# Lab 3

In this lab we will use center differencing and Richardson extrapolation in a particular case to calculate derivatives. We will also find (again!) that even when we do have an analytic expression it is not always useful numerically.

A problem we always encounter is how to access the accuracy of our numerical results. If we know the “true” value (as we should when we are first testing our code, that is, we should always have a test case with a known result when possible) then we can calculate the relative error, |1-(estimate)/(true)|. When we do not we often just calculate the absolute error |(estimate)-(true)| though we need to be careful in how we interpret this. Typically we would want to normalize this to the magnitude of the expected result but we will not always do so.

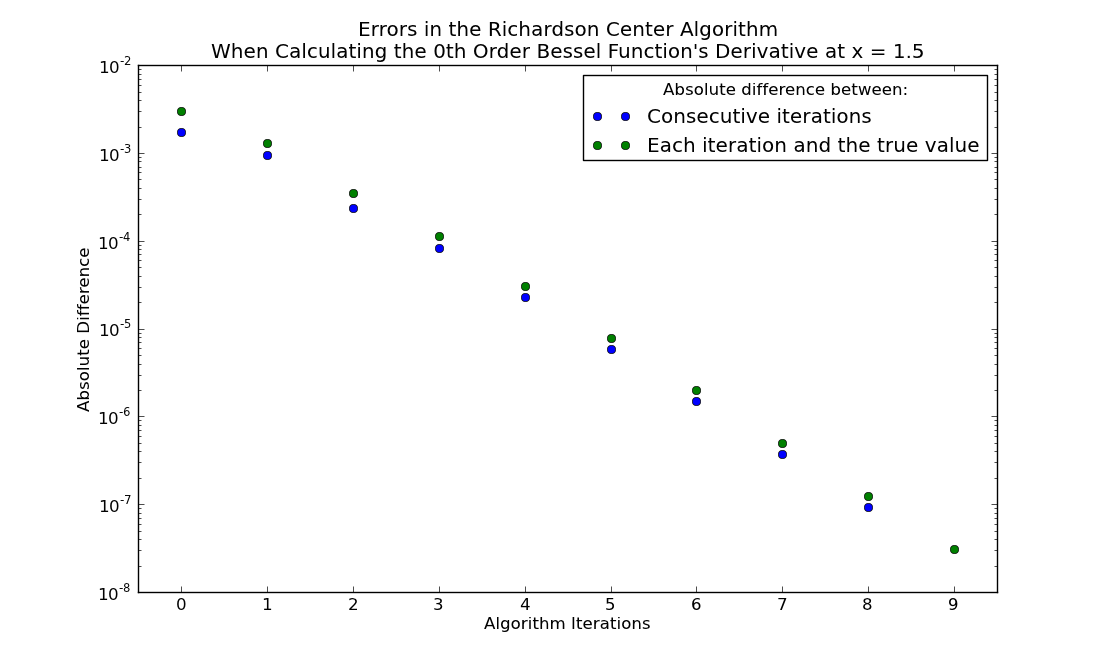
Our example function is the Bessel function. If you have not encountered it yet, you will. It is one of a large class of “special functions” that we encounter in physics. A wide range of these functions are available in scipy.special and we will have occasion to use a few of them throughout the semester. Special functions have many useful properties that allow us to manipulate them mathematically; this is precisely why we define them as “special”! The example seen here is to write an strange seeming derivative of a special function in terms of special functions. We can then use optimized routines for evaluating these special functions making computations easier to perform. We are familiar with this idea from our use of sines and cosines. We know many ways to manipulate them algebraically and often use their properties to simplify calculations.

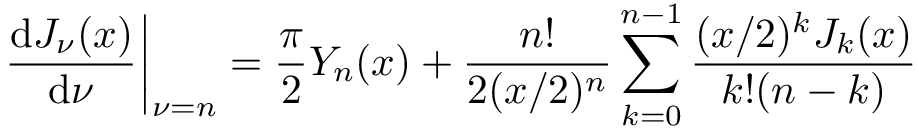
When we see a sum as in part 2 our first thought, probably, is to write a loop to calculate it. Tools like numpy and Matlab perform “vector” processing and this is far more efficient than writing our own loops. You have seen that we can pass an array to most (but not all) functions and they will perform the calculations on all values. For example, scipy.special.j0(x) returns for all values of we pass to it. To calculate the sum in part 2 it is more efficient to calculate all the terms in the sum at once and then use the sum() function to add them. It is worthwhile to try calculating the sum in this manner. For the problems is this lab the time differences are small but we will see how important it can be in future labs. The basic rule when using these types of tools is to never write a loop for a calculation if you can avoid it. As we saw in class basic operations for which we would normally write a loop have functions defined in numpy to perform them. Functions such as diag(), diff(), sum(), prod(), *etc.* make avoiding writing loops easy. We can learn about these by poking around in ipython using its introspection capabilities and by skimming through the numpy book referenced in the course syllabus.

1. Throughout physics we encounter many special functions, one set of these being the Bessel functions. We will use them as an example in this lab. The Bessel functions are defined for any real number order,, represented as . We can access it as scipy.special.jv. Since is a continuous variable we can take derivatives with respect to it. [Note: We are taking the derivative with respect to , not !] For this part of the problem we will use the fact that where is the ordinary Bessel function of the second kind of zeroth order. Naturally it is also available in scipy.special. In fact a number of variants exist including those for when is an integer or the special values 0 and 1. In general it is best to use the version most applicable to the case you are calculating. For the following calculations use and .
   1. Estimate the derivative using both forward and center differencing and calculate the relative error from the known results. Do this for both and .

|  |  |
| --- | --- |
| Error for : | Forward Derivative 0.276020806036  Forward Relative Error: 0.540539162819  Center Derivative 0.603754908861  Center Relative Error: 0.00500299184508 |
| Error for : | Forward Derivative 0.429544948364  Forward Relative Error: 0.284984764675  Center Derivative 0.602466510322  Center Relative Error: 0.0028583394916 |

* 1. Use richardson\_center (available at <http://www.phys.cwru.edu/courses/p250/examples/richardson_center.py>) to estimate the derivative through tenth order. Estimate the errors (really the convergence of the algorithm) by subtracting the estimate from the order and the order. Also fcalculate the absolute error from the known result. Produce a plot of these errors as a function of order. For a plot such as this it makes sense to make a semilog plot. As usual in this case we want the errors (y-axis) to be plotted on a log scale. Again, to make a good figure the plot should well represent the properties of what is being plotted. In this case does it make sense to plot the errors as lines or points? Include the plot below.



1. The derivative may be evaluated at any **integer** . Analytically we can calculate this asEstimate the derivative with the parameters from the previous part but now for using Richardson extrapolation to some order (pick a “reasonable” order). Evaluate the sum given above and use this to calculate the relative error in your estimate using the sum as the “true” value of the derivative. To calculate a factorial, which is only defined for integers, the more general gamma function is used in math libraries. As is shown in the scipy.special.gamma documentation .

|  |  |
| --- | --- |
| Order of Richardson extrapolation and error: | Order of Richardson Extrapolation: 10  Relative Error: 3.8334574759 \* 10 ^ (-7) |

1. You may wonder why we are going through all this trouble when we have an exact mathematical formula that allows us to evaluate the derivative. Contrary to what you may expect it is not just to keep you busy. Repeat the previous part now for . You will find wildly different answers. Which are we to believe? To explore this individually calculate the two terms in the mathematical expression in part 2 and the value of . When you compare these what do you see? From this which method do you gives the correct answer? Explain why one method succeeds while the other fails.

|  |  |
| --- | --- |
| Best estimate of derivative: | The best estimate of the derivative is -4.84926471982e-53, as calculated by Richardson extrapolation. |
| Explanation: | The Richardson extrapolation’s estimate is, in this case, more accurate than the analytical equation in Part 2 because the two terms in the analytical equation are -1.02889154307e+51 and 1.02889154307e+51. These numbers are so large that they require a lot of precision to preserve any small difference their magnitudes may have. But the computer has clearly rounded off these numbers. As a result of rounding these enormous numbers, their difference has been made much greater (by orders of magnitude) than it actually is. |

Enter the names of all people who worked on this lab as a group. By including a name here you are verifying that said person was actively involved in the work and understands the material:

|  |  |
| --- | --- |
| Names or ids | John Dulin (jdd49) and Lucas Flowers (laf62) |