SUPPLEMENT 1

MORE ON ODE SOLVER

SUP1.1. FORMAT OF ODE SOLVER

The format of ode solver is as follows:

[**t**, **y**]=**solver**('F', **tspan**, **y0**) (s.1)

At the left hand side of the above equation, t is the array for time elapsed , **y** is the array for unknown variables, At the right hand side of the equation, the first one is the solver. There are 7 types of solvers to be used for various ode problems for version 5.3 of **MATLAB**. They are:

*Table s.1. Solvers of **MATLAB** ver 5.3.*

Solver	*Numerical method*	*Type of problem*	*Order*	*Tolerance*
ode45	Runge-Kutta	nonstiff	Medium	Low
ode23	Runge-Kutta	nonstiff	Low	High
ode113	Adams-Bashforth-Moulton	nonstiff	Variable	Low
Ode15s	Difference	stiff	Variable	Low
Ode23s	Modified Rosenbrock	stiff	Low	High
Ode23t	trapezoidal rule	moderately stiff	Medium	
Ode23tb		stiff	Low	

The second part at the right hand side is the 'F', which is a string for the **MATLAB** file name containing ode expressions. Format of the 'F' file is as follows:

```
Function ydot=F(t, y)
    y(1)=expression 1;
    y(2)=expression 2;
    y(3)=expression 3;
ydot=[y(1); y(2); y(3)];
```

The third item is tspan, which is a row matrix for the beginning and terminating of the program execution. For example, [0 20] represent begins at 0 and ends at 20. The fourth item is the y0, which is a row matrix for initial conditions. For example, [2; 0 ;1] represents $y(1)_0=2$, $y(2)_0=0$ and $y(3)_0=1$.

SUP1.2. WRITE THE ODE FILE

Before writing an ode program, two steps need to be done to transform the high order ode problem such as $f(t, y, y', y'',\ldots, y^{n-1})$ into an one order problem.

1. Define and replacement

define y_1=y, y_2=y’, y_3=y’’, ……., y_n= y^{n-1}

define a matrix Y=[y_1, y_2, y_3, ….., y_n]

Thus, f(t, y, y’, y’’,…, y^{n-1}) = f(t, y_1, y_2, …, y_n)

Y’= f(t, Y)

2. Write the expressions

$y'_1=y_2$, $y'_2=y_3$, $y'_3=y_4$, ……., $y'_{n-1}=y_n$

Finally, the function ‘F’ can be written as follows:

```
function ydot=F(t,y)
        expression 1:     y'1=y2;
        expression 2:     y'2=y3;
        expression 3:     y'3=y4;
          …….
        excpression n-1:  y'n-1=yn;
ydot=[expression 1; expression 2; expression 3;……expression n-1];
```

SUP1.3. EXAMPLES

Example 1.Write the ODE file for y’’’ + y’’ + y’ =0 using ‘odefuc1.m’ as filename.

Step 1: define y_1=y, y_2 = y’, y_3=y’’

Step 2: expressions: $y'_1=y_2$, $y'_2=y_3$, y'_3=y’’’=-y’’-y’=-y_3-y_2

```
%    odefuc1.m
function ydot= odefuc1(t,y);
%ydot=[y(2);y(3);-y(3)-y(2)];
T1=y(2);      T2=y(3);      T3=-y(3)-y(2);
ydot=[T1;T2;T3];
```

*Figure s.1. Source code of subprogram ‘**odefuc1**’.*

Example 2. Given y0 = [10;1;0] and tspan=[0 20], write the program ‘ode-sup1.m’ to solve y’’’ + y’’ + y’ =0 using ‘odefuc1.m’ as ode file and plot y vs. t and y’ vs. t.

```
% odesup1.m
clear; clc
y0=[10;1;0];    tspan=[0 20];
[t,y]=ode45('odefuc1',tspan,y0);
subplot(3,1,1); plot(t,y(:,1)); xlabel('t'); ylabel('dy/dt');
subplot(3,1,2); plot(t,y(:,2)); xlabel('t'); ylabel('d^2y/dt^2');
subplot(3,1,3); plot(t,y(:,3)); xlabel('t'); ylabel('d^3y/dt^3');
```

Figure s.2. Source code of main program ‘odesup1’.

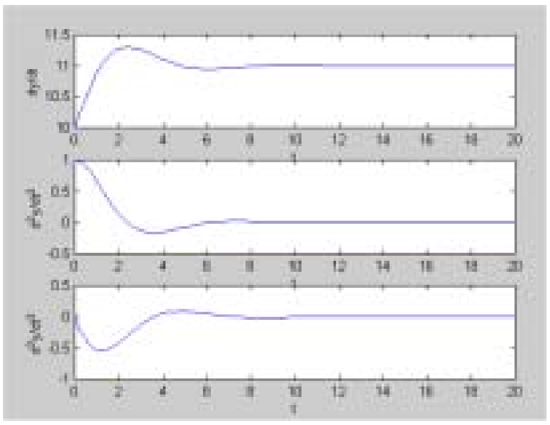

Figure s.3. Output of ‘odesup1’.

SUP1.4. CHECK THE PERFORMANCE OF THE SOLVER

The left hand side of the eq. s.1 was [**t**, **y**]. However, if users like to know the performance of the solver, One more parameter can be added as below:

[**t**, **y**, **s**]=**solver**(‘F’, **tspan**, **y0**) (s.2)

The output s is a column matrix with 6 elements. The first and two elements, s1 and s2, represent number of successful and fail executions. The third element s3 represents number of times calling ode file. The last 3 elements are for stiff problem only. The odesup2.m program listed below shows the time required to run the model using tic and toc commands.

```
% odesup2.m
clear; clc; tic
y0=[10;1;0];     tspan=[0 20];
[t,y,s]=ode45('odefuc1',tspan,y0);
toc
plot(t,y(:,1)); xlabel('t'); ylabel('dy/dt');
s
```

Figure s.4. Source code of main program ‘odesup2’.

The results shows

```
Elapsed_time =
       0.160
s    =
        30
        0
        181
        0
        0
        0
```

Figure s.5. Results of 'odesup2' displayed in command window.

which indicated that the ode45 solver solving the given ODE problem at 30 successful steps with no failed steps and calling function odefuc1 181 times. The time elapsed for simulation from time 0 to time 20 are 0.16 sec. Different solvers will lead to different elapsed times. Same solver in different computer might also lead to different elapsed times. Table s.2 shows the elapsed time and s values of available solvers solving' odesup2.m' model using version 5.3 of **MATLAB**.

Table s.2. Elapsed time and elements of s matrix in various ode solvers.

Solver	*Type of problem*	*Time**	*s*					
ode45	nonstiff	0.160	30	0	181	0	0	0
ode23	nonstiff	0.191	67	0	202	0	0	0
ode113	nonstiff	0.361	66	2	135	0	0	0
Ode15s	stiff	0.271	78	3	103	1	16	97
Ode23s	stiff	0.601	78	0	472	78	78	234
Ode23t	moderately stiff	0.381	125	0	141	1	10	135
Ode23tb	stiff	0.381	92	0	199	1	9	285

* Values listed are for reference only. The actual value depends on types of the computers to run the model.

SUP1.5. SET THE DEFAULT SETTINGS OF THE SOLVER

There are 3 parameters in the solver as described in eq. s.1. One more can be added as shown in eq. s.3.

[**t**, **y**]=**solver**('F', **tspan**, **y0**, **options**) (s.3)

Options can be set using odeset function as listed below.

options = **odeset**('name1',value1,'name2',value2,...)

The named properties have the specified values. Any unspecified properties have default values. It is sufficient to type only the leading characters that uniquely identify the property. Case is ignored for property names.

```
% odesup3.m
clear; clc;
y0=[10;1;0];
tspan=[0 20];
options=odeset(‘stats’,’on’);
[t,y]=ode45('odefuc1',tspan,y0,options);
plot(t,y(:,1)); xlabel('t'); ylabel('dy/dt');
```

Figure s.6. Source code of main program ‘odesup3’.

The result shows computational cost statistics as below:

```
30 successful steps
0 failed attempts
181 function evaluations
0 partial derivatives
0 LU decompositions
0 solutions of linear systems
```

Figure s.7. Results of ‘odesup3’ displayed in command window.

The tolerance related odeset properties are ‘**RelTol**’ and ‘**AbsTol**’ with default values of 1e-3 and 1e-6, respectively in all solver. The estimated error in each integration step satisfies e(i) <= **max**(**RelTol** * **abs**(y(i)),**AbsTol**(i)).

For some complicated problems, the time required for all variables to converge is very long, by resetting these values a little bigger can speed up the execution time with some scarify on accuracy. For example, as shown in ‘odesup4.m’ the ‘**RelTol**’ is 10 times of the default value. Using ode23s solver, the time required to execute was 0.601 sec as shown in Table s.2 and now reduced to 0.421 sec. Number of function evaluations also reduced from 472 to 256.

```
% odesup4.m
clear; clc; tic
y0=[10;1;0]; tspan=[0 20];
options=odeset(‘stats’,’on’,’RelTol’,1e-2);
[t,y]=ode23s('odefuc1',tspan,y0,options);
toc
plot(t,y(:,1)); xlabel('t'); ylabel('dy/dt');
```

Figure s.8. Source code of main program ‘odesup4’.

The result shows computational cost statistics and time required to run as below:

```
42 successful steps
0 failed attempts
256 function evaluations
42 partial derivatives
42 LU decompositions
126 solutions of linear systems
elapsed_time =

    0.4210
```

Figure s.9. Results of ‘odesup4’ displayed in command window.

Ode solvers also provide 2-D and 3-D plots using ‘**odephas2**’ and ‘**odephas3**’, respectively in ‘**OutputFcn**’. The data of variables for drawing are selected by assigning number in the ‘**OutputSel**’. Fig. s.10 and s.11 show the source code and the output of ‘Odesup5.m’ using y(1), y(2) and y(3) for data of x y, z axis.

```
% odesup5.m
clear; clc
y0=[10;1;0]; tspan=[0 20];
options=odeset('OutputFcn','odephas3','OutputSel',[1 2 3]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(1)');ylabel('y(2)'); zlabel('y(3)');
```

Figure s.10. Source code of main program ‘odesup5’.

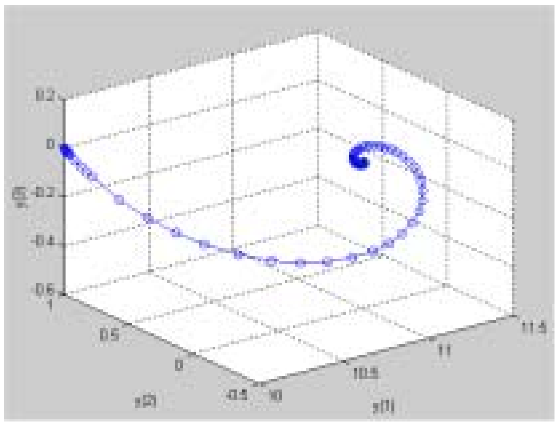

Figure s.11. Results of ‘odesup5’showing 3-D graph

Fig. s.12 and s.13 show the source code and the output of ‘Odesup6.m’ drawing four 2-D graphs in one figure.

.

```
% odesup6.m
clear; clc
y0=[10;1;0]; tspan=[0 20];
subplot(2,2,1);
options=odeset('OutputFcn','odephas2','OutputSel',[1 2]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(1)');ylabel('y(2)');
subplot(2,2,2);
options=odeset('OutputFcn','odephas2','OutputSel',[2 3]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(2)');ylabel('y(3)');
subplot(2,2,3);
options=odeset('OutputFcn','odephas2','OutputSel',[1 3]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(1)');ylabel('y(3)');
subplot(2,2,4);
options=odeset('OutputFcn','odephas2','OutputSel',[3 1]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(3)');ylabel('y(1)');
```

Figure s.12. Source code of main program ‘odesup6’.

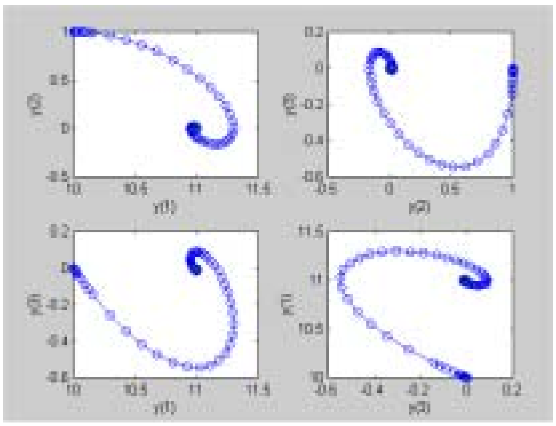

Figure s.13. Results of ‘odesup6’ showing 2-D graph.

Fig. s.14 and s.15 show the source code and the output of ‘Odesup7.m’. This program demos the difference in refinement. The default refine-factor for ode45 solver is 4 and for other solvers is 1. As shown in %-- Second plot of the ‘odesup7.m’, the refine-factor is now changed to 20. The effect can be clearly demonstrated when comparing two plots in Fig. S.15.

```
% odesup7.m
clear; clc
y0=[10;1;0]; tspan=[0 20];
subplot(2,1,1); % ------------------ First plot
options=odeset('OutputFcn','odephas3','OutputSel',[1 2 3]);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(1)');ylabel('y(2)'); zlabel('y(3)');
subplot(2,1,2); % ------------------ Second plot
options=odeset('OutputFcn','odephas3','OutputSel',[1 2 3],'refine',20);
[t,y]=ode45('odefuc1',tspan,y0,options);
xlabel('y(1)');ylabel('y(2)'); zlabel('y(3)');
```

Figure s.14. Source code of main program ‘odesup7’.

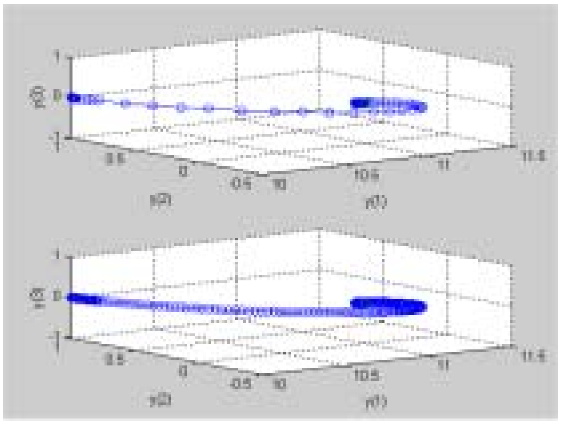

Figure s.15. Outputs of ‘odesup7’ showing results of refinement.

More **odeset** properties can be retrieve by typing ‘help odeset’ in the command window of **MATLAB**.