Matlab Practical II: ODEs etc.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using M-Files

type myfile
x=2; y=5;
myfile
disp([a,b])

type fun
[a,b]=fun(x,y);
disp([a,b])
[c,d]=fun(-3, 4);
disp([c,d])
```

the file `myfile.m`

```
a=x+y;
b=x*y;
```

the file `fun.m`

```
function [a,b]=fun(x,y)
a=x+y;
b=x*y;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simple Plotting

x=[1:10]
x=linspace(1,10,10)
x=linspace(1,10)
plot(x,'k*')

x=linspace(1,10,500);
plot(x,x.^2,'k*')      % x.^2, NOT x^2
semilogy(x,x.^2,'k*')

x=logspace(0,3,500);
plot(x,x.^2,'*')
semilogx(x,x.^2,'*')
loglog(x,x.^2,'*')

fplot('cos(x)/x',[-pi,pi])
axis
axis([-pi,pi,-100,100])
axis square
axis equal
axis([-pi,pi,-5,5])
axis off
```

```
% Quadrature

help quad
more on
help quad

% use symbolic math toolbox
int('x')
int('exp(x)')
int('exp(-x^2)')
help erf

format long
sqrt(pi)/2 * erf(1)
erf(0)
% use adaptive Simpson rule
quad('f', 0, 1)
quad('f', 0, 1, 1e-10)
% use adaptive Lobatto rule
quadl('f', 0, 1)
help quadl
```

the file `f.m`

```
function y=f(x)
y=exp(-x.*x);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Differential Equations

% uses "f1.m"

tic
toc
tic; [x,y]=ode23('f1',[0 10],0); toc  % approximate solution
plot(x,y)
title('y''(x)=e^{-x}cos(x)-y','fontsize',20)
hold on
fplot('exp(-x)*sin(x)',[0 10],'r-')   % exact solution
zoom on                               % can zoom from figure menu
zoom out
zoom off
clf
plot(x,y-exp(-x).*sin(x),'b-')        % plot the error
```

the file `f1.m`

```
function yprime=f1(x,y)
yprime=exp(-x)*cos(x)-y;
```

```
tic; [x,y]=ode45('f1',[0 10],0); toc    % new solution, higher order method
hold on
plot(x,y-exp(-x).*sin(x), 'r-')         % plot new error
clf
semilogy(x(2:end)-x(1:end-1))           % plot time steps
close


help ode23
which ode23
type ode23
clc

% uses "f2.m"
[x,y]=ode23('f2',[0 500],[0; 1]);
plot(y(:,1),y(:,2))
axis square
help odeset
close

% uses "f3.m" and "plottraj.m"
y1=[0:.4:4];
y2=[0:.4:4];
[Y1,Y2]=meshgrid(y1,y2);
U=Y1-Y1.*Y2;
V=-Y2+Y1.*Y2;
quiver(Y1,Y2,U,V)
plottraj('f3')
close

% uses "f4.m" and "plottraj.m"
y1=[0:.4:5];
y2=[0:.4:5];
[Y1,Y2]=meshgrid(y1,y2);
U=Y1.*(3-Y1-Y2);
V=Y2.*(Y1-1);
quiver(Y1,Y2,U,V)
axis([0,5,0,5])
plottraj('f4')
zoom on
close

% Three dimensional plots of ODE solutions
[t,y] = ode45('f3',[0 20],[0.8;0.1]);
plot3(t,y(:,1),y(:,2),'r-')
xlabel('t'), ylabel('y_1'), zlabel('y_2')
rotate3d      % or rotate from figure menu
grid on

[t,y]=ode45('f4',[0 20],[0.8;0.1]);
plot3(t,y(:,1),y(:,2),'r-')
xlabel('t'), ylabel('y_1'), zlabel('y_2')

help funfun
help bvp4c
help twobvp
```

the file `f2.m`

```
function yprime=f2(x,y)
yprime=[y(2);-y(1)];
```

the file `f3.m`

```
function yprime=f3(x,y)
% Bender & Orszag, Example 4, p. 179
% Lotka-Volterra model: y1=prey; y2=predator
yprime=[y(1)-y(1)*y(2);-y(2)+y(1)*y(2)];
```

the file `plottraj.m`

```
function plottraj(fstr)
disp('click mouse on initial point');
[y1,y2]=ginput(1)
[x,y]=ode45(fstr,[0 10],[y1;y2]);
hold on, plot(y(:,1),y(:,2),'r-')
```

the file `f4.m`

```
function yprime=f4(x,y)
% Bender & Orszag, Example 5, p. 179
yprime=[y(1)*(3-y(1)-y(2)); y(2)*(y(1)-1)];
```

To print a figure to a Postscript file:

```
print -deps2  fig.eps    % black and white
print -depsc2 fig.eps    % color
```