Unit IV
Roots and Minima

Unit IV
Roots and Minima

This activity guide looks at methods of solving equations and finding minima. We will look first at roots of equations in one variable, and then minima of functions in two dimensions. The former problem occurs frequently in physics, for example when one applies initial or boundary conditions to a trial solution of a differential equation, and then has to solve the resulting equation to find the parameters of the solution. Or, in quantum mechanics, one often finds approximate solutions by minimizing the total energy of the quantum mechanical system. In either of these cases, it is not unlikely that one will have to deal with transcendental equations, for which no analytic solutions exist. In that case, it is essential to have numerical techniques for finding roots and minima. And while this seems like it should be a simple project (indeed many hand-held calculators can perform this task), one often confronts pathological cases that are resistant to straightforward solutions. We will talk a bit about how one goes about finding roots, and when and why the whole process can go awry.

Guidebook Entry IV.1: I Repeat: "Iteration"

First, I want you to appreciate how few equations we really can solve directly, even with a typical calculator. First, let's take one that we *can* solve. Find the solution to this equation:

$$\cos(x) = 0.5,$$

that is, what value of x makes this equation true?

Now, try to solve the following equation:

$$\cos(x) = x.$$

Describe your attempts. What is your estimate of the answer? How accurate do you think it is?

Now take your calculator (set to radians for the trig functions) or Excel, and repeatedly take the cosine function. That is, start with any number you like, take the cosine of that number, then the cosine of the resulting number, then the cosine of that number, and so on. Show some of your results. How many iterations does it take for the result to stabilize to five decimal places?

Argue why this is a solution to the equation cos(x)=x.

This simple iteration techniques can be used for a number of transcendental equations. For example, let's consider the equation

$$(0.4\,x)(\cos(x)) - \sin(x) = 0\,.$$

Before you try to find any roots, it's a good idea to get an idea of where the roots might be. So, first, simply plot the left-hand side of this equation to see that there *are* roots, and roughly where they might be. Where roughly are the first two roots greater than x = 0?

You should have discovered several roots. Are there an infinite number of roots? Somtimes inspection of the function in terms of the intersection of two more familiar graphs is useful. Regraph this equation as *two* functions on the same graph, y = .4 x and y = tan(x). Explain where the solutions to this equation exist on this graph. Can you now answer the question of the number of roots with authority?

Try rearranging this by pulling the x (from the 0.4 x) over to one side, and all the rest (sin and cos) of the equation to the other side. Recall that sin(x)/cos(x) = tan(x). Now try iterating this equation to see if you can find a solution that way. Describe what happens.

Chances are you found the iteration above was quite unstable. Try rearranging the equation by instead inverting the tangent to an arctangent, and isolating that x. You may wish to know that the arctangent function in Excel is ATAN(). Iterate this form of the equation and describe what happens.

You should have quickly converged to the root at zero. This is because the arctangent gives only the result closest to zero; whereas many angles give the same tangent. To find other roots, say the root around 4.0, add the period of the tangent (π given by pi() in Excel-speak) to the result of the arctangent operation. What is the value of this root near 4.0 accurate to five decimal places?

There are other ways in which one can rearrange this equation to produce other iteration formulae. Some of these converge quickly, some slowly, and some diverge horribly. Worse yet, it is difficult to predict in advance which of these will converge, and which will diverge. Often, successful convergence depends on a good initial value, which emphasizes the importance of graphing the function first.

We have made a good first start in the root finding business, and the technique of writing a simple iterative configuration of the equation can be a fast way to solution. It is critical, especially when computer or calculator graphing is so quick and easy, to use the machine to look at the overall behavior of the function first to avoid unpleasant surprises, such as trying to find a root where none exists. We may also find ourselves finding roots repeatedly of very similar equations, in which case finding a simple iterative algorithm will be of much greater utility than a simple application. Forman Acton's *Numerical Methods that Work* takes considerably more time in investigating the pros and cons of different configurations, and can give some insight into why some iterations work well, and why some don't, although it is not always easy to predict. It is also sometimes useful, at least conceptually, to break the iteration into two steps (e.g. in the simple case above, one part being the inverse tangent, the other being the scaling by 0.4). Then we can envision the convergence process as consecutive motions in the x and y directions to intersect different curves, such as the line and the tangent function you graphed on the same graph on the previous page.

In the next exercise, we will develop a more general tool that is somewhat more cumbersome to apply. The notion of a general tool suggests more general application, which is true, although there are still risks in its application, so it is always prudent to look at your function first!

Guidebook Entry IV.2: Newton's Method

A common theme in numerical methods is the repeated application of the Taylor Series. Root finding is no exception. We know that a function can be approximated as

$$f(x) \approx f(x_o) + f'(x_o)(x - x_o) + ...$$

If we just look at this first order approximation, we can easily invert this to find where the zero of this approximate function is. The value x_o is just a starting value, some point that we hope is close to the actual root we want to find. That is, simply take the right hand side of this equation, set it equal to zero, and then solve for x. Do that below.

As we know well, the Taylor series is a good approximation very close to the the reference point x_o, but can be very bad as we get farther away from that point. This is especially true if we are taking only the linear term. So, what we then must do is use the result you got from the formula derived above to use as a better approximation to the root, that is, as a second x_o, and reapply the algorithm. In some cases, we may have an analytic value for the derivative. Let's choose an example where that is true. Let's return to our solution of cos(x) = x, only now consider this to be finding a zero of the function f(x) = cos(x)-x. What is the analytic derivative of this function?

Let's now put Newton's method (the name for this technique) to work. Create a spreadsheet that has three columns, the first being sequential x estimates of the root, the second being f(x) at that x value, and the third being the derivative at that x value. Now make the entry for the formula for the next estimate of the root, and fill these expressions down. Do you converge to the same value you got by simple iteration? Is the process faster?

Explain in geometric terms, using a graph of the function cos(x)-x, how Newton's method works to get from one approximation of the root to the next. Check your answer with your instructor.

Given your answer to the question above, or by looking at the formula you developed, can you suggest an initial point x_o which will cause Newton's method to make a terrible estimate of the root? Try your guess, and describe what happens.

Sometimes we have functions where it is cumberson to write an analytical solution to the derivative. We can of course write a numerical approximation for the derivative by looking at the function at two x points close to one another. This method, while fundamentally the same as Newton's method, is usually referred to as the secant method. Try this for cos(x)-x. Instead of your derivative column, make the third column the value of the function at x+.01. Use this to create a new formula for the next estimate of the root. How quickly does this method converge?

What about that 0.01 I used for the derivative approximation? What are the limitations of making this a lot smaller? A lot larger? Does the value of the derivative enter into this?

We've seen that we can get into trouble with Newton's method if the absolute value of the derivative gets small. We can avoid a lot of this if we graph first, and make sure our initial guess is a good one. However, this might be difficult if the function is a rapidly varying one, or if we need to design a system that will run more automatically, without requiring so much supervision. One method that won't fly away from a root is the method known as false position. We'll describe this in the next Guidebook Entry, although we won't use Excel to actually calculate with it. That is because it requires making some logical decisions which require more advanced Excel skills than are worth developing at this point, although it is straightforward to incorporate either in Excel or in a more formal programming language.

Guidebook Entry IV.3: False Position: Interpolation Revisited!

Sketch a function that has a root in the space below, most any function will do, as long as the function passes through the x axis.

Now imagine that we have two estimates of the root, one which provides a positive value for the function, and the other which provides a negative value for the function. Use your knowledge of linear interpolation to produce a better estimate of the root, which we can call x_3, in terms of x_1, x_2, $f(x_1)$, and $f(x_2)$.

To make sure that we don't lose the root, the method looks at the sign of $f(x_3)$, and then replaces the previous estimate, either x_1 or x_2, which is on the same side of the root. This way, we make sure that we continue to straddle the root. This method converges more slowly than Newton, in general, but also is very reluctant to fly away. This can be a problem in cases where a root actually doesn't exist! For example, make a graph of the tangent function from zero to 6. Now imagine that we have an automatic root finding routine that looks for a root between 3.0 and 4.0. Describe what happens here. Is there really a root here?

Can you think of a technique for detecting this "false root" and moving on? You might want to discuss this with your instructor.

Finally, let's look at a particularly tough kind of root, a so-called double root. This also leads us to a brief view of the related task of finding minima (or maxima) or functions.

Guidebook Entry IV.4: Double Trouble--Finding Minima

Show that the function f(x)=1-cos(x) has a root at x = 0 by simply evaluating f(0).

Let's see how well we can find this root. Let's choose to use our fastest converging routine, Newton's method. Choose a starting value of perhaps 0.1, pretty close to the root. How quickly does this converge?

Graph the function. What characteristics of this root make Newton not work work very well?

How well would the false position method work on this root? Explain?

This sort of root is often called a double root. The reason for this relates to polynomials, where the function can be written as

$$f(x) = k(x - x_a)(x - x_b)(x - x_c)...$$

where x_a and so forth are the roots of the function. A polynomial such as $f(x)=x^2$ can be written as (x-0)(x-0), so we call this a double root at 0. Explain how this function has the same nasty features as 1-cos(x).

We can, however, disentangle this problem by looking not at a zero in f(x), but rather at the zero in the derivative. This is a very dangerous procedure, because we need to know that this is a genuine multiple root, where the derivative goes to zero at the root. It is completely possible that there might be two separated roots, as one finds for $f(x) = x^2 - .001$, or no roots at all as for $f(x) = x^2 + .001$. But, given that the double root is real, use this technique to find the zero of f(x) = 1-cos(x) using Newton's method on the derivative function f'(x).

This method, while dangerous for finding a root that we think is multiple, but in fact might not be, is very useful for finding a minimum or maximum of a function. You can use this either with analytic or numerical values for the derivative. Clearly the constant offset to a function disappears when you take the derivative, so this is just sensitive to maxima and minima. Apply this to find the first minimum of the function f(x) = sin(2x)-x.

Finally, let's learn how to apply Mathematica to find roots. Mathematica hides all of the details of the root finding algorithms, so we just hope all goes well. Even when using Mathematica, however, it is always a good idea to plot the function first, so you know roughly what to expect.

Guidebook Entry IV.4: Using Mathematica to Find Roots

First, let's try the simple case of x=cos(x). Mathematica can try to find algebraic solutions to a variety of equations. To reassure yourself that you really can't solve this one, try

Solve[Cos[x] == x , x]

where the first argument is the equation to be solved, with a double equals sign, and the second argument is the variable that one wants to find. Does Mathematica like this one any more than you did?

Now, let's allow Mathematica a chance to find a root for this with the command

FindRoot[Cos[x] == x , {x , 1 }]

where the first argument is the equation, and the second argument tells both the variable to be found and a guess for where to start looking.

Now try this on our other friend, $(0.4\,x)(\cos(x)) - \sin(x) = 0$. Can you verify our previous result for the root around 4?

The equation f(x) = x - tan(x) has a root at x = 0, which you can easily verify. How well does Mathematica find that root? Why is there difficulty here? Would you have any indication from Mathematica that there may have been a problem?