Finite Difference Derivatives

CLA365 TECHNICAL REPORT

Nina Brim Lawson Busch Yuxiang Chen

February 13, 2018

Abstract

In this report we begin by discussing methods for approximating the derivative of a function and the accuracy of each method. After selecting a method, we determine the best possible value δ to use for calculating an accurate estimate of of function's derivative. Finally, wrap up with an example of the accuracy and use of our derivative function, $D(f, \delta)$, by implementing Newton's Method for root finding using our derivative function.

1 The Derivative Function

1.1 Overview

We use an approximation function to determine the derivative of a function, as a function approximating a derivative provides accurate results for the derivative while avoiding the computational complexity of calculating the derivative. For derivative approximation, we discuss two possible choices to consider: the forward difference method and the three point centered difference method. We implement and test both methods to determine which gives the best derivative approximation.

The forward difference method is calculated by:

$$\frac{f(x+\delta) - f(x)}{\delta}$$

And the three point centered difference method is calculated by:

$$\frac{f(x+\delta) - f(x-\delta)}{(2 * \delta)}$$

1.2 Implementation

Implementing both the forward difference and three point centered methods is fairly simple, and the code for both functions is relatively similar.

The code for a forward difference approximation:

```
1 fd = function(f, delta=0.001) {
2     df = function(x) {
3       newx = (f(x + delta) - f(x)) / delta
4       return(newx)
5     }
```

```
    return(df)
}
```

And the code for the three point difference approximation:

```
D = function(f, delta=0.001) {
    df = function(x) {
      newx = (f(x+delta) - f(x-delta))/(2*delta)
      return(newx)
    }
    return(df)
}
```

Both implementations take in two inputs: a function f and an optional delta value *delta*, where f is a function of a variable, x, and *delta* is a step size away from x used to approximate the first derivative of f. If no delta value is given, *delta* will automatically be set to a default value of 0.001. Each function works by creating and returning a new function, *df*, which is a function that approximates the first derivative of f using step size *delta* for a given input value x. Where the implementations of the forward difference and three point difference methods differ is their calculation of the x value for the derivative, where each implementation uses its corresponding method to calculate the value of *newx*.

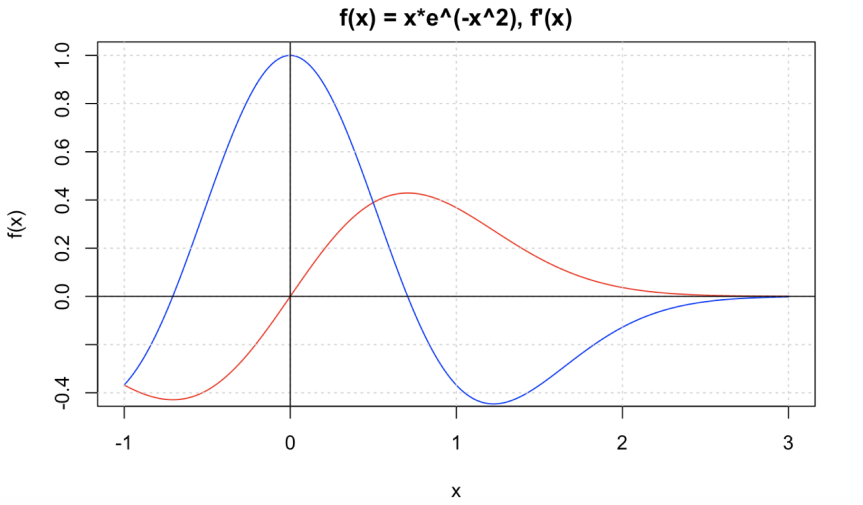

Figure 1: A plot of the functions $f(x) = xe^{-x^2}$ (red) and it's derivative $f'(x)$ (blue) found using our D function, to illustrate that the function works correctly.

1.3 Function Selection

To determine which function gives the most accurate derivative approximation, we calculate the error ratios for each method implementation to determine which produces more accurate results. To do this, we generate the derivative approximation function for a range of delta values on the input function e^x for both methods. We then calculate the error by setting $x = 1$, calculating the approximation of the derivative, and then calculating the error as $error = abs(1 - D(0))$, where D is the derivative approximation function. In the plot below, we show the comparison of the errors between the forward method and the three point difference

method. The red points demonstrate the error for the forward method and the black points demonstrate the error for the thee point difference method.

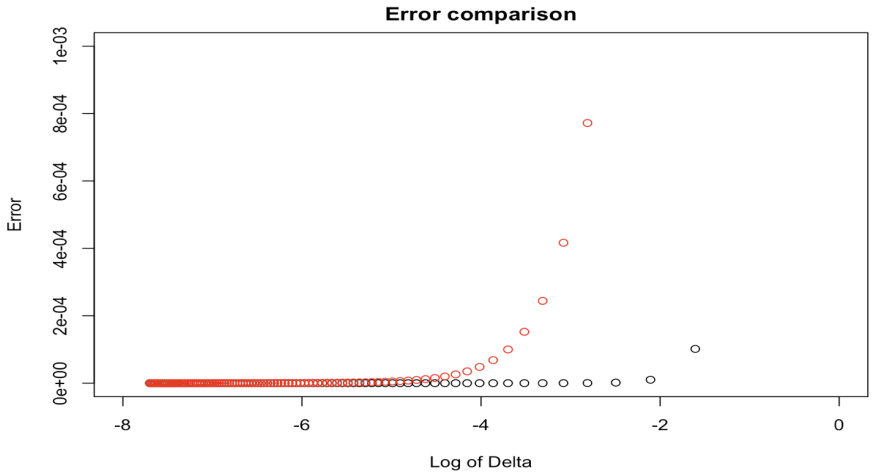

Figure 2: A plot showing the difference in error between the three point difference method (in black) and the forward difference method (in red).

From the graph in Figure 2 you can see that the three point difference method converges faster and more accurately to 0 than the forward method. This implies that the three point difference method gives a better, more accurate approximation of the derivative of f, as it has smaller error values. For this reason, we chose to implement the three point difference method as our D function for approximating derivatives. The code for D is:

```
D = function(f, delta=0.001) {
    df = function(x) {
      newx = (f(x+delta) - f(x-delta))/(2*delta)
      return(newx)
    }
    return(df)
}
```

Further, it is necessary to determine the degree of accuracy for D, and the *delta* value which results in the most accurate approximation of first derivative of the input function f. To determine the optimal *delta* value, we again generate approximation functions for a range of *delta* values. The plot below is a plot of 100 errors corresponding to 100 different *delta* values.

Figure 3 demonstrates that the error minimizes between *delta* values of $1 * 10^{-5}$ and $1 * 10^{-6}$. After $1*10^{-6}$, the errors begin to become unstable and grow up again. This implies that the optimal delta to use is approximately halfway between $1*10^{-5}$ and $1*10^{-6}$, or $1.5*10^{-5}$. So for the most accurate approximations, a *delta* value of $1.5 * 10^{-5}$ should be used.

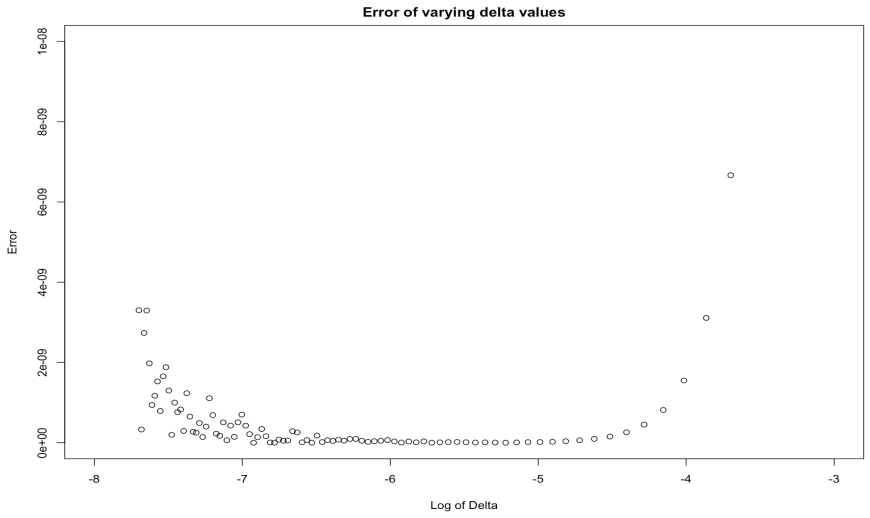

Figure 3: A plot of the error produced by varying values of δ. As we can see, once $\delta < 10^{-6}$, the error begins to increase.

2 Newton's Method

2.1 Overview

An example of how our above implementation of a derivative approximation function can be used is Newton's method. Newton's method is an iterative, root-finding method, which uses tangent lines to the function to narrow in on a root approximation. Given some function $f(x)$, and it's derivative, $f'(x)$, Newton's method can be described by the following equations.

$$x_0 = \text{initial guess}$$
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \text{for } i = 0, 1, 2...$$

2.2 Implementation

In order to write a function to execute Newton's Method, we first wrote a function to take derivatives, which can be seen in the above section. The first step of the Newton function shown below (written in R) is to call the D function. This takes the derivative of f, which is then used iteratively in performing Newton's method. Also note that the number of iterations is currently set to 40, though that may be excessive and a precise answer may be found using much fewer iterations.

```
1 Newton = function(f, x) {
2   df = D(f)
3   newX = x - f(x)/df(x)
4   numIters = 40
5   for(i in 1:numIters) {
6     newX = newX - f(newX)/df(newX)
7   }
8   return(newX)
9 }
```

3 Examples

3.1 Cycling

Example 1.15 in the textbook describes one case in which Newton's method fails to find the roots of a function. This example uses the function $f(x) = 4x^4 - 6x^2 - 11/4$ and an initial guess of $x_0 = 1/2$. When running our Newton function on this function f, with $x_0 = 1/2$, R returns the following output for the first 10 iterations. As we can see, the function will continue to fluctuate ("cycle") between guesses of $\frac{1}{2}$ and $-\frac{1}{2}$, never converging to the actual roots of the function. This is also illustrated in Figure 4 (below).

```
[1]  -0.500002   0.500002  -0.500002 0.500002  -0.500002  0.500002  -0.500002
[8]  0.500002  -0.500002  0.500002
```

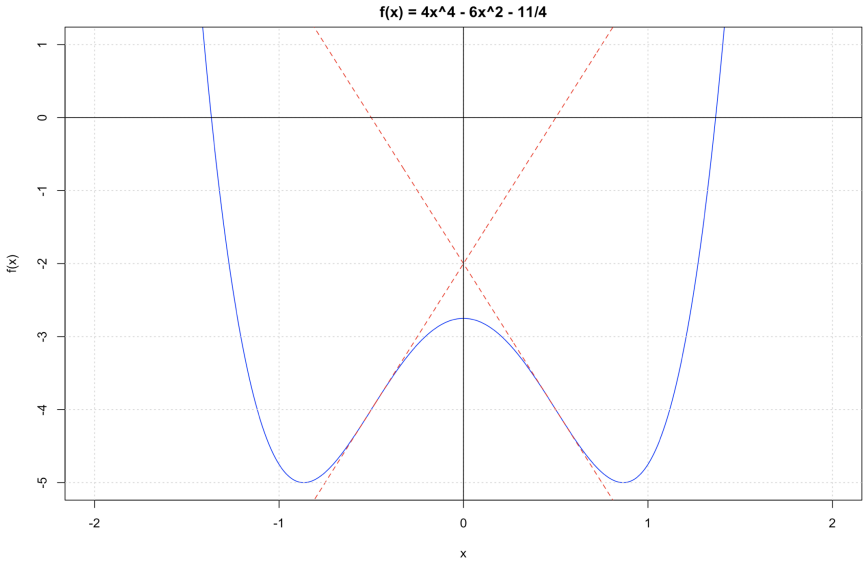

Figure 4: The function $f(x) = 4x^4 - 6x^2 - 11/4$ (in blue) and the two tangent lines (in red) approximated by Newton's method.

4 Conclusion

By using the three point difference method we can provide a very accurate calculation of the derivative for a given function f while saving the computational costs of actually calculating the derivative. This approximation can be useful in many applications, such as implementing Newton's Method for root finding, and enables implementations of complex algorithms to compute accurate results with much less computational intensity.