Matlab and ordinary differential equations

Matlab can solve ODEs both symbolically and numerically. To find symbolic solutions, use `dsolve` as in the examples below:

```
>> dsolve('D2y-3*Dy-4*y=cos(t)')
ans = -1/34*(5*cos(t)*exp(t)+3*exp(t)*sin(t)-34*C1*exp(4*t)*exp(t)-34*C2)/exp(t)
>> dsolve('D2y-3*Dy-4*y=cos(t)','Dy(0)=1','y(0)=2')
ans = -1/34*(5*cos(t)*exp(t)+3*exp(t)*sin(t)-22*exp(4*t)*exp(t)-51)/exp(t)
>> simplify(ans)
ans = -5/34*cos(t)-3/34*sin(t)+11/17*exp(4*t)+3/2*exp(-t)
>> dsolve('y+t*Dy=0')
ans = 1/t*C1
>> [x y] = dsolve('3*Dx=2*y-x','Dy=4*x/3+y/3','x(0)=2,y(0)=1')
x =  exp(-t)+exp(t)
y =  2*exp(t)-exp(-t)
```

The letter upper case D stands for the derivative with respect to t. D2 is the second derivative, D3 is the third, and so on.

To solve ODEs numerically, you can use `ode45`. For example let us solve $y' = y^6 - t^2$ for $0 \le t \le 1$ with the initial condition $y(0) = .9$.

```
>> syms y t
>> f = inline(vectorize(y^6-t^2),'t','y');
>> ode45(f,[0 1],.9)
```

In an apparently undocumented feature, this will produce a graph of the solution, with circles around the points which matlab calculated. To obtain a list of the y and t values, use the command

```
>> [T Y] = ode45(f,[0 1],.9)
```

Note in this example, when $t \ge .3023$ or so, the value of Y is given as NaN, standing for not a number, which indicates that the solution has gone off to infinity there.

By default, `ode45` attempts for three digit accuracy, (or error less than 10^{-6} if the answer has absolute value less than 10^{-3}). This can be changed using `odeset`, typing `help ode45` gives more information.

You can graph several solutions with various initial values by giving a row vector of initial values, for example `ode45(f,[0 1],[.5 1])` graphs two solutions with initial values $y(0) = 1$ and $y(0) = 1/2$.

You can solve systems of equations with ode45 also. For example to solve the system $y_1' = y_1 + t$, $y_2' = y_1 + y_2$, $0 \le t \le 1$, $y_1(0) = 1$, $y_2(0) = 2$

```
>>  g = inline(vectorize('[y(1)+t; y(1)+y(2)]'),'t','y');
>> [T Y] = ode45(g,[0 1],[1;2])
```

There is something to watch when you do this. The function and initial values should be given as a column vector, so you separate entries by a semicolon as above. Also the argument to `vectorize` should be a string enclosed in single quotes as indicated above. Otherwise matlab gets unhappy.

You can have matlab plot your solutions `[T Y]` using `plot` for two dimensional plots or `plot3` for three dimensional plots. For example to plot the five solutions to $y' = 3y + \sin t$ for $0 \le t \le 2$ with initial values $y(0) = 0, 1, -1, 2, -2$ type:

```
>>  f = inline(vectorize(3*y+sin(t)),'t','y')
>> [T Y] = ode45(f,[0 2],[0 1 -1 2 -2]);
>> plot(T,Y)
```

To plot the solutions to $y_1' = y_1 y_2 - t$, $y_2' = y_1 - y_2$, $y_1(1) = 3$, $y_2(1) = 0$ for $0 \le t \le 1$ we can do

```
>> g = inline(vectorize('[y(1)*y(2)-t;  y(1)-y(2)]'),'t','y')
>> [T Y] = ode45(g,[1 0], [3; 0]);
>> plot3(T,Y(:,1),Y(:,2))
```

The second argument of `ode45` is `[1 0]` since our initial t value is 1 and the final t value is 0. The expression `Y(:,1)` is the first column of Y which gives the y_1 values.

Second and higher order equations can be solved numerically by changing them to first order. For example $y'' - y' + 10y = \sin t$, $y(0) = 0$, $y'(0) = 0$ can be changed to $y_1' = y_2$, $y_2' = y_2 - 10y_1 + \sin t$, $y_1(0) = 0$, $y_2(0) = 0$ and solved and graphed by:

```
>> h = inline(vectorize('[y(2); y(2)-10*y(1)+sin(t)]'),'t','y')
>> [T Y] = ode45(h,[0 2*pi], [0;0]);
>> plot(T,Y(:,1))
```

Use matlab to solve the following problems, due Friday, April 4. You may work in pairs if you wish.

Problem 1: p. 66 prob 7.

Problem 2: p. 149 problem 1.

Problem 3: Graph an approximate solution to $y'' + t^2 y' - y = \sin t$, $y(0) = 1$, $y'(0) = 2$ for $0 \leq t \leq 4\pi$.