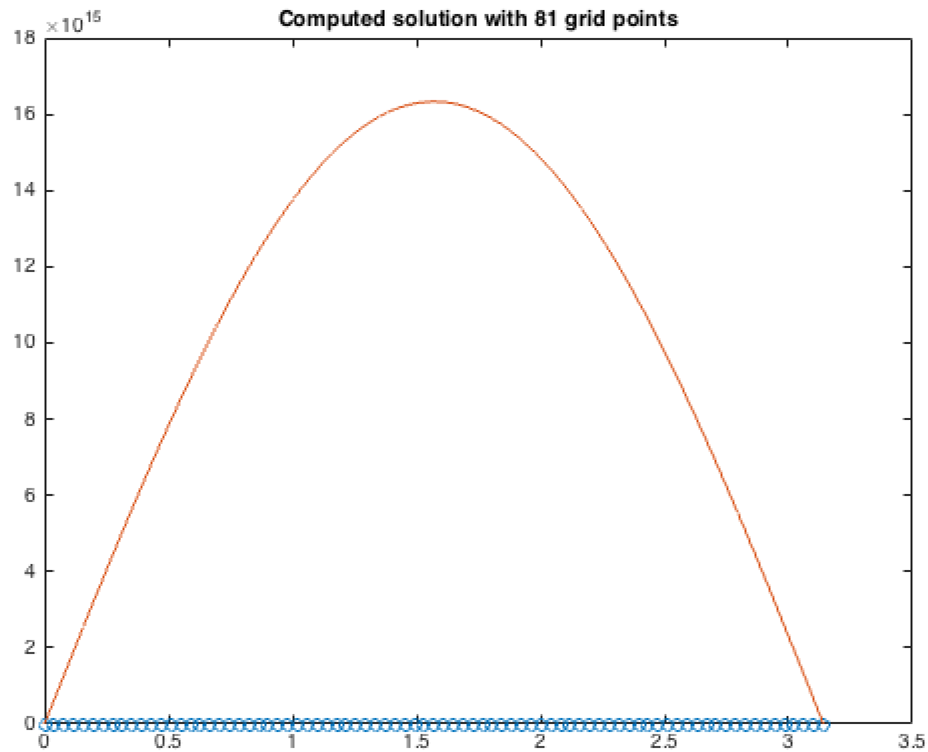
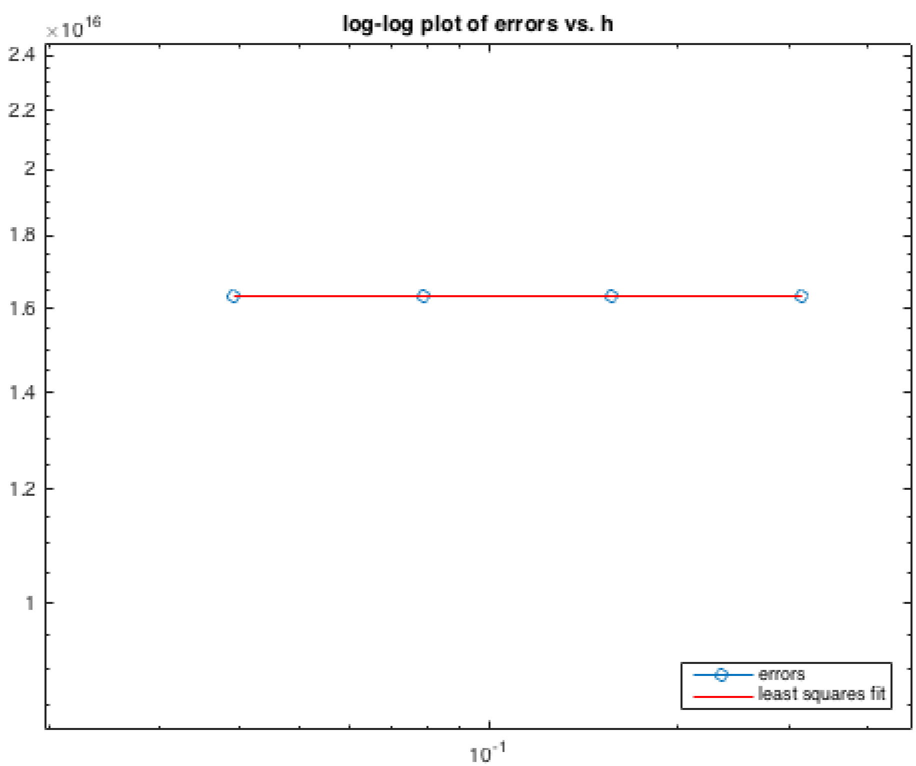
```
%
% bvp_2.m
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% Using 3-pt differences on an arbitrary nonuniform grid.
% Should be 2nd order accurate if grid points vary smoothly, but may
% degenerate to "first order" on random or nonsmooth grids.
%
% 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
```


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


log-log plot of errors vs. h
×10^16
2.4
2.2
2
1.8
1.6
1.4
1.2
1
10^-1
errors
least squares fit

Published with MATLAB® R2015b