Exploring Autodiff and Finite Differences Using the Rosenbrock Function

Nicholas Rubino[1], Adrian Celaya[2], David T. Fuentes Ph.D[3]
Department of Imaging Physics, The University of Texas MD Anderson Cancer Center[2], Houston, TX

THE UNIVERSITY OF TEXAS MD Anderson Cancer Center

Background

- Computer models are growing increasingly complex as more accurate models are created. Optimizing parameters in these models relies on efficiently taking derivatives.
- Using existing machine learning framework, we can compute derivatives of physics based functions quickly.
- Our model is used to find the optimum flip angles for hyperpolarized magnetic resonance imaging (HPMRI).
- Using the Rosenbrock function as an example to see the benefit of this approach as a surrogate for a full physics based model in HPMRI.

Hypothesis

Using automatic differentiation to recover the parameters of the extended Rosenbrock function will be faster than numerical differentiation.

Methods

- The Rosenbrock function is used as a standard reference frequently used to evaluate the efficiency and accuracy of automatic differentiation frameworks.
- The function consists of two variables and is of the form
  - *f(x, y) = (1-x)^2 + (x - y^2)^2.*
  - Constrained by the circle *x^2 + y^2 <= 1.*
- This function has a distinct global minimum that is inside the banana shaped valley and thus is also known as the banana function.

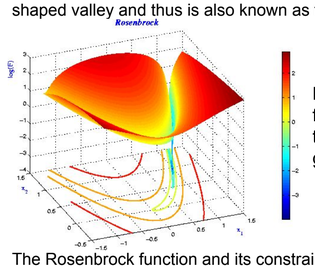

Figure 1. 2D Rosenbrock function. Note the spike under the valley that shows a clear global minimum.

The Rosenbrock function and its constraint can be generalized to any number of variables, which we used to evaluate the speed, the number of iterations, the number of function evaluations and the memory of numerical differentiation versus autodiff in finding the minimum.

Using MATLAB we were able to use the optimization toolbox to set up a Rosenbrock function of *n* parameters and time how long each method took to reach the minimum.

Numerical Differentiation (Sum of Finite Differences)

- Uses the limit definition of a derivative to approximate the value of a derivative for a given function.
- Computational complexity scales with the complexity of the function.
- Much faster than hand calculations but can be unstable.
- Works best with few independent variables as you must fully evaluate the function for each input.

Methods (continued)

Numerical Differentiation (Sum of Finite Differences)

- Computationally expensive, especially for problems with a complex function and a large number of input variables.
- Has problems with truncation error as well as rounding error due to the imprecise nature of floating point arithmetic with numbers very close to zero.
- Cost of computation as well as rounding errors exasperated when computing a gradient.
- Requires *O(n)* evaluations for an *n* dimensional gradient.

What is Automatic Differentiation (Autodiff)?

- A technique used to find the gradient of a function.
- Uses the chain rule to break down functions into elementary operations using dummy variables to store how each piece interacts with the others.
- Uses exact formulas and thus is to the accuracy of floating-point arithmetic.
- Similar to symbolic differentiation but evaluates the function and derivative at each node which eliminates the expression swell problem commonly found with symbolic differentiation.
- Has a forward mode and a reverse mode, which differ by how they obtain the gradient.

Forward Mode

- Applies the chain rule and evaluates the gradient to each basic operation to get a forward primal trace.
  - Primals are an ordered pair of a node element and its derivative. Denoted by variable with a dot on top.
- We then obtain a derivative trace of the function with respect to the independent variable that we seed.
- This method is preferred when the number of independent variables is much smaller than the number of dependent variables.
  - *f:R^n -> R^m such that m>>n*
  - *Time complexity of n*c*ops(f) where ops(f) is the operation count of f and c < 6 (~2 to 3)*
- The final gradient is then computed by multiplying the nodes together from the derivative trace.

Reverse Mode

- Starts with a forward pass where a trace of the primal nodes of the function is computed and each node is augmented with adjoint nodes, which store all intermediate variables and their connections in memory. This is called a Wengert list.
  - Adjoints are denoted by a variable with a bar on top of it.
  - The derivatives of each adjoint are not computed or stored during the forward pass
- Then there is a reverse pass where the Wengert list is followed and the derivatives of each adjoint node are computed.
- The derivatives of each adjoint are chosen with respect to a given dependent variable.
- This method is preferred when the number of dependent variables is much smaller than the number of independent variables as it produces a gradient of multiple independent variables using only one pass.
  - *f:R^n -> R^m such that n>>m*
  - *Time complexity of m*c*ops(f) where ops(f) is the operation count of f and c <6 (~2 to 3)*

Results

For 2, 4, and 10 parameters, finite differences was faster than autodiff. However, for all parameters, Auto reverse completed the objective function optimization with fewer function evaluations. Thus auto reverse is the best candidate for the HPMRI optimization function shown in Figure 5.

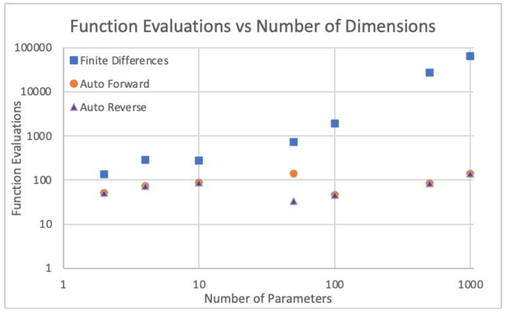

Figure 2. Function Evaluations versus number of dimensions.

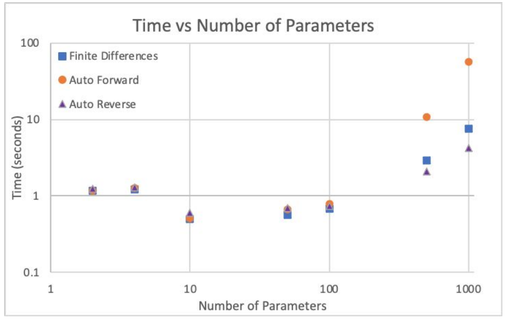

Figure 3. Evaluation time versus number of dimensions.

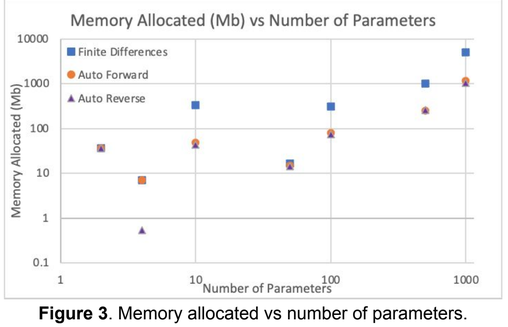

Figure 3. Memory allocated vs number of parameters.

Discussion/Conclusion

Auto reverse was the fastest of the three methods. Auto forward and reverse took the same number of function evaluations but auto reverse was faster, especially a larger number of parameters. Finite differences was not significantly slower in time, but took from 2.6 (2 parameters) to 453.6 (1000 parameters) times as many function evaluations compared to auto forward and auto reverse. As the Rosenbrock function is not very computationally complex, finite differences could keep up even with the discrepancy in number of function evaluations.

Using a more complex function, such as the one for HPMRI, (shown in Figure 5), finite differences should theoretically fall behind as the time needed for each function evaluation should be greater, thus making it much slower overall. We have not been able to get any of these solvers to work for the HPMRI function yet due to the high complexity of the function. We expect auto reverse to be the best option.

This is expected as finite differences is expected to be computationally expensive as the number of parameters increases.

One unexpected result was finite differences using more memory than auto reverse or auto forward. Auto reverse was expected to use the most memory due to the Wengert list. Finite differences using the most memory is due to the number of function evaluations.

Auto forward and reverse had an average of 1690 Kb allocated per function evaluation while finite differences has 163.3 Kb allocated per function evaluation.

$$
\begin{aligned}
-H(z) &= \int p(z)\ln p(z)dz \\
&= \int \sum_{i_N}\omega_{i_N}\sum_{i_2}\omega_{i_2}\sum_{i_1}\omega_{i_1}\frac{1}{\sqrt{\pi}^N}p\left(z|\sqrt{2}\sigma_{P_1}x_{i_1}+\mu_{P_1},\sqrt{2}\sigma_{P_2}x_{i_2}+\mu_{P_2},\sqrt{2}\sigma_{P_N}x_{i_N}+\mu_{P_N}\right)\ln p(z)dz \\
&= \int \sum_{i_N}\omega_{i_N}\sum_{i_2}\omega_{i_2}\sum_{i_1}\omega_{i_1}\frac{1}{\sqrt{\pi}^N}\frac{z}{\sigma_z^2}\exp\left(-\frac{\mathcal{G}^2(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})+z^2}{2\sigma_z^2}\right)I_0\left(\frac{z\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})}{\sigma_z^2}\right)\ln p(z)dz \\
&= \int \sum_{i_N}\omega_{i_N}\sum_{i_2}\omega_{i_2}\sum_{i_1}\omega_{i_1}\frac{1}{\sqrt{\pi}^N}\frac{z}{\sigma_z^2}\exp\left(-\frac{2\sigma^2y^2-2\,\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})z}{2\sigma_z^2}\right)I_0\left(\frac{z\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})}{\sigma_z^2}\right)\ln p(z)dz \\
&= \int \sum_{i_N}\omega_{i_N}\sum_{i_2}\omega_{i_2}\sum_{i_1}\omega_{i_1}\frac{1}{\sqrt{\pi}^N}\exp(-y^2)\underbrace{\frac{z}{\sigma_z^2}\exp\left(\frac{\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})z}{\sigma_z^2}\right)I_0\left(\frac{z\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N})}{\sigma_z^2}\right)\ln p(z)\sqrt{2}\sigma_z}_{=h(z)}dy \\
&\approx \sum_k\omega_k\sum_{i_N}\omega_{i_N}\sum_{i_2}\omega_{i_2}\sum_{i_1}\omega_{i_1}\frac{1}{\sqrt{\pi}^{N+1}}h(\sqrt{2}\sigma_z\,y_k+\mathcal{G}(\mathcal{K};\mathcal{P}_{i_1,i_2,i_N}))
\end{aligned}
$$

.**Figure 5.** Objective function for HPMRI.

References

Margossian, Charles C. (2019). *A Review of Automatic Differentiation and its Efficient Implementation.* 1-10.

Baydin, Atılım Güneş, et al. (2018). *Automatic Differentiation in Machine Learning: a Survey.* 1-14.

Lange, Robert (2019). *Forward Mode Automatic Differentiation and Dual Numbers*.

Canonical Duality Theory for Solving Minimization Problem of Rosenbrock Function - Scientific Figure on ResearchGate.

Cohen, William. *Automatic Reverse-Mode Differentiation*. 1-3

Bayesian Signal model for HPMRI

Griewank, Andreas. (1997). *On Automatic Differentiation.*