Nathaniel W. Morris

Numerical Analysis Final Project – Spring 2025

Automatic Differentiation (Forward Mode)

Automatic differentiation is a method in numerical analysis used for calculating derivatives of functions with a computer. It belongs to the same family of methods as numeric differentiation and symbolic differentiation. With it, one can compute derivatives both quickly and accurately.

Before discussing automatic differentiation further, consideration should be given to the other methods of computationally estimating derivatives. First among these methods is the method of numeric differentiation, represented by the function . That is to say, the derivative is calculated by finding the tangent line at a point in the function. However, this method tends to be ineffective, as it is subject to two forms of error: truncation error and roundoff error. The truncation error comes about due to the fact that the equation cannot be calculated with h = 0, as division by zero is impossible. Consequently, the equation is calculated with the smallest h possible. However, the difference caused by calculating the equation at a very small number instead of zero naturally causes error: this is known as roundoff error. Although this error tends to be small in initial calculations, as more and more calculations are made based on the original, slightly erroneous error, the error will propagate, causing worse and worse results further down the line.

In addition, this method is subject to roundoff error. This is error that is caused by the finite nature of computer memory, particularly by the limitations of float32 numbers. As h gets smaller and smaller, the roundoff error accumulates, and causes the number to be stored to be unfortunately cut off. Though there exist a few methods to reduce the error from this function, the error will inevitably exist. This is because the two forms of error, truncation error and roundoff error, are inversely related. As h becomes smaller, truncation error is reduced, but, due to finite precision, roundoff error increases. Conversely, to reduce roundoff error, h must be made larger, but this in turn causes roundoff error to increase.

Alternatively, one could employ symbolic differentiation. Through this method, one uses the rules of derivation obtained from calculus to make the computer solve the derivative. In other words, this is the computational equivalent of taking the derivative by hand. By doing this, one completely eliminates truncation error, and almost entirely eliminates roundoff error. However, there are still issues with this method, the foremost of which being expression swell. In short, while computers can very effectively compute derivatives by applying the rules of calculus, they will not simplify the results in any way, leading to expressions full of redundant or useless expressions, such as (1/sin(x))\*sin(x). Unfortunately, the computational cost of implementing a program to help the computer simplify the results generally tends to eliminate the benefits of using a computer to calculate the derivative in the first place.

This brings us, then, to automatic differentiation, which is a method of derivative calculating by which the function to be differentiated is broken up into the variables and elementary operators of which it consists. As visually demonstrated, an evaluation trace is used to keep track of the progress of this method upon an equation. This method provides the same answer as normally calculating the equation by hand. However, this method, effective though it is, the current method does not actually calculate a derivative, merely the original function.

To actually calculate derivatives, the forward mode of automatic differentiation must be introduced. To do so, the tangents are added to the evaluation trace, creating the forward tangent trace. It is by using this that the derivative is calculated, equal to the actual answer that would be obtained by calculating the derivative by hand.

Interestingly enough, this method can also be used to calculate Jacobian matrices. For any vector-value function, by selecting the inputs such that f: Rn->Rm has inputs a ∈ Rn such that x = a, then the output yj for j = 1, …, m will be such that very pass through the automatic differentiation forward mode function will generate a column of the associated matrix. While I am unsure of the practical applications of this, it seems to be an interesting feature of automatic differentiation forward besides just calculating derivatives.

This method has interesting applications in the neural network, large learning model, and machine learning world. This is because the gradient descent method used to train these large neural networks involves calculating large numbers of derivatives on a regular basis in the process. As such, a method that can quickly calculate large numbers of derivatives automatically is needed so that the training of these models is not stalled by unnecessarily complex equations. In addition, one generally assumes that the goal is for these large learning models and neural networks to be able to generate the most accurate output possible. As such, the method that they use to calculate their derivatives should be as accurate as possible. Although personally I am not particularly in favor of the popularization of AI based on neural networks and large language models, it is interesting to learn how techniques from numerical analysis are applied in their development and training. It will be interesting to watch as their development progresses, and to see if someone invents a more efficient method than gradient descent to train large language models and neural networks, possibly eliminating the need for automatic differentiation.