Lab 7 Tuesday, October 30

Euler's Method

Download the file `euler.nb` from the course web page, and evaluate the first block of code. This will give you four functions.

`euler[df,x0,y0,xfin,n]` uses Euler's method to approximate $f(x_{fin})$ given the initial data $y' = df, f(x_0) = y_0$ and n steps. It outputs the estimate of $f(x_{fin})$.

`eulerplot[df,x0,y0,xfin,n]` runs the same approximation, but instead of reporting the estimate as a number, it plots the data points.

`compareplot[f_, x0_, y0_, xfin_, n_]` takes a known function and plots that function on the domain $[x_0, x_{fin}]$; against it it plots the data points generated by using Euler's method to estimate $f(x_{fin})$ with the initial data $f(x_0) = y_0$, but computing the true derivative at every point.

`comparewitherrors[f_, df_, x0_, y0_, xfin_, n_]` plots $f(x)$ on $[x_0, x_{fin}]$ as before. But instead of comparing to Euler's method with the true derivative, it compares to Euler's method as estimated with the differential equation $y' = df$.

1. Consider the example equation $f'(t) = f(t) - f(t)^2/2$ with $f(0) = 1$.
   (a) Use the command `euler[y - y^2/2, 0, 1, 3, 3]` to estimate $f(3)$ given $y' = y - y^2/2$ and $f(0) = 1$, with three steps. How does this compare to the answer we got before?
   (b) Use the command `eulerplot[y - y^2/2, 0, 1, 3, 3]` to see these results graphically.
   (c) Now try using nine steps, with `euler[y - y^2/2,0,1,3,9]`. What changes? Do the same with `eulerplot`
   (d) Now try using 100 steps.
2. Now let's play around with the comparison plot. We already looked at $\sin(x)$.
   Now let's consider the function $f(x) = e^x$. We haven't studied this function in class yet, but it's a fact that it satisfies the differential equation $y' = y$.
   (a) Use the functions `compareplot[E^x, 0, 1, 5, 5]` and `comparewitherrors[E^x,y, 0,1,5,5]` to compare the actual function to the results of Euler's method.
   (b) Why did we pick the initial values we did?
   (c) Which approximation is better? Why?
   (d) Try again with 10 steps instead of 5. Try 100 steps. Play around and see what happens as you change the step size.
3. Another important differential equation is the *logistic growth equation*, which is often used to model population growth. The differential equation is $y' = y(1 - y)$ and the corresponding function is $f(x) = \frac{1}{1+\frac{1-x_0}{x_0}e^{-x}}$ or `L[x_]:=1 / (1 + (1-x0)/x_0 E^(-x))`, where x_0 is the initial condition
   (a) We'll take the initial condition $x_0 = 1/2$. (This corresponds to a population at 50% of maximum capacity). This gives the function `L[x_]:=1/(1 + E^(-x))`
   (b) Use the command `compareplot[1/(1+E^(-x)),0,.5,10,10]` to estimate $L(10)$. What happens?

(c) What if we raise the step size to 100?

(d) Now let's compare to the differential equation. Use the command `comparewitherrors[1/(1 + E^(-x)), y (1 - y), 0, .5, 10, 10]` to use Euler's method to fit the equation. What happens? How is this different from before?

(e) Increase the step size to 100.

4. Let's look at the same function with different initial conditions.

   (a) We'll take the initial condition $x_0 = 2$, corresponding to a population at double capacity. This gives the function `L[x_] := 1/(1 - E^(-x)/2)`.

   (b) Use the command `compareplot[1/(1-E^(-x)/2),0,2,10,10]` to estimate $L(10)$. What happens?

   (c) What if we raise the step size to 100? To 1000?

   (d) Now let's compare to the differential equation. Use the command `comparewitherrors[1/(1 - E^(-x)/2), y (1 - y), 0, 2, 10, 10]` to use Euler's method to fit the equation. What happens? How is this different from before?

   (e) Increase the step size to 100. To 1000.

   (f) Does using the "true" derivative, or the differential equation, work better? Why do you think this is?

5. The differential equation $y' = y - e^x \sin(5x)/2 + 5e^x \cos(5x)$ with initial conditions $y(0) = 0$ has solution $y = e^x \sin(5)$.

   (a) Run `compareplot[E^x Sin[5x],0,0,10,10]`. What happens?

   (b) Try with 100 steps. Try with 100.

   (c) Run `comparewitherrors[E^x Sin[5x],y-E^x Sin[5x]/2+5E^x Cos[5x],0,0,10,10]`

   (d) Try with 100 steps. Try with 1000.

6. If we take $y' = 2y/x - x^2y^2$, you can check that $f(x) = \frac{5x^2}{x^5+4}$ is a solution, with initial condition $f(1) = 1$.

   (a) Use the command `compareplot[5x^2/(x^5+4),1,1,5,5]` to estimate $L(5)$. What happens?

   (b) What if we raise the number of steps to 10? To 100?

   (c) Now let's compare to the differential equation. Use the command `comparewitherrors[5 x^2/(x^5 + 4), 2 y/x - x^2 y^2, 1, 1, 5, 10]` to use Euler's method to fit the equation. What happens? How is this different from before?

   (d) Increase the step size to 100. To 1000.

Implict Functions and their Tangents

When using the `ContourPlot` command, note the double `==` signs.

1. In class, we showed that the tangent line to $x^3 + y^3 = 6xy$ at the point $(3, 3)$ is $y - 3 = 3 - x$. Verify this with the command
`ContourPlot[{x^3 + y ^3 == 6 x*y, y-3 == 3 -x}, {x,-5,5},{y,-5,5}]`

2. (a) Use ContourPlot to plot the "cardioid" with equation:
   `x^2 + y^2 == ( 2x^2 + 2y^2 -x)^2`. (x and y domains from -1 to 1).
   (b) Compute the derivative at the point $(0, 1/2)$ by running the commands
   `D[x^2 + y[x]^2 == (2x^2 + 2y[x]^2 - x)^2,x]` and
   `D[x^2 + y[x]^2 == (2x^2 + 2y[x]^2 - x)^2,x] /. y[x] -> 1/2 /.x -> 0`

   Note some important details here. Mathematica can't figure out that `y` is a function of `x` instead of a constant unless we tell it, so we write `y[x]` instead of `y`. We can have Mathematica automatically substitute for us, but it matters that we do `/.y[x]->1/2` before `/.x->0`. Why? Try it the other way and see what happens.
   (c) Plot the tangent line to the cardioid at that point in Mathematica.
   (d) What do you expect to happen if you try to find the tangent line at $(0, 0)$? Are you right? What does Mathematica say?
   (e) Looking at the graph, what do you think is the tangent line at the point $(1, 0)$? Can you get this from your derivative formula? Try computing the (implicit) derivative with respect to y instead of x. What happens?
3. (a) Plot the "devil's curve" `y^2(y^2 -4) == x^2 (x^2 - 5)`
   (b) Compute the derivative at $(0, -2)$.
   (c) Plot the devil's curve and its tangent line simultaneously.
   (d) Run the command
   `ContourPlot[y^2(y^2-4) - x^2(x^2 -5),{x,-5,5},{y,-5,5}]` What happens? Why?
4. (a) Plot `(x^2 + y^2 -1)^3 - x^2 * y^3 == 0`
   (b) Check that $(1, 1)$ is a solution to this equation, and compute the derivative at $(1, 1)$.
   (c) Plot the tangent line.
   (d) Now try plotting without the equals sign, as in (3).
5. (a) Plot `Sin[x^2 + y^2] == Cos[x * y]` from -5 to 5.
   (b) As before, replace the `==` with a `-` sign.

Just Because They're Pretty

1. Some other functions to try:
   - `Sin[Sin[x] + Cos[y]] == Cos[Sin[x * y] + Cos[y]]`
   - `Abs[Sin[x^2 - y^2]] == Sin[x + y] + Cos[x * y]`
   - `Csc[1-x^2] * Cot[2-y^2] == x * y`
   - `Abs[Sin[x^2 + 2 * x * y]] == Sin[x - 2 y]`
   - `(x^2 + y^2 - 3) Sqrt[x^2 + y^2] + .75 + Sin[4 Sqrt[x^2  + y^2]]`
     `Cos[84 ArcTan[y/x]] - Cos[6 ArcTan[y/x]] ==  0`
2. Try replacing the `==` signs with `-` signs.
3. Look at the examples on the Wolfram Alpha page
   `https://www.wolframalpha.com/examples/PopularCurves.html`