)

% ** modified
close all
ax = 0;
bx = 3;
sigma = -5;   % Dirichlet boundary condition at ax
alpha = 3;     % Neumann boundary condtion at bx

f = @(x) exp(x);  % right hand side function *modified
utrue = @(x) exp(x) + (sigma-exp(bx))*(x) + alpha - exp(ax);  % true
 soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test
 convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                 % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for convergence
 tests

  % set grid points:
  gridchoice = 'uniform';
  x = xgrid(ax,bx,m,gridchoice);

  % set up matrix A (using sparse matrix storage):
  A = spalloc(m2,m2,5*m2);   % initialize to zero matrix

  % first row for Dirichlet BC at ax: *modified
```

```
    A(1,1:3) = fdcoeffF(0, x(1), x(1:3));
    % second row for u''(x(2))
    A(2,1:6) = fdcoeffF(2, x(2), x(1:6));

    % interior rows:
    for i=3:m
       A(i,i-2:i+2) = fdcoeffF(2, x(i), x((i-2):(i+2)));
       end

    % next to last row for u''(x(m+1))
    A(m1,m-3:m2) = fdcoeffF(2,x(m1),x(m-3:m2));
    % last row for Nuemann BC at bx: *modified
    A(m2,m:m2) = fdcoeffF(1,x(m2),x(m:m2));

    % Right hand side:
    F = f(x);
    F(1) = alpha; % **modified
    F(m2) = sigma;

    % solve linear system:
    U = A\F;


    % compute error at grid points:
    uhat = utrue(x);
    err = U - uhat;
    E(jtest) = max(abs(err));
    disp(' ')
    disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

    clf
    figure(i)
    plot(x,U,'o')  % plot computed solution
    title(sprintf('Computed solution with %i grid points',m2));
    hold on

    plot(xfine,ufine)  % plot true solution
    hold off

    % pause to see this plot:
    drawnow
    %input('Hit <return> to continue ');

    end

  error_table(hvals, E);   % print tables of errors and ratios
  figure(2)
  error_loglog(hvals, E);  % produce log-log plot of errors and least
   squares fit
```

Error with 11 points is 1.46048e+00

Error with 21 points is 4.04732e-01

```
%
% bvp_2.m
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% Using 3-pt differences on an arbitrary nonuniform grid.
% Should be 2nd order accurate if grid points vary smoothly, but may
% degenerate to "first order" on random or nonsmooth grids.
%
% Different BCs can be specified by changing the first and/or last
 rows of
% A and F.
%
% From  http://www.amath.washington.edu/~rjl/fdmbook/  (2007)
addpath ../fdmbook
close all


f = @(x) zeros(length(x), 1);  % right hand side function ?? what to
 make this?
syms A B
eqn1 = A-(alpha-B*sin(ax))/cos(ax) == 0;
eqn2 = B-(beta-A*cos(bx))/sin(bx) == 0;
[C, D] = equationsToMatrix([eqn1, eqn2], [A, B]);
X= linsolve(C, D);
utrue = @(x) X(1)*cos(x) +X(2)*sin(x)  % true soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test
 convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                 % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for convergence
 tests

  % set grid points:
  gridchoice = 'uniform';          % see xgrid.m for other choices
  x = xgrid(ax,bx,m,gridchoice);

  % set up matrix A (using sparse matrix storage):
  A = spalloc(m2,m2,3*m2);   % initialize to zero matrix

  % first row for Dirichlet BC at ax:
```

```
A(1,1:3) = fdcoeffF(0, x(1), x(1:3));

% interior rows:
for i=2:m1
   A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1)));
   end

% last row for Dirichlet BC at bx:
A(m2,m:m2) = fdcoeffF(0,x(m2),x(m:m2));
disp('The eigen values of A are')
eigensA = eigs(A)
disp('The 2 norm of A inverse is')
twoNormA = norm(full(inv(A)))
% Right hand side:
F = f(x);
F(1) = alpha;
F(m2) = beta;

% solve linear system:
U = A\F;


% compute error at grid points:
uhat = utrue(x);
err = U - uhat;
E(jtest) = max(abs(err));
disp(' ')
disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

clf
 figure(i)
plot(x,U,'o')  % plot computed solution
title(sprintf('Computed solution with %i grid points',m2));
hold on

plot(xfine,ufine)  % plot true solution
hold off

% pause to see this plot:
drawnow
%input('Hit <return> to continue ');

end

error_table(hvals, E);   % print tables of errors and ratios
figure(2)
error_loglog(hvals, E);  % produce log-log plot of errors and least
 squares fit


utrue =

    @(x)X(1)*cos(x)+X(2)*sin(x)
```

```
The eigen values of A are

eigensA =

  -39.5367
  -36.6583
  -32.1753
  -26.5262
  -20.2642
  -14.0022

The 2 norm of A inverse is

twoNormA =

    2.5044


Error with 11 points is 1.63312e+16
The eigen values of A are

eigensA =

 -161.1159
 -158.1467
 -153.2792
 -146.6334
 -138.3729
 -128.7010

The 2 norm of A inverse is

twoNormA =

    3.3599


Error with 21 points is 1.63312e+16
The eigen values of A are

eigensA =

 -647.4561
 -644.4638
 -639.4971
 -632.5867
 -623.7752
 -613.1169

The 2 norm of A inverse is

twoNormA =

    4.6152
```

```
Error with 41 points is 1.63312e+16
The eigen values of A are

eigensA =

   1.0e+03 *

   -2.5928
   -2.5898
   -2.5848
   -2.5779
   -2.5689
   -2.5580

The 2 norm of A inverse is

twoNormA =

    6.4269


Error with 81 points is 1.63312e+16

      h         error       ratio       observed order
   0.31416   1.63312e+16       NaN              NaN
   0.15708   1.63312e+16   1.00000          0.00000
   0.07854   1.63312e+16   1.00000          0.00000
   0.03927   1.63312e+16   1.00000          0.00000


Least squares fit gives E(h) = 1.63312e+16 * h^6.09851e-15
```