# CSE 380: Final Project

## Canned Project

Evan Ott

December 9, 2016

# Overview


1. Third-Party Libraries
2. Problem 1
   - Results
3. Problem 2
   - Simulation Results

# Third-Party Libraries

| | |
|---|---|
| Catch | unit testing, header-defined |
| Eigen | working with vectors and matrices easily, header-defined |
| HDF5 | output format to store data |
| GSL | scientific library |
| INIReader | way to define configuration options easily, header-defined |

# Problem 1

For this simple problem, I solved the simple ODE

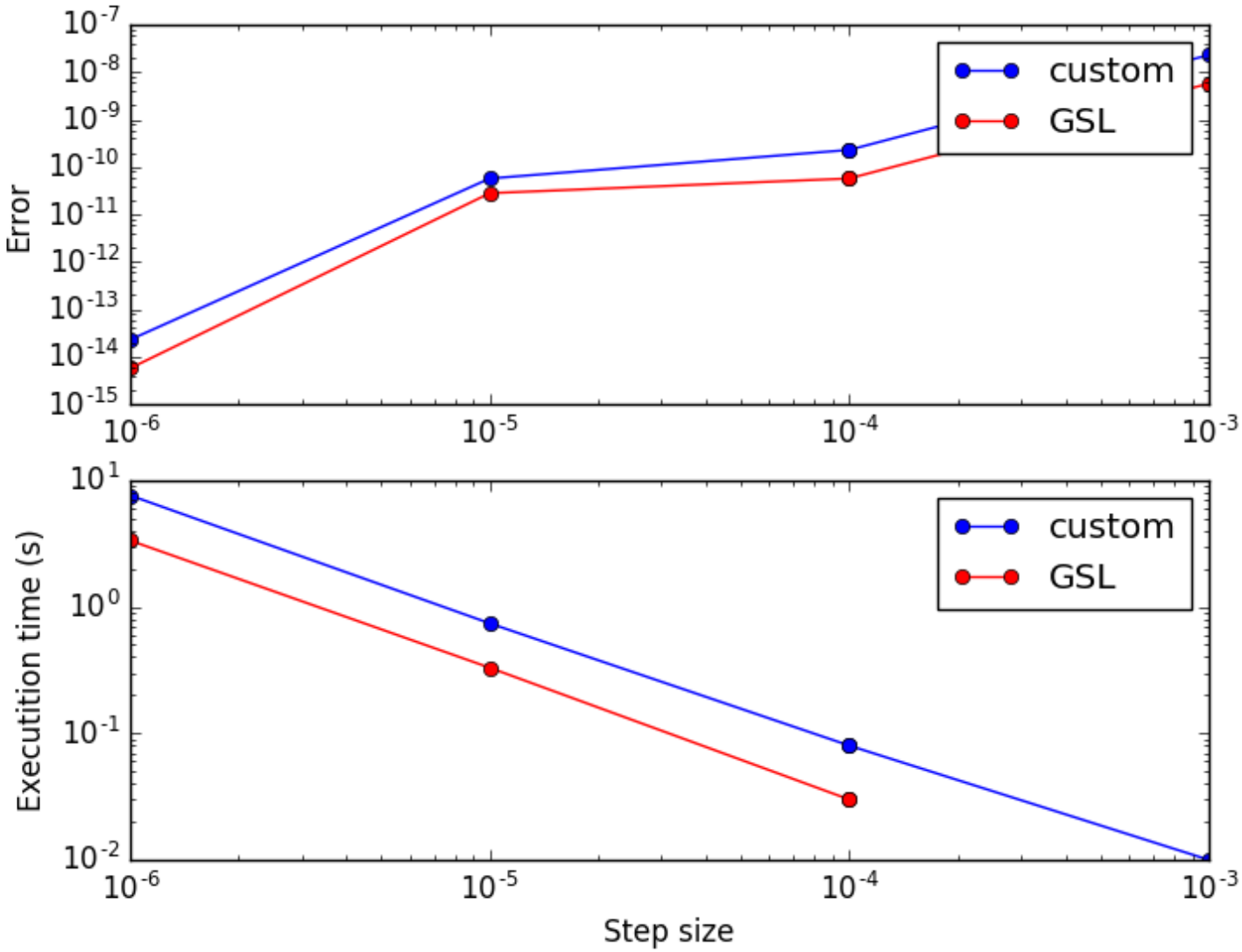
$$\dot{y}(t) = -0.1 \cdot y(t)$$

which of course has the solution

$$y(t) = y(t_0)e^{-0.1(t-t_0)}$$

for an initial condition specified at $t_0$.

# Results



# Results

- My implementation of forward Euler is slower than GSL's RK4 method on the same step size
- My implementation is also less accurate than GSL's RK4 on the same step size

This shouldn't be surprising, as forward Euler is in general less accurate than more general Runge-Kutta methods. Furthermore, I decided to use `Eigen` for handling vectors for problem 2 and the same overhead is present here.

# Problem 2

I used *Mathematica* to solve the system of ODEs for the following analytical solutions

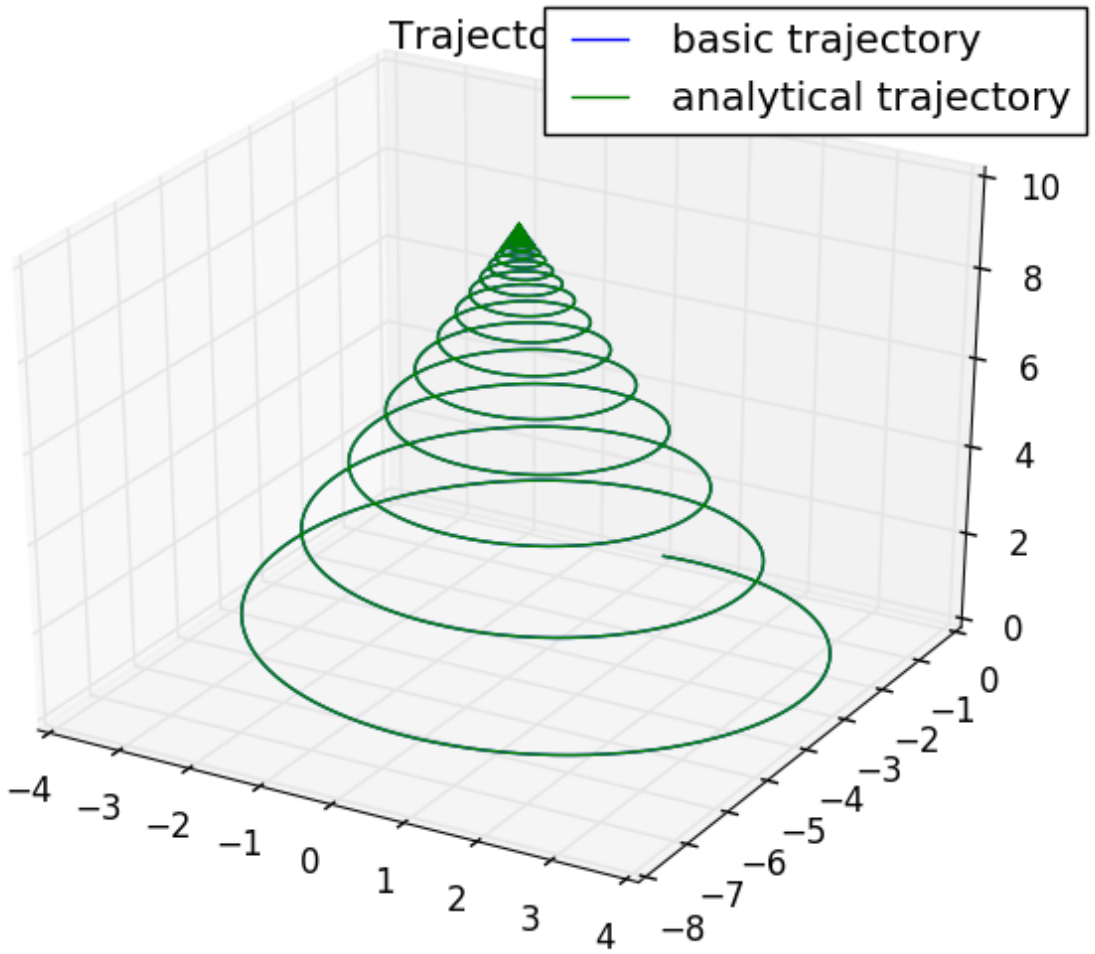
$$x(t) = \frac{20\tau e^{-\frac{t}{\tau}}\left(\tau\omega\sin(t\omega) + e^{t/\tau} - \cos(t\omega)\right)}{\tau^2\omega^2 + 1}$$

$$y(t) = -\frac{20\tau e^{-\frac{t}{\tau}}\left(\tau\omega e^{t/\tau} - \tau\omega\cos(t\omega) - \sin(t\omega)\right)}{\tau^2\omega^2 + 1}$$

$$z(t) = 2\tau e^{-\frac{t}{\tau}}\left(e^{t/\tau} - 1\right)$$

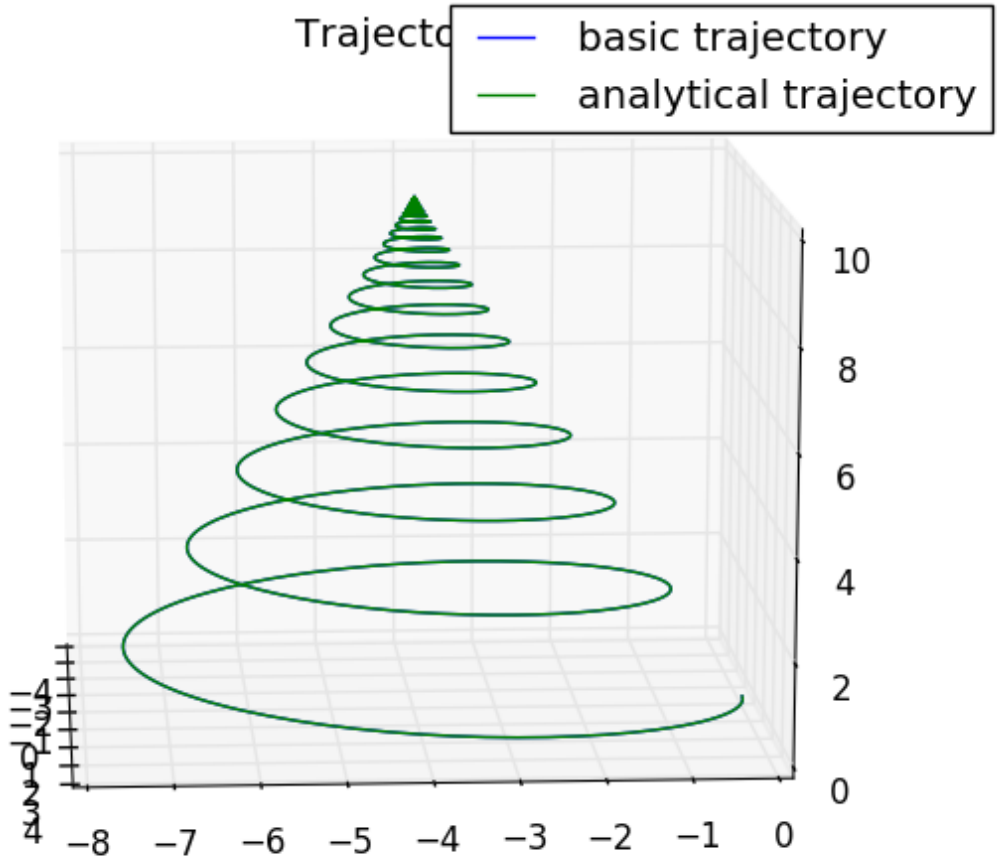
# Simulation Results

This is a high-resolution result computed on Stampede for 1M iterations with a step size of 0.0001 (RK4).



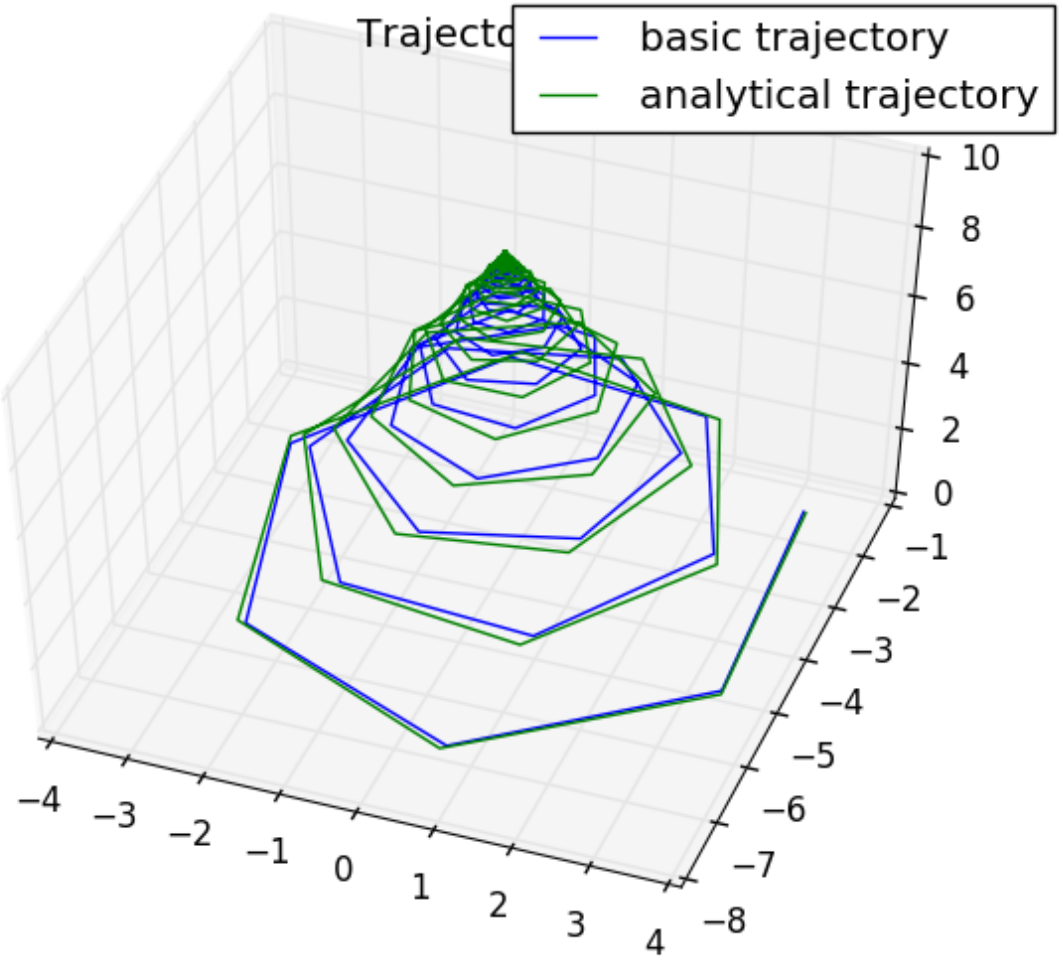
# Simulation Results

This is a lower-resolution result computed on Stampede for 10K iterations with a step size of 0.01 (RK4).



# Simulation Results

This is a minimal-resolution result computed on Stampede for 500 iterations with a step size of 0.2 (RK4).



# References

# Appendix A