Using MATLAB to solve ODEs

MATLAB has a few different ODE solvers, including:

ode23: Uses simultaneously second and third order Runge Kutta formulas to make estimates of the error, and calculate the time step size. Since the second and third order RK require less steps, ode23 is "less expensive" in terms of computation demands than ode45, but is also lower order. Use for non stiff ODEs.

ode45; Uses simultaneously fourth and fifth order RK formulas to make error estimates and adjust the time step accordingly. MATLAB recommends that ode45 is used as a first solver for a problem. For nonstiff ODEs.

ode113: Uses variable-order Adams-Bashforth-Moulton solver. For problems with stringent error tolerances or for solving computationally intensive problems. For nonstiff ODEs.

ode15s: If using ode45 turns out to be a slow process, then the DEs you are solving are stiff. Use ode15s or one of the solvers below.

ode23s, ode23t, ode23tb

For more information on using a particular solver, type at the matlab prompt:

```
help ode45
```

Usage

```
[t,y] = ode45(@problem1, [0, 6], 2.4);
plot(t,y)
xlabel('time')
ylabel('function values')
```

The above assumes that the definition of $f(t, y)$ for the ODE

$$\frac{dy}{dt} = f(t, y)$$

is in the matlab file problem1.m, and that the ODE is solved in the time interval $0 \leq t \leq 6$ with initial condition $y(0) = 2.4$.

The form of the ODE is defined within the file problem1.m. For a single ODE the general format is always the same. For a single, first order ODE of the form:

$$y' = f(t, y)$$

the corresponding program looks like:

```
function yp = program1(t,y)
yp = f(t,y);
```

For example, the program1.m

```
function yp = program1(t,y)
yp = 2*y + 1./t;
```

defines the ODE $y' = 2y + \frac{1}{t}$, to be solved as

```
[t,y] = ode45(@program1, [t_0 t_final], y0)
```

for some time interval $t_0 \leq t \leq t_{\text{final}}$ and some initial condition $y(0) = y_0$.

MATLAB can also handle a system of ODEs, such as

$$\begin{aligned} y_1' &= f_1(t, y_1, y_2, ..., y_n) \\ &... \\ y_n' &= f_n(t, y_1, y_2, ..., y_n) \end{aligned}$$

For example, the file program5.m

```
function yp = program5(t,y)
%
alpha = 0.3;
beta = 0.4;
xx = y(1);
yy = y(2);
yp1 = -beta*xx*yy + xx;
yp2 = beta*xx*yy - alpha*yy;
yp=[yp1; yp2];
```

defines the system of ODEs

$$\begin{aligned} x' &= -\beta xy + x, \\ y' &= \beta xy - \alpha y \end{aligned}$$

to be solved as

```
[t,y] = ode45(@program5, [0 50], [0.1, 0.6])
```

for some (randomly chosen) initial condition $x(0) = 0.1$ and $y(0) = 0.6$.

Note: Neither the name, nor the number after the name (e.g. program1) play a role in the definition of a problem. The names can be anything, as long as the conform to MATLAB naming rules for functions and m-files.