Mathematica Assignment #2

In this assignment we'll use Mathematica to do implicit differentiation, and integration. Then we'll take advantage of its programming abilities to create a program that finds roots of functions using Newton's Method.

Due by 9am Friday 12/2. Submit your file using Blackboard.

A. Implicit Differentiation

The Mathematica function `D` can also be used to do implicit differentiation, although the syntax is a little ugly. Rather than differentiate an expression, we differentiate a whole equation. Mathematica uses two equals signs in its equations. And then we need to tell it which variable is a function of what. The expression

```
D[y, x, NonConstants -> {y}]
```

is Mathematica's way of saying dy/dx or y'(x).

1. First we use `D` to differentiate the equation, then we use `Solve` to solve for dy/dx. Another useful piece of syntax: the percent symbol `%` refers to the last line of output. Look through the next lines of input/output very carefully, and make sure you understand how the syntax works. What equation are we implicitly differentiating, and what is the answer?

```
D[x^2 + y^2 == 9, x, NonConstants → {y}]
```

$$2\,x + 2\,y\,\text{D}[y,\,x,\,\text{NonConstants} \to \{y\}] == 0$$

```
Solve[%, D[y, x, NonConstants → {y}]]
```

$$\left\{\left\{\text{D}[y,\,x,\,\text{NonConstants} \to \{y\}] \to -\frac{x}{y}\right\}\right\}$$

2. Implicitly differentiate the following equations and solve for dy/dx.

(a) $x^2 y + y^3 = 4$
(b) $x^3 + y^3 = 6xy$
(c) $\sin(x + y) = y^2 \cos(x)$

3. Now let's look at some plots of these functions. The way to implicitly plot an equation (that we can't put into y(x) = ... form) is using `ContourPlot`. Look at the help menu and examples for `ContourPlot`, by evaluating the input `?ContourPlot` and clicking >>.

In a single window, plot the equation in 1(b) and the tangent line to the point (3,3). You should figure out the equation of this tangent line using your answer to 1(b), and paper and pencil. Since you are plotting implicitly with `ContourPlot`, you want to feed it the whole equation, like `y == 2 x + 5`, rather than just the right-hand side `2 x + 5` as you do with `Plot`. Also, you'll need to figure out how to plot two equations in the same window, and what x and y bounds the window should have so that you can see both curves.

4. Do the same thing with the equation in 1(c), and the tangent line to that equation at the point (π, 0). That is, calculate this tangent line (by hand and using the results of 1(c)) and then plot both in the same window using `ContourPlot`.

B. Integration

Yes, Mathematica can calculate definite and indefinite integrals! The function to look up is `Integrate`.

5. Choose three definite integrals from old homework assignments, and calculate them using Mathematica.

6. Choose three indefinite integrals from old homework assignments, and calculate them using Mathematica.

C. Newton's Method for finding roots of functions

Newton's Method is a way of finding roots, or zeros, of functions. It is discussed in Section 3.6 of your textbook, which we are skipping in class but you're welcome to read. If $f(x)$ is a function, a root is a value c so that $f(c) = 0$. These are the x-intercepts of $y = f(x)$.

We know how to find the roots of a quadratic. But what about a cubic like $f(x) = x^3 - 2x - 5$? Or what if we want to solve the equation $\cos(x) = x$? We can rewrite this as looking for the roots of $g(x) = \cos(x) - x$, but how do we find these roots?

Newton's Method starts with a function $f(x)$, and a guess $x_1$for a value close to a root of $f(x)$. We make the tangent line $L_1(x)$ to $f(x)$ at x_1. **The key idea: since the tangent line to *f* at x_1 is close to the graph of *f*, the root of the tangent line is close to the root of *f*.** We know how to find the root of a line -- this is just the x-intercept and easy to find. So we find the zero of L_1 and take this as our second guess x_2. In most cases, x_2 will be closer than x_1 to the actual root of $f(x)$, and we can repeat (i.e. iterate) this process: x_1, x_2, x_3, x_4,... until our numbers stabilize and we've found (a numerical approximation for) the root of $f(x)$.

This method, or a version of it, is how calculators and computers find roots.

The rest of this assignment leads you through making a Mathematica function from scratch that automates this process.

1. First let's look at an example. Given the following function and three of its values, what theorem from class guarantees that there must be some root r, with $f(r) = 0$, such that $2 < r < 3$?

```
f[x_] := x^3 - 2 x - 5;
f[1]
```

```
-6
```

```
f[2]
```

```
-1
```

```
f[3]
```

```
16
```

2. Let's use $x_1 = 2$ as our first guess. Calculate the equation for the tangent line L_1 to f at 2. (You can do this with pencil and paper). Using `Plot`, in a single window plot the graphs of f and L_1 nearby the value 2, showing that the zero of L_1 is close to the zero of f. (If you want to refer to the function `f` defined above, you need to first evaluate that cell where `f` is defined.)

3. On paper, find the zero of L_1, and let's make that our second guess x_2. (Answer: $x_2 = 2.1$.) Now, I claim that a formula for x_2 in terms of x_1 is given by:

$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$,

where that denominator is the derivative of f evaluated at the first guess. Think about how you found L_1 and x_2. Can you derive this formula? Can you explain in words where it comes from?

4. Using our second guess x_2, iterate this process to find a third guess x_3.

5. Using the third guess x_3, find a fourth guess x_4. Are the guesses converging to a stable value?

6. Now comes the hard part. Look at the function `NextGuess` that I define below. Describe in words what this function does. What are its inputs? What does it output? (Hint: don't get thrown off by the `N` function wrapper. Look up the function `N` to see what it does.)

```
NextGuess[f_, xn_] := N[xn - (f[xn]) / (f'[xn])]
```

7. Evaluate `NextGuess[f, 2]`. For this to work, make sure you've evaluated (i.e. clicked within and then hit SHIFT-ENTER) the cell above where we defined the cubic function `f`, and the cell where I define `NextGuess`. If you don't actually run these cells, their contents aren't in Mathematica's running memory.

8. You should get 2.1. Iterate with `NextGuess[f, 2.1]`, etc, and see if you can get back your third and fourth guesses.

9. Let's start with a slightly different starting guess, say $x_1 = 2.3$. Use `NextGuess` to calculate the second, third, and fourth guesses. Are they converging to a stable value?

10. Evaluate and explain what the following code does. (Hint: the `100` makes sure it doesn't go forever.)

```
FixedPoint[NextGuess[f, #] &, 2, 100]
```

```
2.09455
```

10. Finally, can you make sense of the following function? What are its inputs? What is its output?

```
NewtonMethod[f_, x1_] := FixedPoint[NextGuess[f, #] &, x1, 10]
```

11. Evaluate `NewtonMethod` for f twice, with initial guesses 2 and 2.3. Then use the `Solve` function to solve $f(x) = 0$, and see if your answers all agree. (The function `N` may be useful here.)

12. Use our new function `NewtonMethod` to find all the solutions to $\cos(x) = x$. (How do you rewrite this as a question about roots to a new function `g[x_]:= something`? How do you find all the roots, not just one of them?) Check your answers using `Solve`.