# Presentation for Final Project
# Tools and Techniques of Computational Science

Farzan Memarian

The University of Texas at Austin

December 11, 2016

# Introduction

## Problem Statement

- This presentation is about the the Canned Project
- There are a total of two problems
- The programming language chosen for this project is c++
- Python has been used for post processing and plotting the figures
-

## Tools used for the project

- Several tools have been applied, including debug, verification, input parsing using Grvy, timing using grvy

# Problem1

## Differential Equation to Solve

- In this problem, we could choose a simple ODE of our choice
- $y'(t) = y(t), y(0) = 1$

## Methods used

- Forwad Euler implemented by the developer (me)
- Analytical Solution
- Explitic Runge-Kutta (4th order)

# Problem 2

## Differential Equation to Solve

- In this problem, we solve the equations of motion for a charged particle in an Electric field
- $\ddot{x} = \omega\dot{y} - \dot{x}/\tau$
- $\ddot{y} = -\omega\dot{x} - \dot{y}/\tau$
- $\ddot{z} = -\dot{z}/\tau$

  Initial Conditions:
  - $x(0) = y(0) = z(0) = 0$
  - $\dot{x}(0) = 20, \dot{y}(0) = 0, \dot{z}(0) = 2$

## Methods used from GSL library

- Explicit embedded Runge-Kutta (2, 3) method, rk2
- Explicit 4th order (classical) Runge-Kutta, rk4
- Explicit embedded Runge-Kutta-Fehlberg (4, 5) method, rkf45

# Build System

## Modules Required

- to run the code on stampede, the follwoing modules need to be loaded:
  - load gcc/4.7.1
  - load gsl
  - load grvy

## Makefile

- to create the executable we run ”make”
  - creates object filed by compiling the source codes
  - links the object files to create the executable which is ”toolsProject”
- To perform a check on the results, we run ”make check”:
  - it runs the code based on a certain input files
  - compares the output file to an output file that we know is correct

# Version Control

## git and github

- git has been used as the version control system
- commits were made whenever some part of the code was changed
- the link to the project git hub: https://github.com/frUTCS16/toolsProject

# Verification Plan

the verification mode can be turned on or off from the input file

## Problem 1

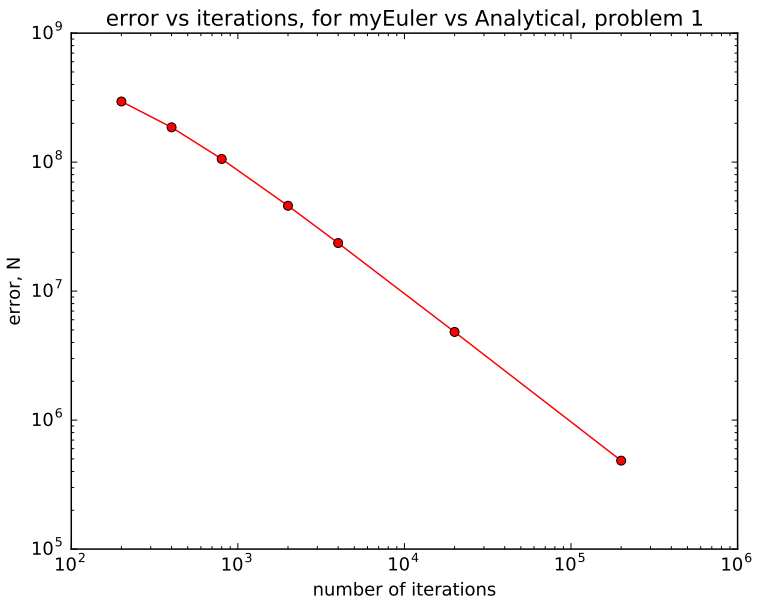
- for this problem we have the analytical solution
- the error can easily be calculated by comparing the results to the analytical solution
- we either take the norm of the error or compare the error at the last point

## Problem 2

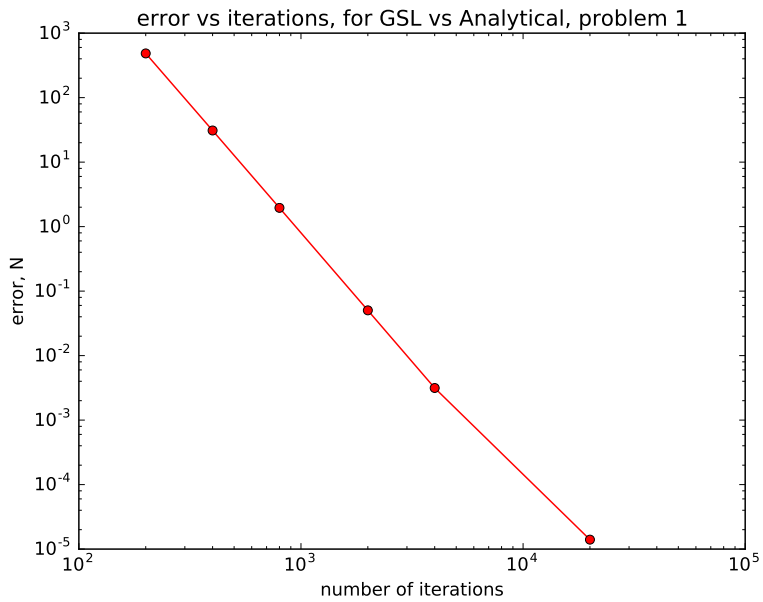
- as the exact solution, we run the code for a very small step size
- compare the results for other stepsizes to the result from run with very small step size
- we either take the norm of the error or compare the error at the last point

# Convergence, Problem 1

## Convergence of forward Euler



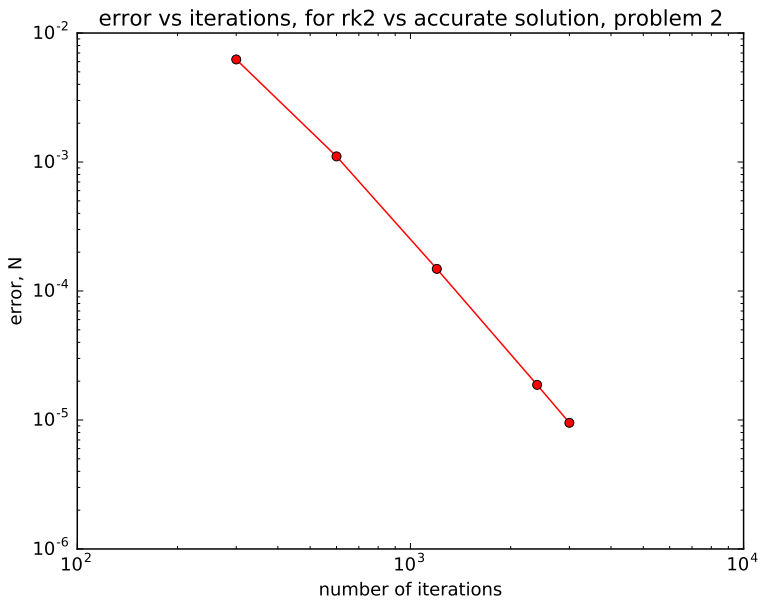

## Convergence of rk4 from GSL



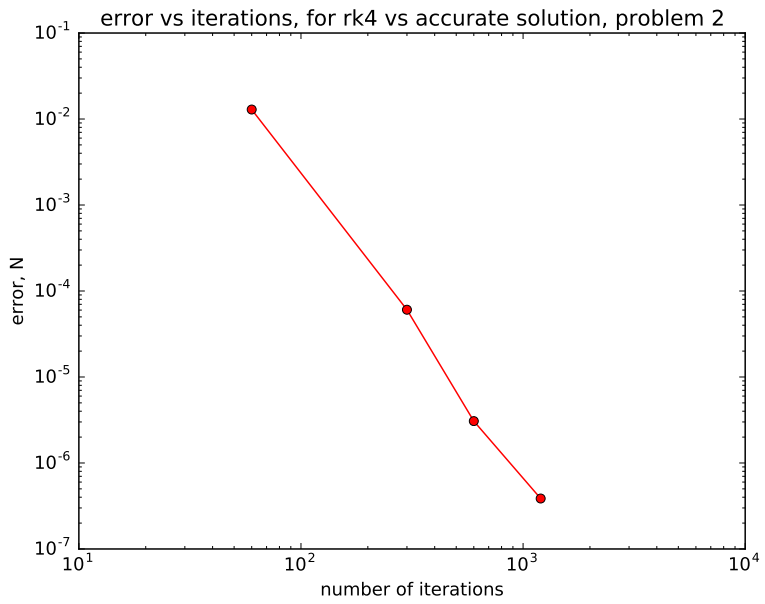

- It can be observed that the convergence rate (i.e negative of the slope) of the Forward euler method is 1
- The convergence rate for RK4 is 4

# Convergence, Problem 2

## Convergence of rk2

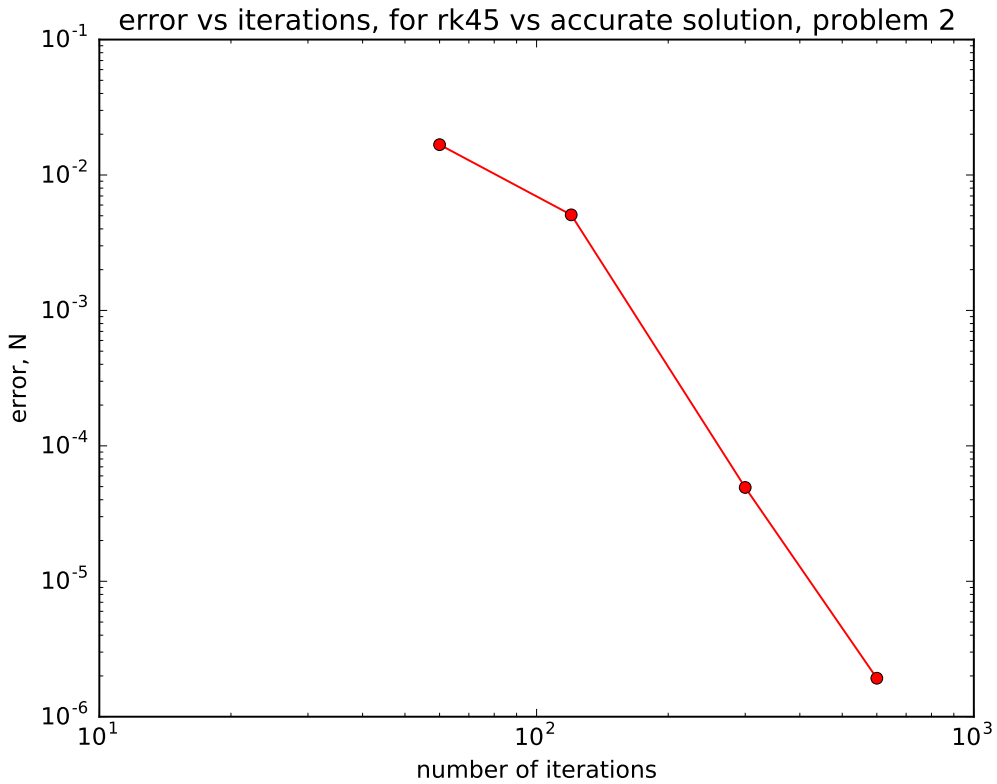



## Convergence of rk4



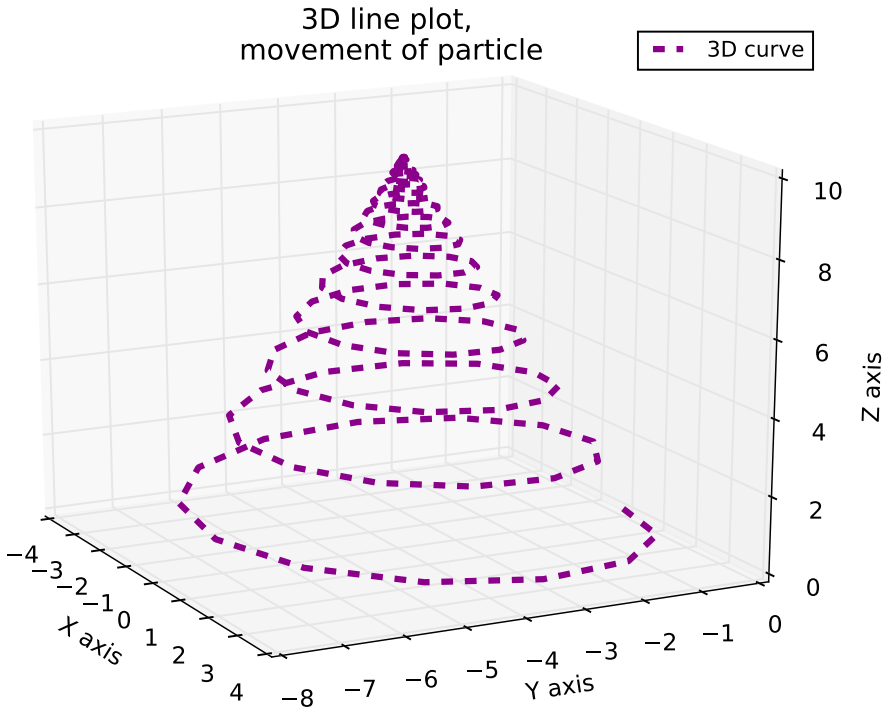

- The convergence rate for RK2 is 2
- The convergence rate for RK4 is 4

# Convergence, Problem 2




- the convergence rate for RKF45 is almost 4 (negative of the slope)

# Problem 2, 3D tragectory




- This is the trajectory of the object, all the methods give the same trajectory

# Input parsing

## Input Parsing using GRVY

- at run time we type we must type the name of the input file in front of the executable
- the following variables will be read by grvy
  - problem – Problem number
  - odeMethod – method used for problem 2, either rk4, or rk2, or rkf45
  - verification – 1 turns on the verification mode and zero turns it off
  - debug – determines the level of debug, zero is no debug, 1 only prints the timing performace, 2 prints more
  - maxTime – Max Time
  - h – step size
  - numberPrint – the number of stepsizes to print the results

# Timing Problem 1

```
-------------------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                      |      Mean      Variance       Count
--> myEuler             : 1.15761e+00 secs ( 99.4078 %) | [1.15761e+00  0.00000e+00         1]
--> GRVY_Unassigned     : 6.89673e-03 secs (  0.5922 %)

   Total Measured Time = 1.16450e+00 secs (100.0000 %)
-------------------------------------------------------------------------------------------



-------------------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                      |      Mean      Variance       Count
--> gslSolverProblem1   : 5.67484e-01 secs ( 99.9991 %) | [5.67484e-01  0.00000e+00         1]
--> myEuler             : 0.00000e+00 secs (  0.0000 %) | [    N/A     0.00000e+00         0]
--> GRVY_Unassigned     : 5.00679e-06 secs (  0.0009 %)

   Total Measured Time = 5.67489e-01 secs (100.0000 %)
-------------------------------------------------------------------------------------------
```

- in this figure, the forward euler function and the gsl rk4 have been compared
- both cases were run when Tmax = 20, and time step is $10e-4$
- it can see seen that gslSolver takes less time

# Timing Problem 2

```
-------------------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.01803e+00 secs ( 99.5818 %) | [1.01803e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 4.27508e-03 secs (  0.4182 %)

    Total Measured Time = 1.02230e+00 secs (100.0000 %)
-------------------------------------------------------------------------------------------

login1.stampede(153)$ ./toolsProject input.dat

-------------------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.62174e+00 secs ( 99.5661 %) | [1.62174e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 7.06673e-03 secs (  0.4339 %)

    Total Measured Time = 1.62880e+00 secs (100.0000 %)
-------------------------------------------------------------------------------------------

login1.stampede(154)$ ./toolsProject input.dat

-------------------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.54674e+00 secs ( 99.5255 %) | [1.54674e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 7.37381e-03 secs (  0.4745 %)

    Total Measured Time = 1.55411e+00 secs (100.0000 %)
-------------------------------------------------------------------------------------------
```

- these times correspond to rk2, rk4 and rkf45 respectively
- it can be seen that the run with rk2 is shorer.

# Timing Problem 2

```
-------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.01803e+00 secs ( 99.5818 %) | [1.01803e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 4.27508e-03 secs (  0.4182 %)

    Total Measured Time = 1.02230e+00 secs (100.0000 %)
-------------------------------------------------------------------------------

login1.stampede(153)$ ./toolsProject input.dat

-------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.62174e+00 secs ( 99.5661 %) | [1.62174e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 7.06673e-03 secs (  0.4339 %)

    Total Measured Time = 1.62880e+00 secs (100.0000 %)
-------------------------------------------------------------------------------

login1.stampede(154)$ ./toolsProject input.dat

-------------------------------------------------------------------------------
GRVY Timing - Performance Timings:                    |      Mean      Variance       Count
--> rk4 or rk2 or rkf45 : 1.54674e+00 secs ( 99.5255 %) | [1.54674e+00  0.00000e+00          1]
--> GRVY_Unassigned     : 7.37381e-03 secs (  0.4745 %)

    Total Measured Time = 1.55411e+00 secs (100.0000 %)
-------------------------------------------------------------------------------
```

- these times correspond to rk2, rk4 and rkf45 respectively
- it can be seen that the run with rk2 is shorer.

# How did I start up

## Problem Statement

- I started by wrinting a pseudo code
- Then I started to write the main and from there all the corresponding functions
- I was testing all the function as I was writting them
- I set up a git and commited when there was a change
- I originally started the developmene on my laptop but then switech to stampede because of the libraries

# What would I do differently

## Problem Statement

- I would spend more time thinking about the structure of the code rather than start coding right away
- I would make sure I understand the problem well
- I spend too much time on code verification and it was mainly because I was my code was originally written in a way that was not very suitable for verficiation so I had to change it significantly

# Lessons learned

## Problem Statement

- it matters to write a clean code and keep good documentation
- its better to start coding on stampede rather than your personal laptop if it is going to be run on stampede at the end
- git is a great tool to use!
- writting modular code is really important
- I learned many new skill, such as input parsing, version control, unit testing and regression testing, timing and etc.