Mathematica Tutorial - NDSolve

NDSolve numerically solves a set of ODEs with boundary conditions. Consider this code:

```
theta=Pi/4;  v0=3;
soln1 =  NDSolve[{y''[t] == -9.8,    x''[t] == 0,
                        x'[0] == v0*Cos[theta],   y'[0] == v0*Sin[theta],
                        x[0] == 0, y[0] == 0}, {y, x}, {t, 0, 10}]
```

1) Jeopardy question: The code above is the answer, what physics question is it asking?

Now, try it out. To *show* the solution, try adding:

```
   ParametricPlot[{x[t], y[t]} /. soln1, {t, 0, 2},  PlotRange -> {{0, 1}, {0, .5}}]
```

What do you think "/." does, in this expression?

Plot y vs time.   Do your plots (this one, and the parametric one) match what you expect?

2) Consider the following three questions. (Answer them without running any codes yet.)
A) If you add linear drag, what happens (qualitatively) to the horizontal range (with angle fixed)?   Why?

B) You launch a projectile straight up, first in an ideal (frictionless) world, and then with linear air drag. In which of these cases is the time to reach its *peak height* larger?
(Or, does it depend? Bear in mind that it won't go as high if there is air drag)
What is your reasoning? (Try to be rigorous!)

C) You launch a projectile at some angle. With linear air drag, does the time to get up to the peak equal the time to fall back down to the ground? (If not, which is larger?)
Does your answer depend on the launch angle? Why/why not?

(continued on back)

Now, copy your above code, and change the differential equation code to include linear air drag, -b **v**. (Don't forget to add in a value for b in the top line.)
To check for syntax errors, start by setting b=0. You should reproduce the previous plot)
Rerun that "ParametricPlot" with non-zero b to see the new trajectory with drag.

Check your answers to the 3 questions on the previous page, which I summarize as:
A) What happens to the horizontal range?
B) Compare t (to peak) in the real and ideal situations.
C) Compare t(up) to t(down)

Though definitely not needed for qualitative answers, you might find these MMA commands helpful, where t0 is a numerical value where you want MMA to start "hunting" for a numerical solution:

```
FindRoot[y[t] == 0 /. soln, {t, t0}]
FindMaximum[y[t] /. soln, {t,t0}]
```

Bonus: (If you don't get to it - try this at home!!)
Get rid of *everything* except the NDSolve and ParametricPlot command.
Surround the code with the Manipulate command at the top:

```
Manipulate[
<commands here>
```

and at the end, finish off the manipulate command with a line which reads

```
, {{v0,3}, 0,3},    {{theta, Pi/4}, 0,Pi/2},    {b,0,2}]
```

(Is this cool, or what? The syntax here is that v0 is initially 3, but can vary from 0 to 3 (etc) Copy it carefully! Try it! This is a great tool for quickly manipulating variables. *)*