Optimizing with Calculus

- Introduction
- Challenges
- Programming a Numerical Derivative

Introduction to Calculus Optimization

Many of the programs written in the first two units could have been done (in most cases, done better) with calculus.

Some examples are:

- finding function zeroes using secant method (when Newton’s method is better)
- guessing at the limiting slope for sawtooth method (versus finding it with derivatives)
- finding maximum and minimum values (without referencing derivatives or second derivatives)

Introduction to Calculus Optimization

In this unit, we will be revisiting many of these ideas using calculus as a tool.

In addition, we will be extending calculus concepts into higher dimensions (ie, more than one variable) and applying that to find maxima and minima in 3D or higher.

Challenges with Programming

The biggest challenge with programming a computer to “do” calculus is that computers are not abstract thinkers. They can do operations on *numbers* very quickly, but do not understand *functions* as an abstract concept.

For example, 4^2 makes perfect sense to a computer.

However, x^2 does not make sense unless you input a value of x first.

Challenges with Programming

Try typing these in:

```
4^2               16
x^2               ERROR: x not defined
x = 4             4
x^2               16
f(x) = x^2        f(generic...)
f(a)              ERROR: a not defined
f(x)              16
```

Challenges with Programming

The reason the errors happened is because the language knows perfectly well how to square a number, but *unless x is a number* any operation on x is meaningless.

Computers seem really smart, but they would not pass Algebra 1.

Programming a Derivative

For this reason, most of the rules that you learned for derivative shortcuts will not be useful to a computer. For example,

the derivative of $ax = a$

the derivative of $x^n = n \cdot x^{n-1}$

Both rules require the viewer to see the left side (“ax” or “x^n”) as a function with different, separable parts. The computer sees them as a single number, or not at all.

Programming a Derivative

This is why programming languages (including the ones on graphing calculators, and Julia) can only calculate a derivative when given a value of x. They do it using the *definition of derivative*:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

(…and you always wondered why you had to learn it!)

Practice Problem 1

Using the definition of derivative (without the limit part), calculate the following. Use Julia for the math.

a. $f(x) = x^3$, $x = 3$, $h = 0.5$
b. $f(x) = 3x$, $x = 5$, $h = 1$
c. $f(x) = 5x^2$, $x = 2$, $h = 0.5$
d. $f(x) = 5x^2$, $x = 2$, $h = 0.1$
e. $f(x) = 5x^2$, $x = 2$, $h = 0.01$
f. $f(x) = 5x^2$, $x = 2$, $h = 0.001$
g. (what is the actual derivative of $5x^2$ at $x = 2$?)

Practice Problem 2

If you didn’t already do so for practice problem 1, write a little function g(x, h) that evaluates the derivative of f, given x and h. Use it to find derivatives of $5\sqrt{x}$ (aka `5*x^(1/2)` or `5*sqrt(x)`) when x = 12 and h is…

a. 0.1
b. 0.01
c. 0.001
d. 0.0001
e. 0.00001
f. 0.000000000001

Programming a Derivative

This procedure, of calculating the derivative using actual numbers plugged into a function, is called finding the “numerical derivative”.

The definition of derivative is only one way to do it. Another way would be to use secants:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h}$$

This method can be more effective for small values of h.

Programming a Derivative

When Calculus was invented, derivative shortcuts were important because Newton and friends did not have calculators; they would have had to do this by hand:

$$\frac{\sqrt{5.0000001} - \sqrt{5}}{.0000001}$$

…which is why the shortcuts got discovered pretty quickly. But for most applications, computers can do the job just fine.

Programming a Derivative

As you went through decreasing values of h with the same x in the practice problems, hopefully you noticed two things:

1) the values of the numerical derivative converged towards the “correct” value of the derivative
2) for very small h (but not too small), the numerical derivative was really close to the “correct” value of the derivative.

Of these two, the second is the most important, because computers can’t recognize patterns of convergence, but they can calculate numerical derivatives for very small values of h.

Practice Problem 3

Write a program nderiv(f, x) that calculates the derivative of a function f given an x-value x. You will need to choose h carefully.

test and save your program!

So, What About the TI Nspire CAS?

If you have worked with a TI-89, Nspire CAS, or similar calculator, you know that it can take "real" derivatives, for example

derivative($4x^3$)

will return $12x^2$

These calculators use the same types of languages with the same limitations, so they're not *actually* taking the derivative the way a human would. Someone programmed them to do it. How might this have been done?