Table of Contents

```
% 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)
```

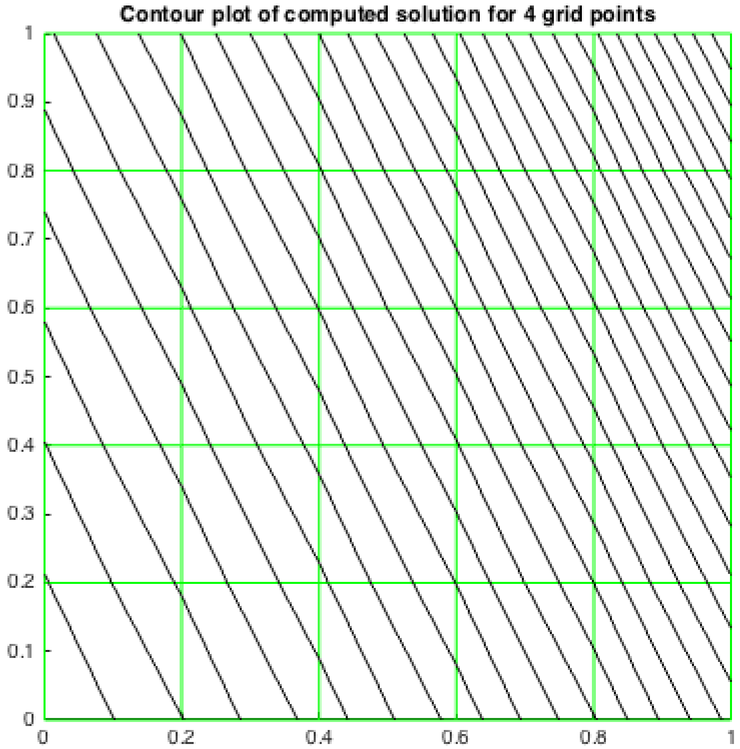
Problem 3.1.a

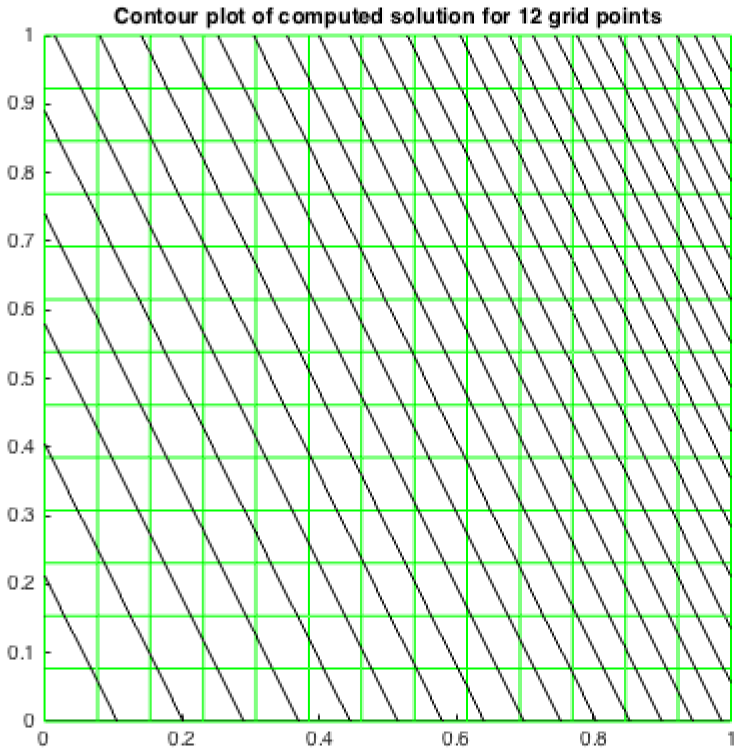
```
clear all
close all
count = 0;
fprintf('m   error\n')
for m = 4:8:48
count = count+1;
a = 0;
b = 1;
h = (b-a)/(m+1);
x = linspace(a,b,m+2);   % grid points x including boundaries
y = linspace(a,b,m+2);   % grid points y including boundaries


[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:
```

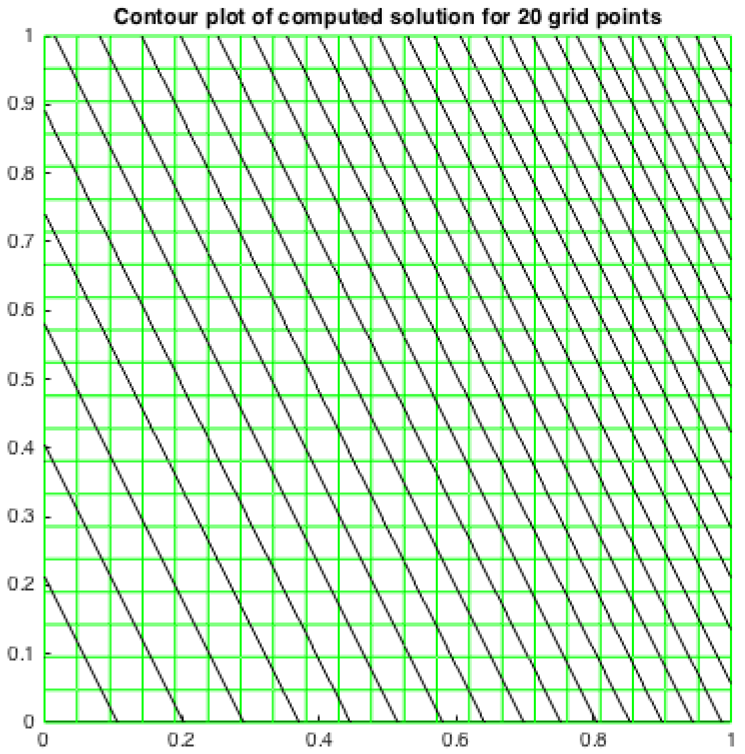
```
usoln = utrue;                % use true solution for this test problem
                              % This sets full array, but only boundary
 values
                              % are used below.  For a problem where
 utrue
                              % is not known, would have to set each
 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:
err = max(max(abs(usoln-utrue)));
%fprintf('grid size: %dx%d\n', m, m);
%fprintf('Error relative to true solution of PDE = %10.5e \n',err)
fprintf(' %d & %10.5e \\\\ \n', m, err);
% plot results:

figure(count)
hold on

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

% plot solution:
```

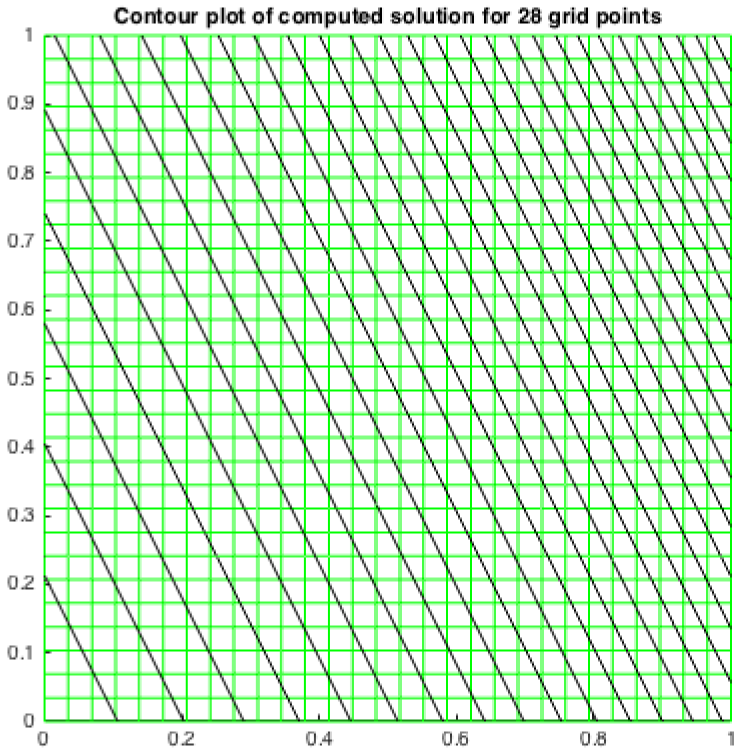
```
contour(X,Y,usoln,30,'k')

axis([a b a b])
daspect([1 1 1])
name = sprintf('Contour plot of computed solution for %d grid points',
 m);
title(name)
hold off
end
```

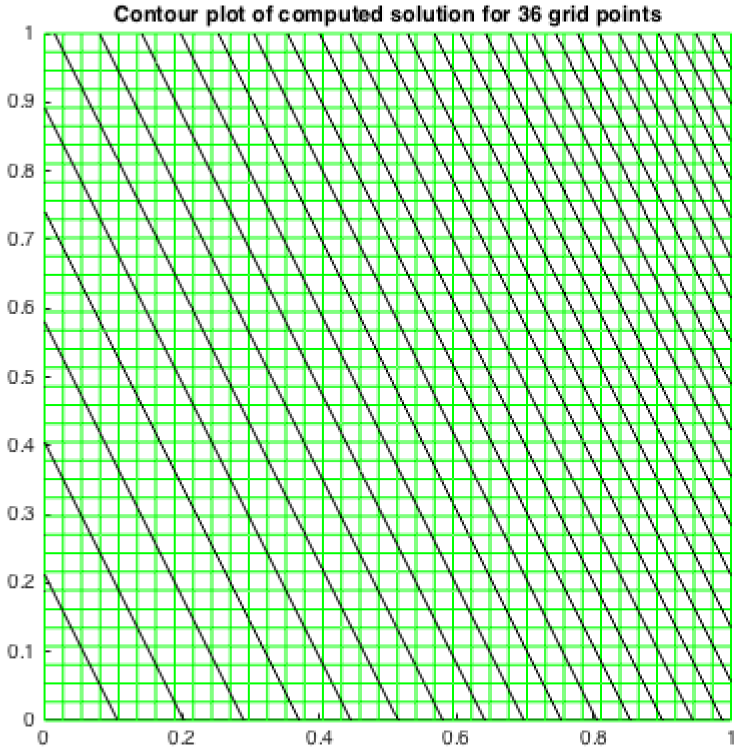
```
m   error
 4 & 5.50547e-04 \\
 12 & 8.48461e-05 \\
 20 & 3.27323e-05 \\
 28 & 1.71710e-05 \\
 36 & 1.05646e-05 \\
 44 & 7.14325e-06 \\
```

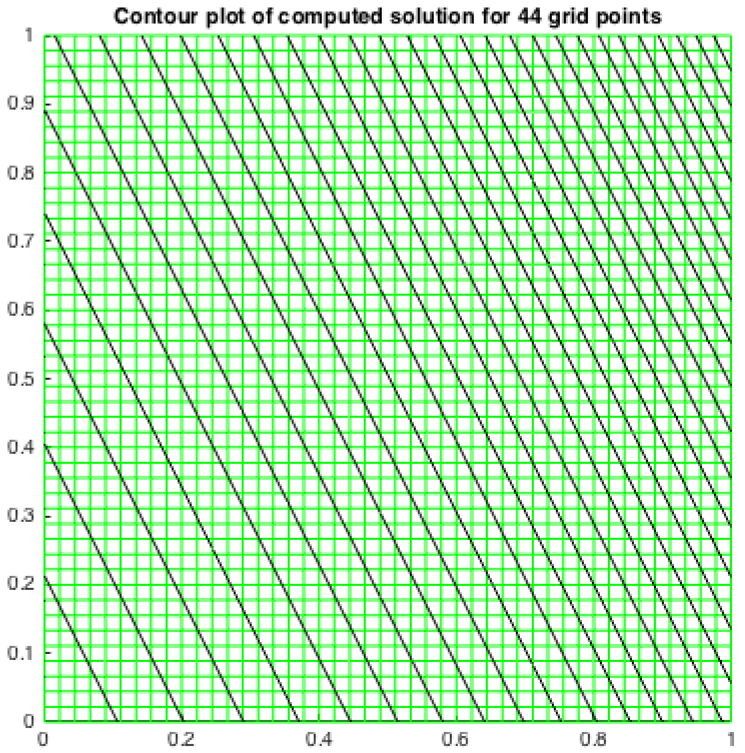



Contour plot of computed solution for 12 grid points
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1


Contour plot of computed solution for 20 grid points
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1


Contour plot of computed solution for 28 grid points
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1


Contour plot of computed solution for 36 grid points
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1



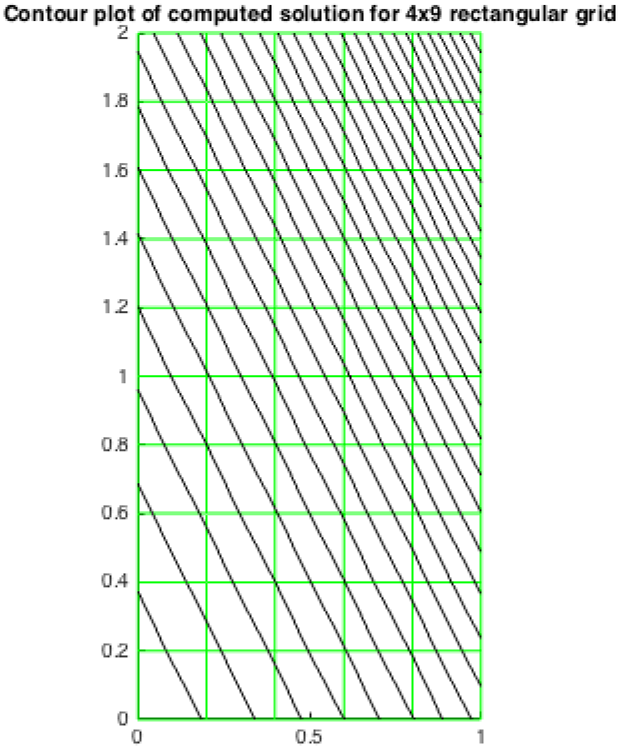
Problem 3.1.b

```
clear all
count = 6;
count = count+1;
m = 4;
ax = 0;
bx = 1;
ay = 0;
by = 2;
h = (bx-ax)/(m+1);
mx = (bx-ax)/h-1;
my = (by-ay)/h-1;

x = linspace(ax, bx, mx+2);  % grid points x including boundaries
y = linspace(ay, by, 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)/h^2;
rhs(:,my) = rhs(:,my) - usoln(Iint,my+2)/h^2;
rhs(1,:) = rhs(1,:) - usoln(1,Jint)/h^2;
rhs(mx,:) = rhs(mx,:) - usoln(mx+2,Jint)/h^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);
e = ones(my,1);
T = spdiags([e -4*e e],[-1 0 1],mx,mx);
S = spdiags([e e],[-1 1],my,my);
A = (kron(Iy,T) + kron(S,Ix)) / 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,mx,my);

% assuming true solution is known and stored in utrue:
```

```
err = max(max(abs(usoln-utrue)));
fprintf('grid size: %dx%d\n', mx, my);
fprintf('Error relative to true solution of PDE = %10.5e \n',err)

% plot results:

figure(count)
hold on

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

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

axis([ax bx ay by])
daspect([1 1 1])
name = sprintf('Contour plot of computed solution for %dx%d
 rectangular grid', mx, my);
title(name)
hold off
```

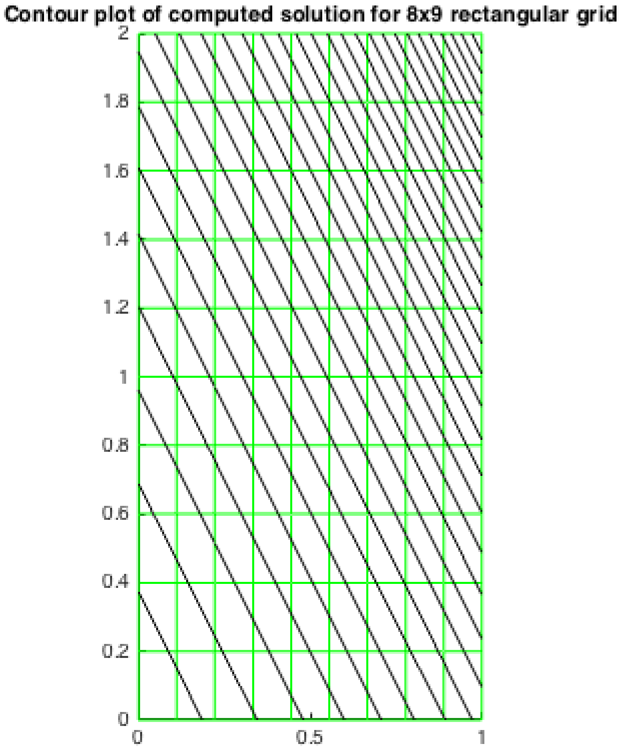
```
grid size: 4x9
Error relative to true solution of PDE = 1.18510e-03
```



Problem 3.1.c

```
clear all
count = 7;
count = count+1;
mx = 8;
my = 9;
ax = 0;
bx = 1;
ay = 0;
by = 2;
hx = (bx-ax)/(mx+1);
hy = (by-ay)/(my+1);

x = linspace(ax, bx, mx+2);  % grid points x including boundaries
y = linspace(ay, by, 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);
e = ones(my,1);
Tx = spdiags([e -2*e e],[-1 0 1],mx,mx);
Ty = spdiags([0*e -2*e 0*e],[-1 0 1],mx,mx);
S = spdiags([e e],[-1 1],my,my);
A = (kron(Iy,Tx)/hx^2 + kron(Iy,Ty)/hy^2 + kron(S,Ix)/hy^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('grid size: %dx%d\n', mx, my);
fprintf('Error relative to true solution of PDE = %10.5e \n',err)

% plot results:

figure(count)
hold on

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

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

axis([ax bx ay by])
daspect([1 1 1])
name = sprintf('Contour plot of computed solution for %dx%d
 rectangular grid', mx, my);
title(name)
hold off
```

```
grid size: 8x9
Error relative to true solution of PDE = 4.20526e-04
```



Published with MATLAB® R2015b

```
%function u = conjugate_gradient(A,f,tol)
%
%   Example:
%   x = conjugate_gradient(A,b,tol)
%
f = F;
tol = 1e-5;
MAXITS = length(f);

u = 0*f;
r = f-A*u;
p = r;
for k = 1:MAXITS
    w = A*p;
    alpha = (r'*r)/(p'*w);
    unew = u+alpha*p;
    rnew = r - alpha*w;
    if( norm(rnew) < tol ),
        fprintf('Converged! its= %7.0f, tol=%10.3e\n', [k tol]);
        return;
    end
    beta = (rnew'*rnew)/(r'*r);
    p = rnew + beta*p;
    r = rnew;
    u = unew;
end
fprintf('Caution: CG went to max iterations without converging!\n');
fprintf('MAXITS = %7.0f, tol =%10.3e\n', [MAXITS tol]);

%end
```

```
Converged! its=      36, tol= 1.000e-05
```

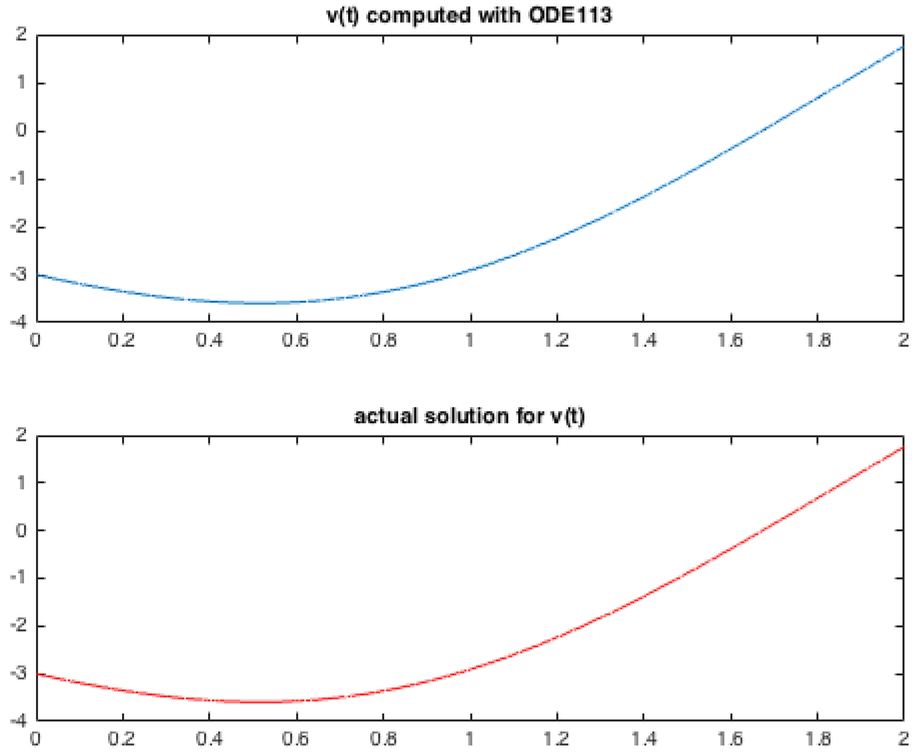
Published with MATLAB® R2015b

Table of Contents

```
% odesampletest
% test odesample for various tolerances
%
% From  http://www.amath.washington.edu/~rjl/fdmbook/chapter5  (2007)
```

Part A

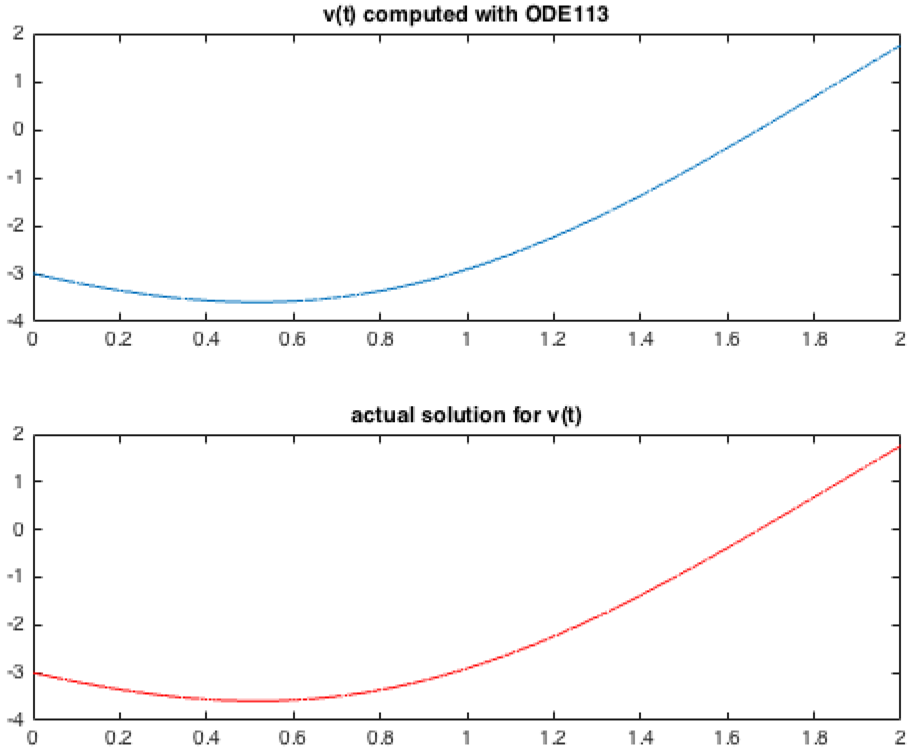
```
ODE113 = 'ode113';
tol = 1e-3;
[error] = Problem5_8_a(tol, 'on', ODE113);
```



Part C

```
close all
```

```
ODE113 = 'ode113';
tol = 1e-3;
err = Problem5_8_a(tol, 'on', ODE113);
```



Part D

```
clear all
ODE45 = ' ode45';
ODE113 = 'ode113';
global fcnevals
fprintf('Results or %s Solver', ODE113)
disp(' ')
disp('       tol       &     max error   &   f evaluations \\')
disp(' ')
for tol = logspace(-1,-13,13)
   %odesample(tol)
   err = Problem5_8_a(tol, 'off', ODE113);
   disp(sprintf('   %12.3e &   %12.3e   & %7i \\\\ ',tol, err,fcnevals))
end
disp(' ')
```

```
Results or ode113 Solver
       tol       &     max error   &   f evaluations \\

     1.000e-01 &      6.271e-04   &      27 \\
     1.000e-02 &      4.875e-04   &      29 \\
     1.000e-03 &     6.338e-04  &      33 \\
     1.000e-04 &     1.196e-04  &      41 \\
     1.000e-05 &     1.996e-05  &      47 \\
     1.000e-06 &     7.727e-07  &      63 \\
     1.000e-07 &     2.087e-07  &      73 \\
     1.000e-08 &     1.283e-08  &      87 \\
     1.000e-09 &     4.231e-10  &     115 \\
     1.000e-10 &     6.669e-11  &     131 \\
     1.000e-11 &     6.143e-12  &     147 \\
     1.000e-12 &     1.364e-12  &     157 \\
     1.000e-13 &     5.418e-14  &     177 \\
```

Part E

```
fprintf('Results or %s Solver', ODE45)
disp(' ')
disp('        tol      &     max error  &  f evaluations \\')
disp(' ')
for tol = logspace(-1,-13,13)
   %odesample(tol)
   err = Problem5_8_a(tol, 'off', ODE45);
   disp(sprintf('  %12.3e  & %12.3e &  %7i \\\\',tol, err,fcnevals))
end
```

```
Results or  ode45 Solver
       tol      &     max error   &  f evaluations \\

     1.000e-01   &     9.882e-06 &      67 \\
     1.000e-02   &     1.024e-05 &      67 \\
     1.000e-03   &     1.044e-05 &      67 \\
     1.000e-04   &     9.925e-06 &      67 \\
     1.000e-05   &     5.394e-06 &      85 \\
     1.000e-06   &     5.069e-07 &     127 \\
     1.000e-07   &     4.763e-08 &     199 \\
     1.000e-08   &     4.573e-09 &     313 \\
     1.000e-09   &     4.398e-10 &     493 \\
     1.000e-10   &     4.359e-11 &     781 \\
     1.000e-11   &     4.382e-12 &    1237 \\
     1.000e-12   &     4.325e-13 &    1951 \\
     1.000e-13   &     4.396e-14 &    3091 \\
```

Published with MATLAB® R2015b

```
function [error] = Problem5_8_a(tol, figDisp, solver)

% odesample.m
% Sample code for solving a system of ODEs in matlab.
% Solves  v'' = v^2 + (v')^2 - v -1  with v(0)=1, v'(0)=0
% with true solution v(t) = cos(t).
% Rewritten as a first order system.
% From  http://www.amath.washington.edu/~rjl/fdmbook/chapter5  (2007)


global fcnevals

t0 = 0;                      % initial time
u0 = [-3; -2; 2];    % initial data for u(t) as a vector
tfinal = 2;                  % final time
fcnevals = 0;                % counter for number of function
 evaluations

% solve ode:
options = odeset('AbsTol',tol,'RelTol',tol);
if(solver == 'ode113')
    odesolution = ode113(@f,[t0 tfinal],u0,options);
else %ODE45 default
    odesolution = ode45(@f,[t0 tfinal],u0,options);
end

% plot v = u(1) as a function of t:

figure('Visible', figDisp)
subplot(2, 1, 1)
t = linspace(0, tfinal, 500);
u = deval(odesolution, t);
v = u(1,:);
plot(t,v)
title('v(t) computed with ODE113')

% compare to true solution:
vtrue = -sin(2*t)+t.^2-3;
%hold on
subplot(2, 1 , 2)
plot(t,vtrue,'r')
title('actual solution for v(t)')
%hold off

error = max(abs(v-vtrue));
end


%-----------------------------------

function f = f(t,u)
global fcnevals
```

```
f1 = u(2);
f2 = u(3);
f3 = -u(3)-4*u(2)-4*u(1)+4*t^2+8*t-10;
f = [f1; f2; f3];

fcnevals = fcnevals + 1;
end
```

```
Not enough input arguments.

Error in Problem5_8_a (line 20)
options = odeset('AbsTol',tol,'RelTol',tol);
```

Published with MATLAB® R2015b