A comparison of automatic differentiation tools: Enzyme vs Zygote

Jack Hau, Zack McKevitt, Kasper Seglem

Problem Overview

1. How do the two leading automatic differentiation tools perform when differentiating algorithmic procedures?
2. What are the design considerations of Enzyme vs Zygote?
3. How do the design decisions of Enzyme/Zygote influence overall differentiation performance and correctness?
4. Which tool is better overall?

Background - Enzyme [1]

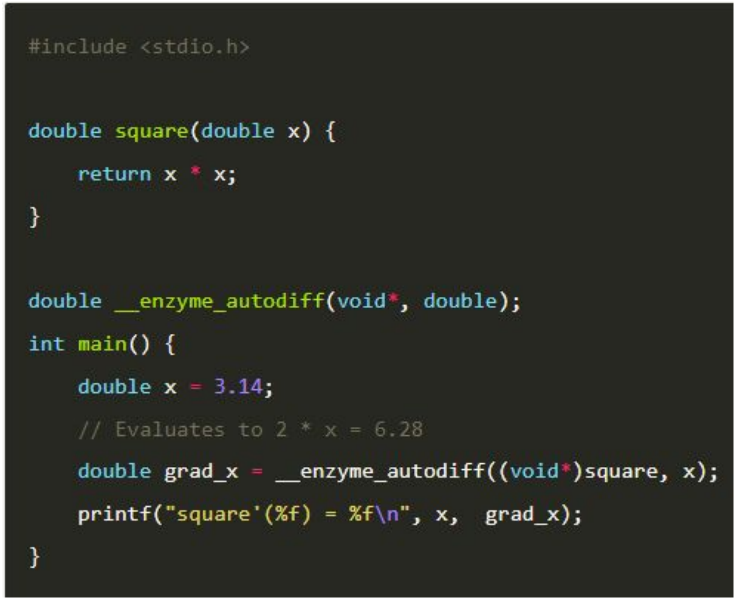
Design overview:

- Convert code to LLVM IR
- Optimize LLVM
- Differentiate LLVM IR code
- Optimize differentiated LLVM code
- Compile LLVM code

Compatible with many programming languages

Published at NeurIPS 2020

Can differentiate parallelizable GPU functions [2]



```c
#include <stdio.h>

double square(double x) {
    return x * x;
}

double __enzyme_autodiff(void*, double);
int main() {
    double x = 3.14;
    // Evaluates to 2 * x = 6.28
    double grad_x = __enzyme_autodiff((void*)square, x);
    printf("square'(%f) = %f\n", x,  grad_x);
}
```

Automatic Differentiation for C function with Enzyme

https://github.com/EnzymeAD/Enzyme

[1] - Moses, William et al. "Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients." *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2020.

[2] - Moses, William S. et al. "Reverse-Mode Automatic Differentiation and Optimization of GPU Kernels via Enzyme." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* Association for Computing Machinery, 2021.

Background - Zygote [3]

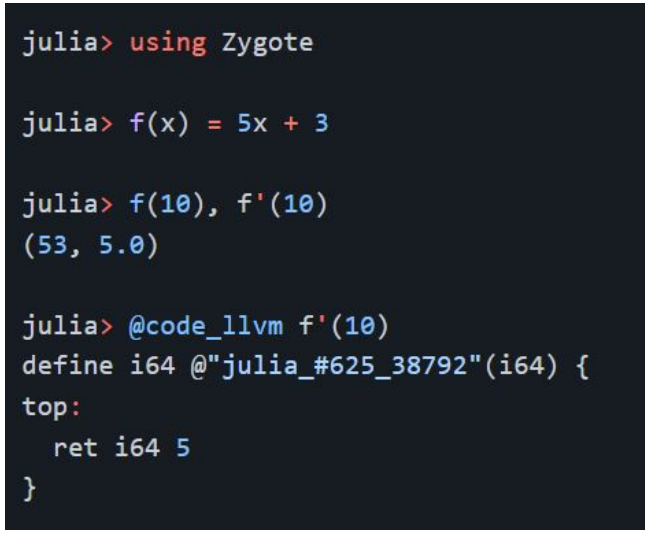
Design Overview:

- Perform source to source AD
- Convert differentiated code to LLVM
- Optimize LLVM code
- Compile LLVM code

Exclusive to Julia programming language

Augments Julia compiler to add new optimizations (SSA) when performing AD

Paper uploaded to arXiv in 2018 (not published)



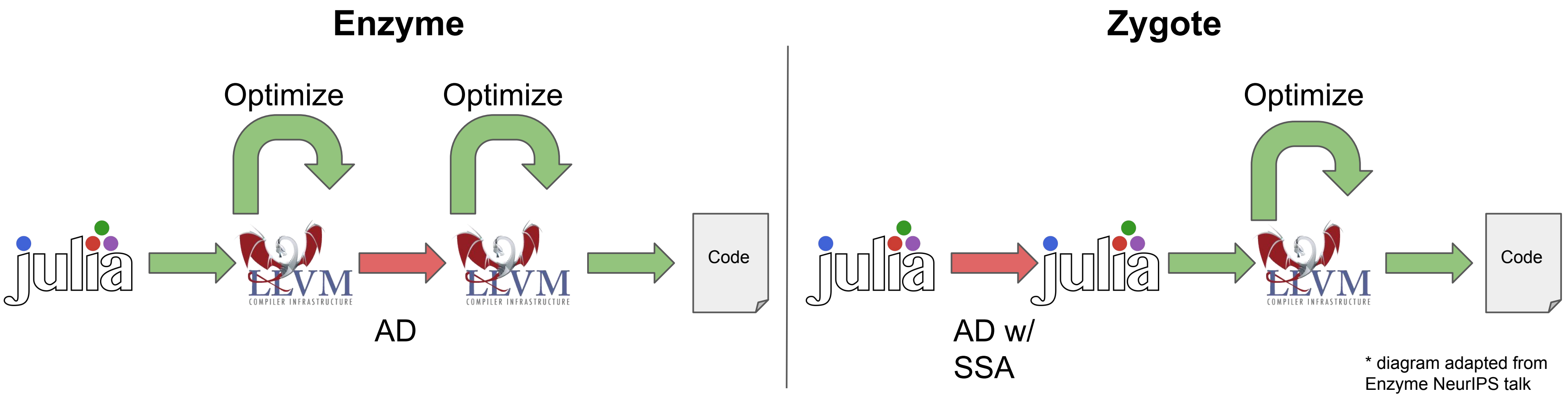
```
julia> using Zygote

julia> f(x) = 5x + 3

julia> f(10), f'(10)
(53, 5.0)

julia> @code_llvm f'(10)
define i64 @"julia_#625_38792"(i64) {
top:
  ret i64 5
}
```

Zygote differentiation with LLVM optimizations

[3] - Innes, Michael. "Don't Unroll Adjoint: Differentiating SSA-Form Programs." (2018).

https://github.com/FluxML/Zygote.jl

Revised Research Question



Does Enzyme’s optimized LLVM code perform better than Zygote’s optimized Julia code (using SSA)?

How did we test?

Tested timing and memory allocation of Enzyme and Zygote

- Timing - @timed
    - Used the returned value, time, and memory allocation for method call

Packages used for measurements

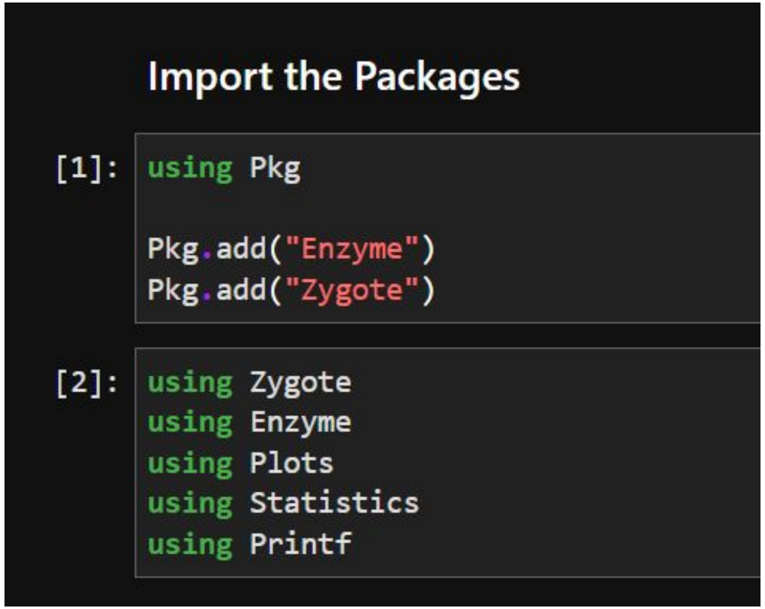
- Enzyme.jl, Zygote.jl, Statistics.jl

If anyone is interested in plotting our results

- Included Plots.jl

Packages for printing

- Printf - @printf


Import the Packages

```
[1]: using Pkg

     Pkg.add("Enzyme")
     Pkg.add("Zygote")
```

```
[2]: using Zygote
     using Enzyme
     using Plots
     using Statistics
     using Printf
```

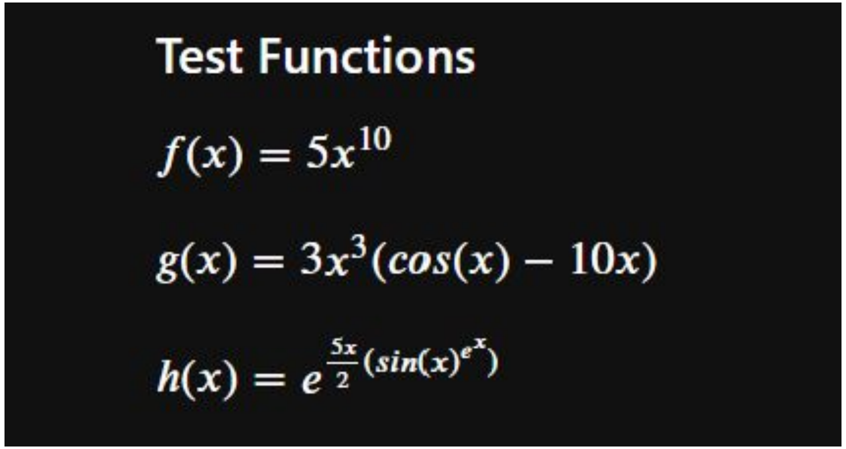
Initial Tests

We decided to start by running some basic tests

- Feel out how differentiation actually worked in the packages
- Look for differences
- Figure out which values from @timed would be useful
- We used three different functions
- Each function was to be differentiated at x = 2

Methods

- Zygote - gradient()
- Enzyme - autodiff()



Test Functions

$$f(x) = 5x^{10}$$

$$g(x) = 3x^3(cos(x) - 10x)$$

$$h(x) = e^{\frac{5x}{2}(sin(x)^{e^x})}$$

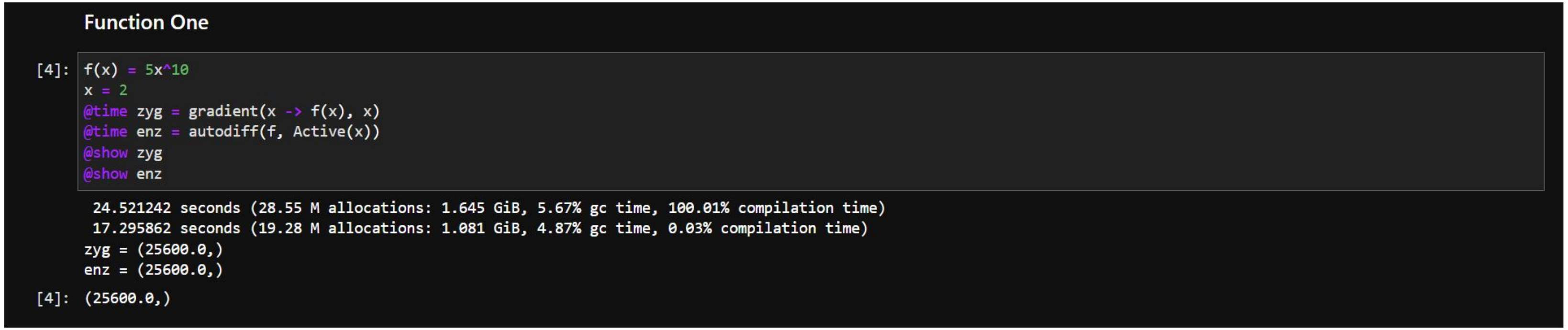
Initial Findings

First results

- After running the initial tests it was hard to determine which was better
- Sometimes Zygote was more optimal, sometimes Enzyme was
- The more times we ran the same tests, the faster the differentiation became

Outputs

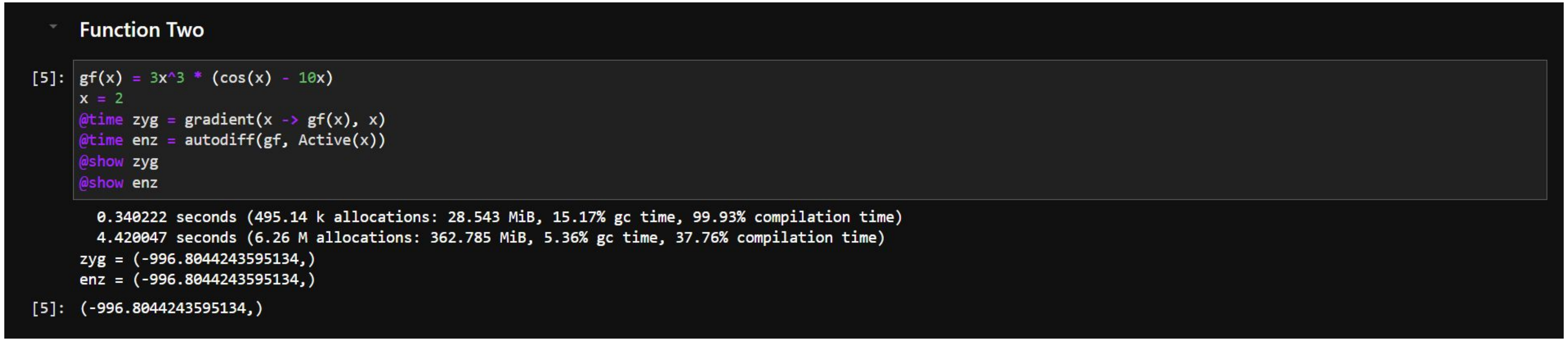
- Function One
  - Zygote - 24.52s
  - Enzyme - 17.29s
- Function Two
  - Zygote - 0.34s
  - Enzyme - 4.42s
- Function Three
  - Zygote - 0.32s
  - Enzyme - 2.50s



Function One

```
[4]: f(x) = 5x^10
     x = 2
     @time zyg = gradient(x -> f(x), x)
     @time enz = autodiff(f, Active(x))
     @show zyg
     @show enz
```

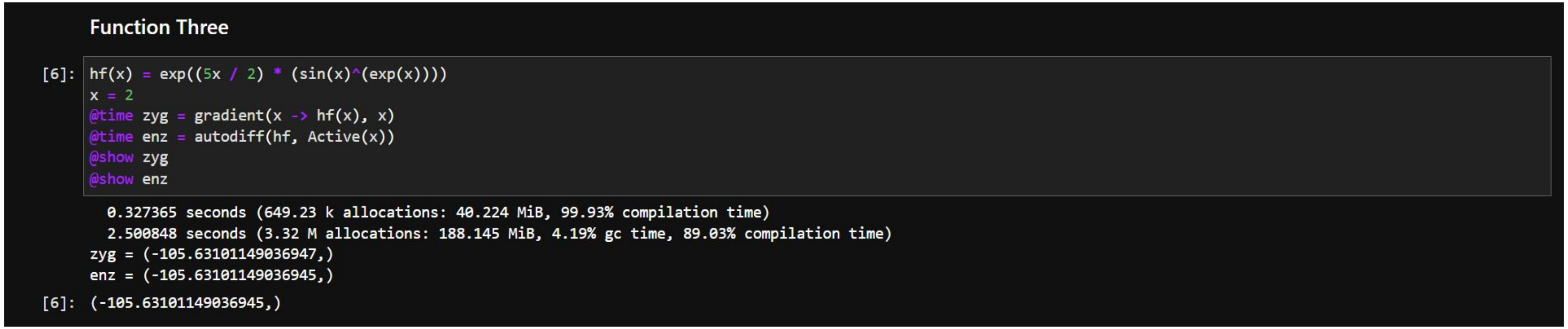
```
 24.521242 seconds (28.55 M allocations: 1.645 GiB, 5.67% gc time, 100.01% compilation time)
 17.295862 seconds (19.28 M allocations: 1.081 GiB, 4.87% gc time, 0.03% compilation time)
zyg = (25600.0,)
enz = (25600.0,)
[4]: (25600.0,)
```



Function Two

```
[5]: gf(x) = 3x^3 * (cos(x) - 10x)
     x = 2
     @time zyg = gradient(x -> gf(x), x)
     @time enz = autodiff(gf, Active(x))
     @show zyg
     @show enz
```

```
  0.340222 seconds (495.14 k allocations: 28.543 MiB, 15.17% gc time, 99.93% compilation time)
  4.420047 seconds (6.26 M allocations: 362.785 MiB, 5.36% gc time, 37.76% compilation time)
zyg = (-996.8044243595134,)
enz = (-996.8044243595134,)
[5]: (-996.8044243595134,)
```



Function Three

```
[6]: hf(x) = exp((5x / 2) * (sin(x)^(exp(x))))
     x = 2
     @time zyg = gradient(x -> hf(x), x)
     @time enz = autodiff(hf, Active(x))
     @show zyg
     @show enz
```

```
  0.327365 seconds (649.23 k allocations: 40.224 MiB, 99.93% compilation time)
  2.500848 seconds (3.32 M allocations: 188.145 MiB, 4.19% gc time, 89.03% compilation time)
zyg = (-105.63101149036947,)
enz = (-105.63101149036945,)
[6]: (-105.63101149036945,)
```

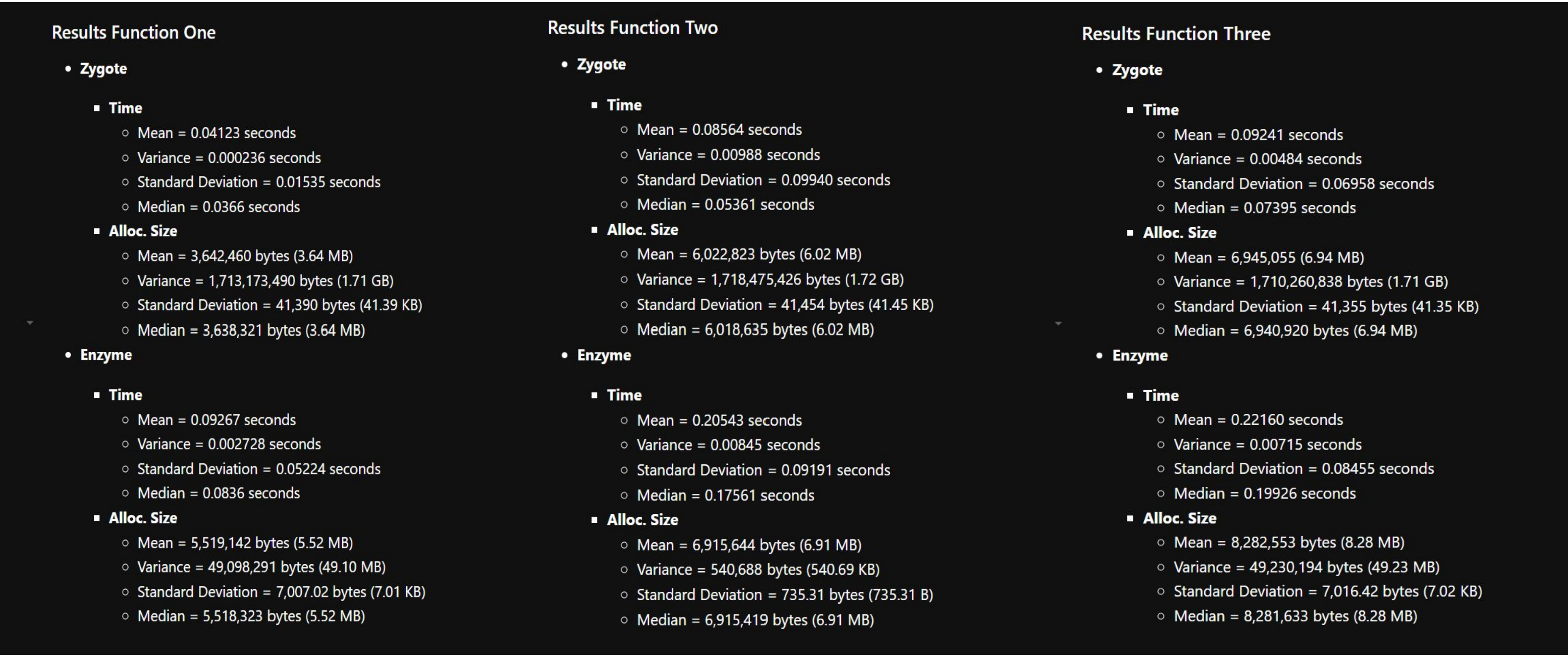
Data Science

Zygote wins in cases two and three but Enzyme wins in one

- Things seems generally inconclusive, so each function was run 100 times and then analyzed
    - Mean
    - Median
    - Variance
    - Standard Deviation

Results

- These tests had more conclusive results



Results Function One

- **Zygote**
  - **Time**
    - Mean = 0.04123 seconds
    - Variance = 0.000236 seconds
    - Standard Deviation = 0.01535 seconds
    - Median = 0.0366 seconds
  - **Alloc. Size**
    - Mean = 3,642,460 bytes (3.64 MB)
    - Variance = 1,713,173,490 bytes (1.71 GB)
    - Standard Deviation = 41,390 bytes (41.39 KB)
    - Median = 3,638,321 bytes (3.64 MB)
- **Enzyme**
  - **Time**
    - Mean = 0.09267 seconds
    - Variance = 0.002728 seconds
    - Standard Deviation = 0.05224 seconds
    - Median = 0.0836 seconds
  - **Alloc. Size**
    - Mean = 5,519,142 bytes (5.52 MB)
    - Variance = 49,098,291 bytes (49.10 MB)
    - Standard Deviation = 7,007.02 bytes (7.01 KB)
    - Median = 5,518,323 bytes (5.52 MB)

Results Function Two

- **Zygote**
  - **Time**
    - Mean = 0.08564 seconds
    - Variance = 0.00988 seconds
    - Standard Deviation = 0.09940 seconds
    - Median = 0.05361 seconds
  - **Alloc. Size**
    - Mean = 6,022,823 bytes (6.02 MB)
    - Variance = 1,718,475,426 bytes (1.72 GB)
    - Standard Deviation = 41,454 bytes (41.45 KB)
    - Median = 6,018,635 bytes (6.02 MB)
- **Enzyme**
  - **Time**
    - Mean = 0.20543 seconds
    - Variance = 0.00845 seconds
    - Standard Deviation = 0.09191 seconds
    - Median = 0.17561 seconds
  - **Alloc. Size**
    - Mean = 6,915,644 bytes (6.91 MB)
    - Variance = 540,688 bytes (540.69 KB)
    - Standard Deviation = 735.31 bytes (735.31 B)
    - Median = 6,915,419 bytes (6.91 MB)

Results Function Three

- **Zygote**
  - **Time**
    - Mean = 0.09241 seconds
    - Variance = 0.00484 seconds
    - Standard Deviation = 0.06958 seconds
    - Median = 0.07395 seconds
  - **Alloc. Size**
    - Mean = 6,945,055 (6.94 MB)
    - Variance = 1,710,260,838 bytes (1.71 GB)
    - Standard Deviation = 41,355 bytes (41.35 KB)
    - Median = 6,940,920 bytes (6.94 MB)
- **Enzyme**
  - **Time**
    - Mean = 0.22160 seconds
    - Variance = 0.00715 seconds
    - Standard Deviation = 0.08455 seconds
    - Median = 0.19926 seconds
  - **Alloc. Size**
    - Mean = 8,282,553 bytes (8.28 MB)
    - Variance = 49,230,194 bytes (49.23 MB)
    - Standard Deviation = 7,016.42 bytes (7.02 KB)
    - Median = 8,281,633 bytes (8.28 MB)

Basic Test Conclusions

Function One

- Zygote Average - 0.041s and 3.64 MB
- Enzyme Average - 0.092s and 5.52 MB

Function Two

- Zygote Average - 0.085s and 6.02 MB
- Enzyme Average - 0.205s and 6.91 MB

Function Three

- Zygote Average - 0.092s and 6.94 MB
- Enzyme Average - 0.221s and 8.28 MB

Enzyme always had a very small standard deviation and variance for space

- Usually in B and sometimes in 1s of KB

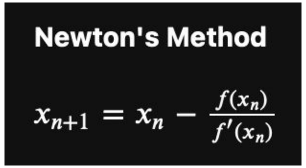
Zygote had much greater standard deviation and variance for space

- Always in 10s of KB

Conclusions?

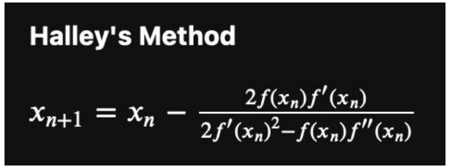
Root-finding Methods

- Newton’s
- Halley’s
- Golbabai and Javidi’s
- Noor’s
- Zhanlav’s



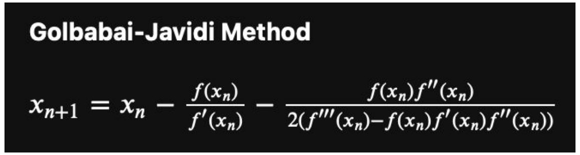
Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



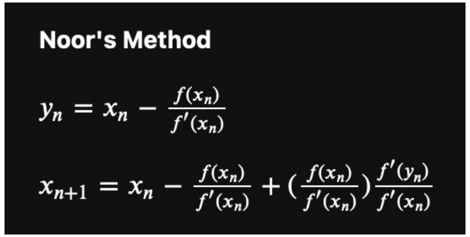
Halley's Method

$$x_{n+1} = x_n - \frac{2f(x_n)f'(x_n)}{2f'(x_n)^2 - f(x_n)f''(x_n)}$$



Golbabai-Javidi Method

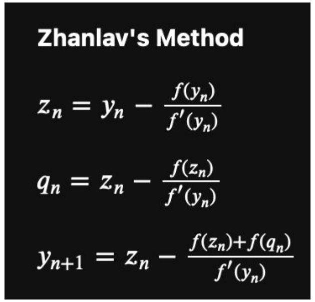
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f(x_n)f''(x_n)}{2(f'''(x_n) - f(x_n)f'(x_n)f''(x_n))}$$



Noor's Method

$$y_n = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} + \left(\frac{f(x_n)}{f'(x_n)}\right)\frac{f'(y_n)}{f'(x_n)}$$



Zhanlav's Method

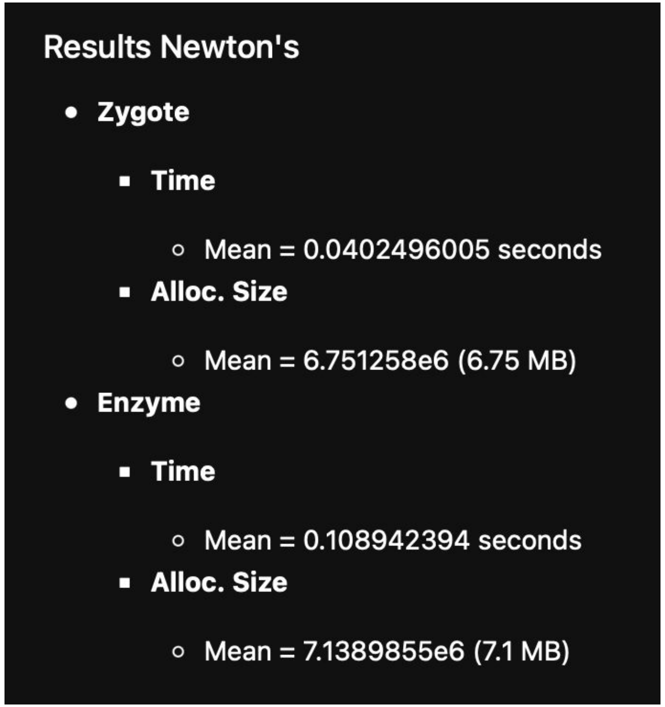
$$z_n = y_n - \frac{f(y_n)}{f'(y_n)}$$

$$q_n = z_n - \frac{f(z_n)}{f'(y_n)}$$

$$y_{n+1} = z_n - \frac{f(z_n) + f(q_n)}{f'(y_n)}$$

Results

- Zygote ~ 2x faster
- Memory allocation ~ 7MB
- Enzyme using slightly more
- Halley’s showed biggest time diff.
    - Likely due to second derivative



Results Newton's

- **Zygote**
    - **Time**
        - Mean = 0.0402496005 seconds
    - **Alloc. Size**
        - Mean = 6.751258e6 (6.75 MB)
- **Enzyme**
    - **Time**
        - Mean = 0.108942394 seconds
    - **Alloc. Size**
        - Mean = 7.1389855e6 (7.1 MB)

Questions?