Solution of Week 11 Lab (Prepared by Yuan Yin)

November 6, 2020

1 Exercise 1: Runge-Kutta fixed step codes:

Run and examine 'myShowRK().m', 'RKStep.m', 'FixedRK.m', 'blobtrack2.m'.

2 Exercise 2: Embedded RK methods:

(a).

Go through 'predator-prey dynamics' in the Documentation.

(b).

[1]:

```
%%file EX2bDriver.m

function EX2bDriver

clc

tspan = [0, 2];
y0 = 1;
f = @(t, y) y.^2;

[TOUT1,YOUT1] = ode23(f,tspan,y0);
YOUT1

[TOUT2,YOUT2] = ode45(f,tspan,y0);
YOUT2

end
```



```
Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK
11/EX2bDriver.m'.
```

[2]:

```
EX2bDriver
```

```
Warning: Failure at t=1.001616e+00.  Unable to meet integration tolerances
without reducing the step size below the smallest value allowed (3.552714e-15)
at time t.
> In ode23 (line 299)
```

```
  In EX2bDriver (line 10)

YOUT1 =

   1.0e+13 *

    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0001
0.0001
0.0001
0.0001
0.0001
0.0001
0.0002
0.0002
0.0003
0.0003
0.0004
0.0004
0.0005
0.0007
0.0008
0.0010
0.0012
0.0014
0.0017
0.0021
0.0026
0.0031
0.0038
0.0046
0.0056
0.0068
0.0083
0.0100
0.0122
0.0148
0.0180
0.0218
0.0265
0.0322
```

```
    0.0391
    0.0475
    0.0577
    0.0701
    0.0851
    0.1033
    0.1255
    0.1524
    0.1851
    0.2248
    0.2731
    0.3316
    0.4028
    0.4892
    0.5941
    0.7215
    0.8763
    1.0642
    1.2925
    1.5698
    1.9065
    2.3154
    2.8121
    3.4152
    4.1478
    5.0375
    6.1330

Warning: Failure at t=9.999694e-01.  Unable to meet integration tolerances
without reducing the step size below the smallest value allowed (1.776357e-15)
at time t.
> In ode45 (line 360)
  In EX2bDriver (line 13)

YOUT2 =

   1.0e+14 *

    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
    0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
```

```
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000
0.0001
0.0001
0.0001
0.0001
0.0001
0.0001
0.0001
0.0001
0.0001
0.0002
0.0002
0.0002
0.0002
0.0003
0.0003
0.0004
0.0004
0.0004
0.0005
0.0006
0.0006
0.0007
0.0009
0.0010
0.0011
```

```
0.0013
0.0014
0.0016
0.0018
0.0021
0.0024
0.0029
0.0032
0.0035
0.0040
0.0046
0.0052
0.0059
0.0068
0.0081
0.0089
0.0099
0.0113
0.0130
0.0145
0.0165
0.0191
0.0227
0.0251
0.0280
0.0317
0.0366
0.0410
0.0465
0.0538
0.0639
0.0707
0.0789
0.0893
0.1030
0.1155
0.1312
0.1517
0.1803
0.1991
0.2227
0.2519
0.2901
0.3248
0.3686
0.4281
0.5081
0.5622
```

```
0.6246
0.7083
0.8182
0.9192
1.0465
1.2305
1.4720
1.6035
1.7943
2.0352
2.2934
2.5536
2.8823
3.3128
3.8921
```

Since the question is terribly conditioned near $t = 1$, neither of these methods will give an accurate answer. However, by fixing the step size to some relatively large number, we can skip some points near $t = 1$. Therefore, fixed-step method is more helpful. (Please check Tute10, Exercise Set 2, 'Try Yourself' part for more detailed explanation.)

(c).

Work through Section 7.7 of Moler.

(d).

Explanation of the output:

- For ode23, the local error $\sim h^3$. Therefore, we should have $h^3 \sim AbsTol$, where $AbsTol = 10^{-8}$. This implies that the step size for ode23 should be around $(10^{-8})^{\frac{1}{3}} \sim 0.04$;
- Similarly, for ode45, the local error $\sim h^5 \sim AbsTol = 10^{-8}$. $\Rightarrow$ The step size for ode45 is around $(10^{-8})^{\frac{1}{5}} \sim 0.4$;
- $\frac{step\ size\ for\ ode23}{step\ size\ for\ ode45} \sim \frac{1}{10} \Rightarrow \frac{Number\ of\ steps\ we\ need\ to\ take\ for\ ode23}{Number\ of\ steps\ we\ need\ to\ take\ for\ ode45} \sim 10$;
- However, for ode23, we need to take 3 function evaluations at each step while for ode45, we need 6. This implies that $\frac{The\ Number\ of\ function\ evaluations\ for\ ode23}{The\ Number\ of\ function\ evaluations\ for\ ode45} \sim 0.5$.
- Above is my analysis when $AbsTol = 10^{-8}$. One can see that this explanation is consistent with the output. Now, you can try to explain the case where we use default tolerance for ode45.

(e).

[7]:
```
%%file MyRigidode.m

function MyRigidode
%% EXERCISE SET 2 (e):

clc
```

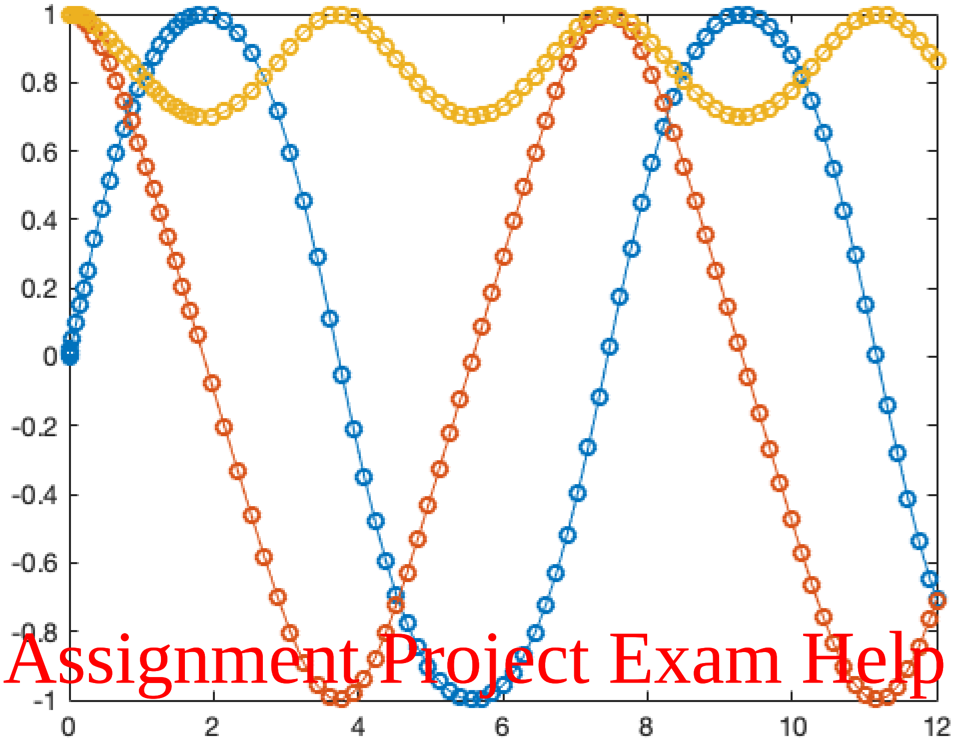
```
fprintf('\n\nFOR EXERCISE SET 2 (e):\n\n');
tspan = [0 12];
y0 = [0; 1; 1];

% solve the problem using ODE45
figure;
options_1 = odeset('Stats', 'on');
ode45(@f,tspan,y0, options_1);

fprintf('---------------------------\n');

figure;
options_2 = odeset('Stats', 'on', 'RelTol', 1.e-4);
ode45(@f,tspan,y0, options_2);


%% EXERCISE SET 3 (a):

fprintf('\n\n****************************************************\n\nFOR␣
 ↪EXERCISE SET 3 (a):\n\n');
ttspan = [0 12];
y00 = [0; 1; 1];

figure;
options = odeset('Stats', 'on', 'RelTol', 1.e-7);
ode45(@f,ttspan,y00, options);

fprintf('---------------------------\n');

figure;
options = odeset('Stats', 'on', 'RelTol', 1.e-7);
ode113(@f,ttspan,y00, options);

fprintf('---------------------------\n');

fprintf('As one can see from the output, ode113 is more efficient.\n');

end

function dydt = f(t,y)
dydt = [    y(2)*y(3)
   -y(1)*y(3)
   -0.51*y(1)*y(2) ];

end
```

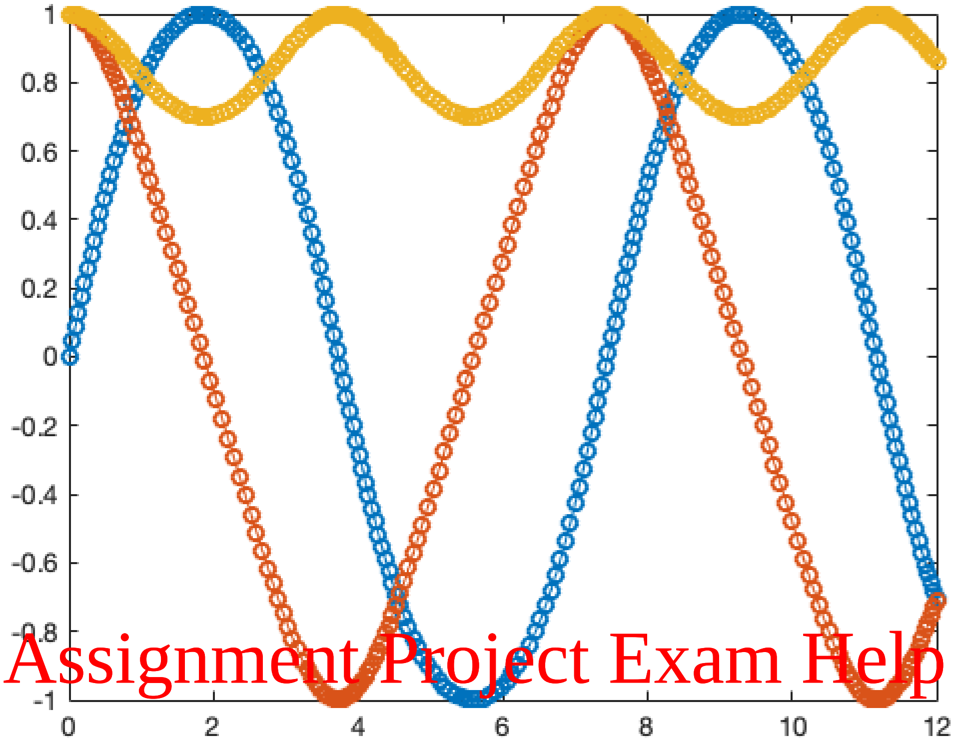

Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK

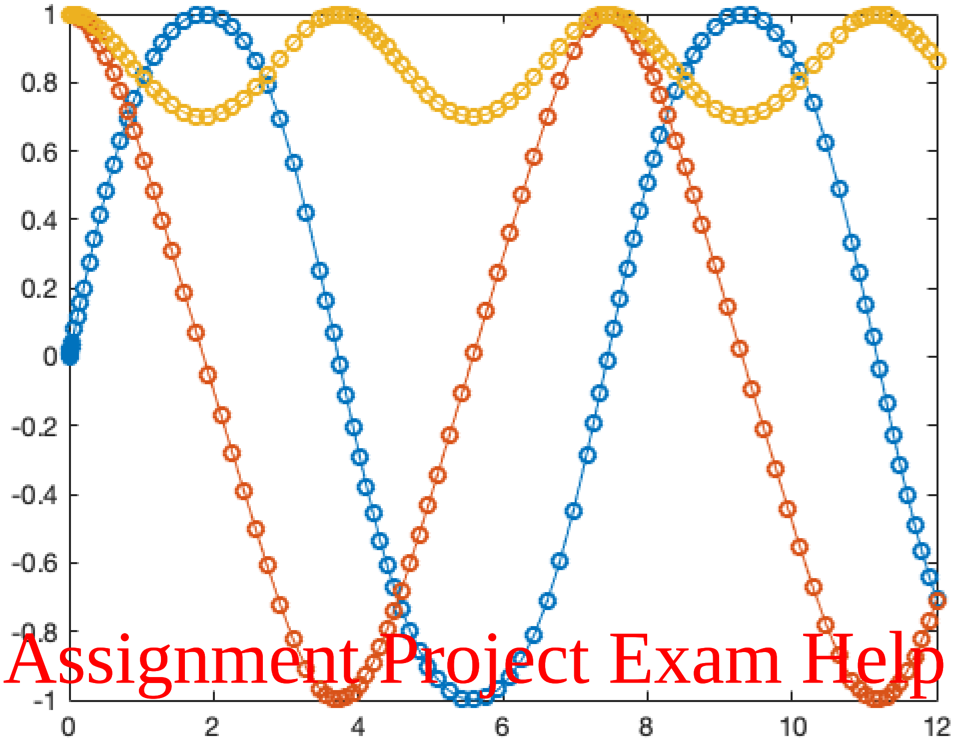
```
11/MyRigidode.m'.
```

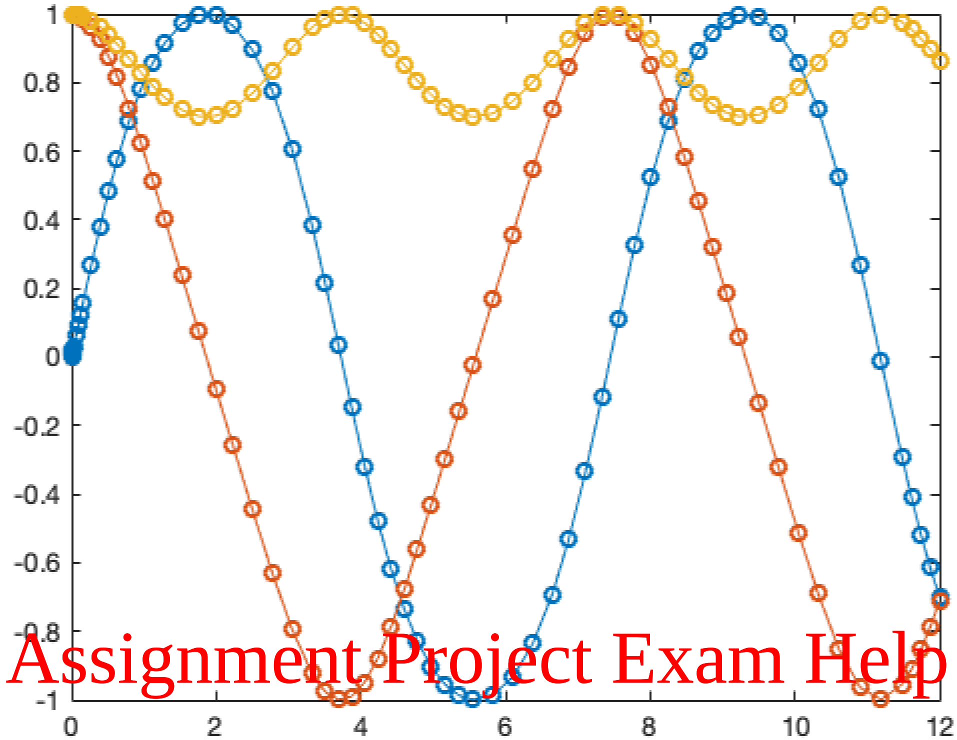
[8]:
```
MyRigidode
```

```
FOR EXERCISE SET 2 (e):

19 successful steps
2 failed attempts
127 function evaluations
---------------------------
27 successful steps
3 failed attempts
181 function evaluations


****************************************************

FOR EXERCISE SET 3 (a):

50 successful steps
1 failed attempts
307 function evaluations
---------------------------
92 successful steps
0 failed attempts
185 function evaluations
---------------------------
As one can see from the output, ode113 is more efficient.
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12



3 Exercise 3: Other MATLAB solvers:

(a).

Check my outputs from above (i.e. run MyRigidode.m to investigate the results.)

(b).

What is going on here? (What is the problem of ode45?)

Since this is a stiff problem, i.e. $|J| \to \infty$, we need to keep the step size 'tiny' when using ode45.

As you can see from the output figure, the step size is actually around 10^{-4}.

However, our tspan is from 0 to 2. This means that we need to take $\sim (2 \div 10^{-4})$ steps in total. However, each step involves 4 function evaluations……

Not efficient at all!

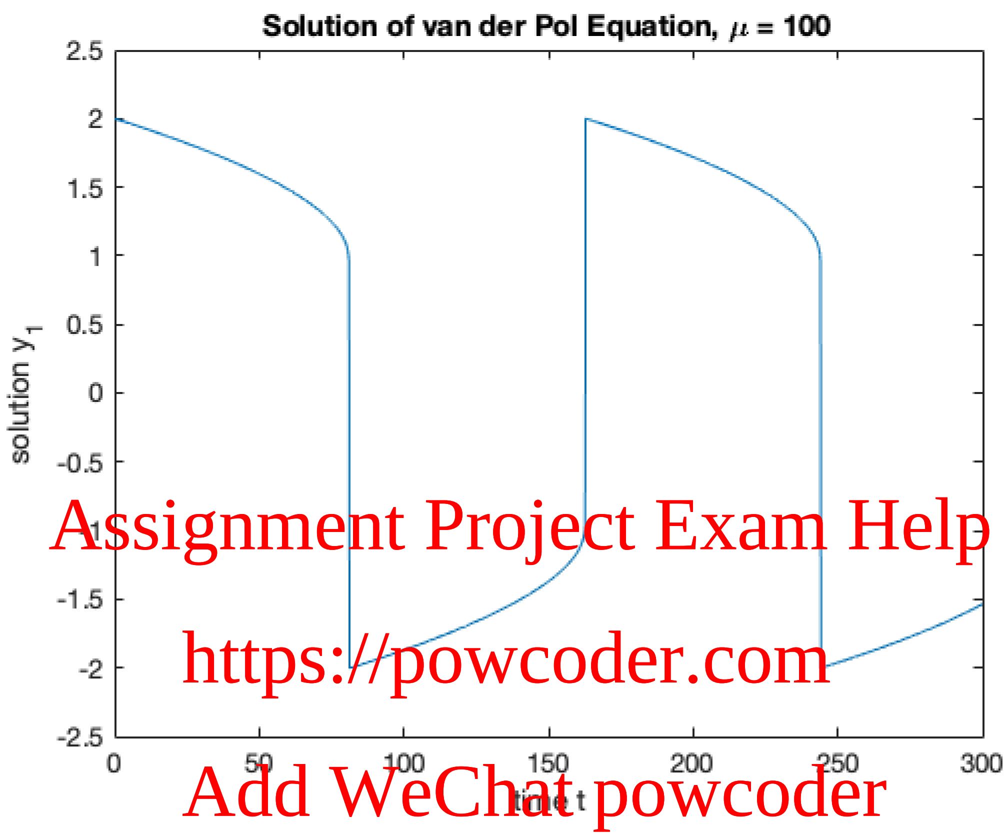
I mean, using ode45 to solve a stiff problem WILL give you a correct solution, it just costs so much time!

(c).

[9]:
```
%%file MyVdpode.m

function  MyVdpode(MU)

if nargin < 1
   MU = 100;
  % MU = 400;
  % MU = 800;
  % MU = 1000;      % default
end

tspan = [0; max(20,3*MU)];                   % several periods
y0 = [2; 0];
options = odeset('Jacobian',@J);

% [t,y] = ode15s(@f,tspan,y0,options);    % default stiff solver
 [t,y] = ode45(@f,tspan,y0,options);
% [t,y] = ode113(@f,tspan,y0,options);


figure;
plot(t,y(:,1));
title(['Solution of van der Pol Equation, \mu = ' num2str(MU)]);
xlabel('time t');
ylabel('solution y_1');

axis([tspan(1) tspan(end) -2.5 2.5]);

% ------------------------------------------------------------------------
% Nested functions -- MU is provided by the outer function.
%

   function dydt = f(t,y)
      % Derivative function. MU is provided by the outer function.
      dydt = [            y(2)
         MU*(1-y(1)^2)*y(2)-y(1) ];
   end
% ------------------------------------------------------------------------

   function dfdy = J(t,y)
      % Jacobian function. MU is provided by the outer function.
      dfdy = [         0                  1
         -2*MU*y(1)*y(2)-1    MU*(1-y(1)^2) ];
   end
% ------------------------------------------------------------------------

end  % vdpode
```

```
Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK
11/MyVdpode.m'.
```

[10]:
```
MyVdpode
```



When μ is small, e.g. $\mu = 100$, ode45 and ode113 solve the problem quite quickly. However, as μ increases its value, the problem becomes stiffer and stiffer, and using the non-stiff solvers, such as ode45 and ode113, is very time-consuming.