```
%
% 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

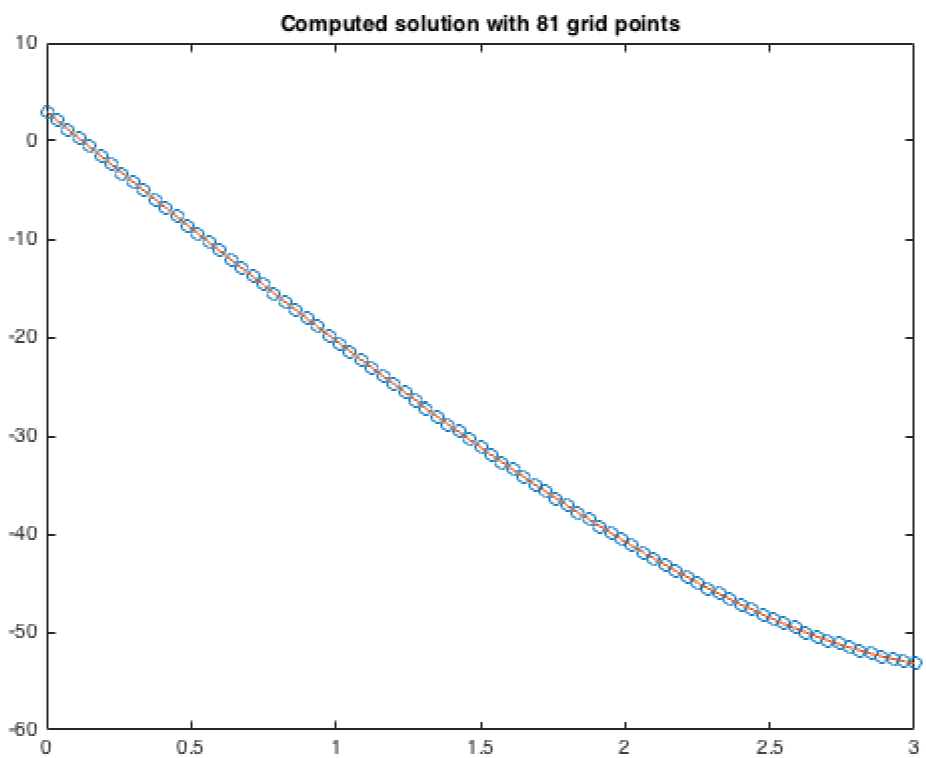
Error with 21 points is 4.04732e-01

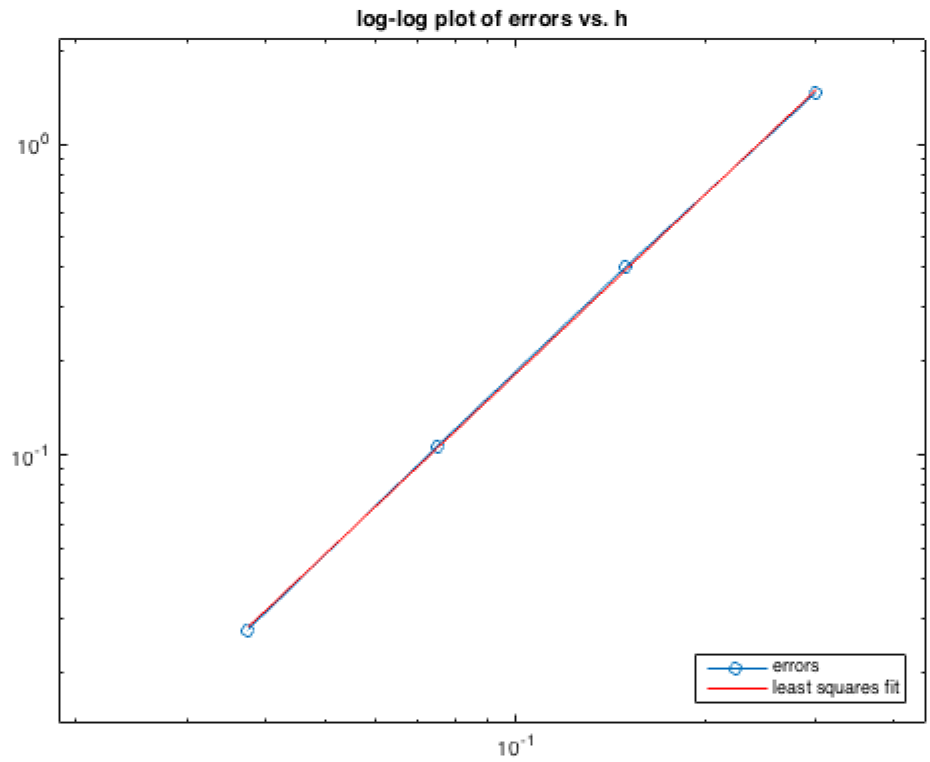
```
Error with 41 points is 1.06847e-01

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



Published with MATLAB® R2015b