Computer_Homework1

Generated by Doxygen 1.9.1

Chapter 1

Computer Homework 1

Solve Kepler problem via finite difference Method

1.1 Requirements

To install this program, you should have

- C++ compiler like g++
- gnu make or cmake

1.2 Installation

- gnu make
  - Type make, then we can see hw1 executable file in bin directory
- cmake
  1. make build directory
  2. go to build directory and type cmake .. -DPRECISION_LEVEL precision_level
     - precision_level 0: float
     - precision_level 1: double
  3. Type make then we can see hw1 executable in build directory

1.3 How To Use

Execute hw1 then, it will interactively read

- inital condition
- number of gird points to evaluate
- output file name

Then it computes and saves solution to file. You can plot the result using usual plotting software like gnuplot

1.4 Copyright

1.5 License

Chapter 2

Finite difference Method

To solve Kepler problem, we need to solve

$$\frac{\mathrm{d}^2\zeta}{\mathrm{d}t^2} = \frac{1}{\zeta^3} - \frac{1}{\zeta^2} \tag{2.1}$$

with initial condition

$$\zeta(t_0) = \zeta_0$$
$$\zeta'(t_0) = \zeta'_0$$

To solve above 2nd order ordinary differential equation (2.1) numerically, we need to approximate 2nd derivative as finite difference. Suppose that the solution $\zeta(t)$ has
continuous 4th order derivative in the Domain $[t_0, t_f]$. then

$$\zeta(x+h) = \zeta(x) + \zeta'(x)h + \frac{1}{2!}\zeta''(x)h^2 + \frac{1}{3!}\zeta'''(x)h^3 + \frac{1}{4!}\zeta^{(4)}(\eta)h^4 \tag{2.2}$$

for some $\eta(x,h) \in (t_0, t_f)$. Using 4th order taylor approximation (2.2) , we can get following equation

$$\zeta(x-h) - 2\zeta(x) + \zeta(x+h) = h^2\zeta''(x) + O(h^4) \tag{2.3}$$

Next uniformly divide the domain $[t_0, t_f]$ into n sub intervals. let x_i be the end points of the sub intervals then for $0 \le i \le n$,

$$x_i = t_0 + ih \tag{2.4}$$

, where $h = (t_f - t_0)/n$. Now for $0 \le i \le n$, define ζ_i as following

$$\zeta_i = \zeta(x_i) \tag{2.5}$$

Then we can rewrite finite difference equation (2.3) as following

$$\zeta_{i-1} - 2\zeta_i + \zeta_{i+1} = h^2\zeta''_i + O(h^4) \tag{2.6}$$

for $1 \le i \le n-1$. Plug this equation (2.6) into 2nd order ode (2.1) , the we have following recurrence relation

$$\zeta_{i-1} - 2\zeta_i + \zeta_{i+1} = h^2\left(\frac{1}{\zeta_i^3} - \frac{1}{\zeta_i^2}\right) \tag{2.7}$$

In above equation (2.7) , we turncate, so local turncation error is $O(h^4) = O(n^{-4})$. Therefore global turncation error can be roughly estimated to $O(n^{-3})$. To solve recurrance relation, we need to know both ζ_0 and ζ_1. However only ζ_0 is explictly given by the initial condition. To approximate ζ_1 with $O(n^{-3})$ error bound, I use 2nd order talyor expension.

$$\zeta_1 \approx \zeta_0 + \zeta'_0 h + \frac{1}{2!}\zeta''_0 h^2 \tag{2.8}$$

ζ''_0 can be derived by 2nd order ode (2.1)

$$\zeta''_0 = \frac{1}{\zeta_0^3} - \frac{1}{\zeta_0^2} \tag{2.9}$$

2.1 Complexity

Clearly

$$O(n).$$

2.2 Accuracy

Global turncation error is roughly estimated by

$$O(n^{-3}).$$

2.3 Convergence

- Initial Condition

$$\zeta(0) = 0.9$$
$$\zeta'(0) = 0$$

- Initial time: 0
- Final time: 10

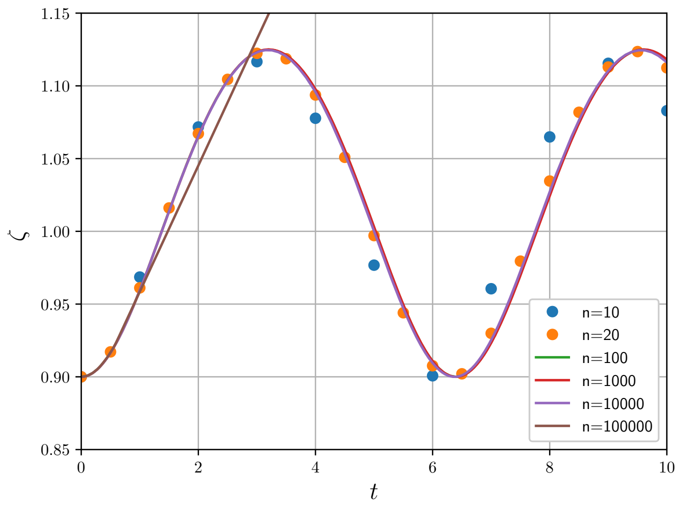


Figure 2.1 Convergence plot: single precision

Due to tuncation error it diverges at $n = 10^5$.

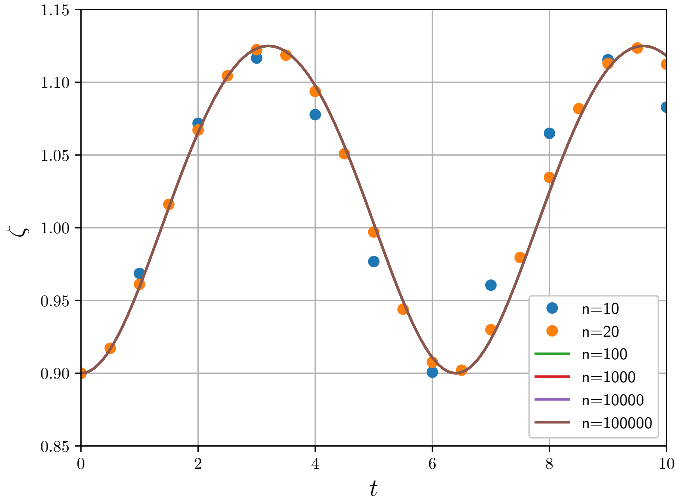


Figure 2.2 Convergence plot: double precision

However in double precision, it converges to exact solution.

Chapter 3

Theta

By conservation of angular momentum, Angle θ satisfies following relation

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{1}{\zeta^2} \tag{3.1}$$

Integrate both side then we can deduce

$$\theta(t) = \theta_0 + \int_{t_0}^{t} \frac{1}{\zeta^2}\mathrm{d}t \tag{3.2}$$

Let $\theta_i = \theta(t_i)$ as in Finite difference Method, then for $1 \leq i$,

$$\theta_i = \theta_{i-1} + \int_{t_{i-1}}^{t_i} \frac{1}{\zeta^2}\mathrm{d}t \tag{3.3}$$

Next approximate the integral using trapezoidal rule then

$$\theta_i \approx \theta_{i-1} + \frac{t_i - t_{i-1}}{2}\left(\frac{1}{\zeta_{i-1}^2} + \frac{1}{\zeta_i^2}\right) \tag{3.4}$$

θ_i has $O(n^{-3})$ local turncation error for trapezoidal rule and additional $O(n^{-3})$ for the global turncation error of ζ (see Finite difference Method Accuracy). So the global turncation error of θ can be estimated to $O(n^{-2})$

3.1 Complexity

Clearly

$$O(n) \tag{3.5}$$

3.2 Accuracy

The global turncation error of θ is roughly estimated to

$$O(n^{-2}) \tag{3.6}$$

3.3 Convergence

- Initial Condition

$$\zeta(0) = 0.9$$
$$\zeta'(0) = 0$$
$$\theta(0) = 0$$

- Initial time: 0
- Final time: 10

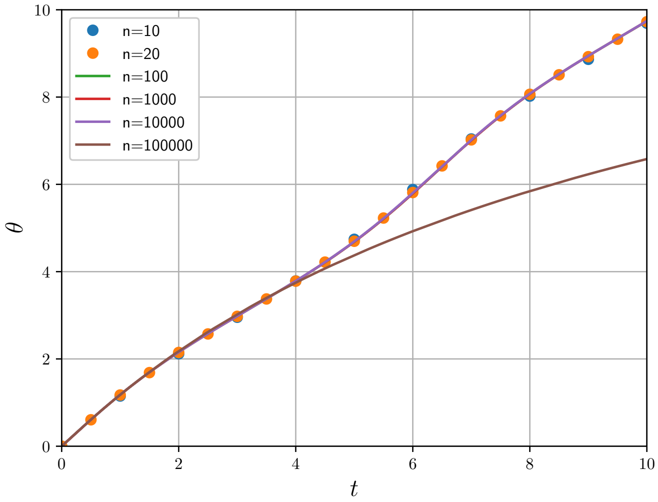


Figure 3.1 Convergence plot: single precision

Due to tuncation error it diverges at $n = 10^5$.

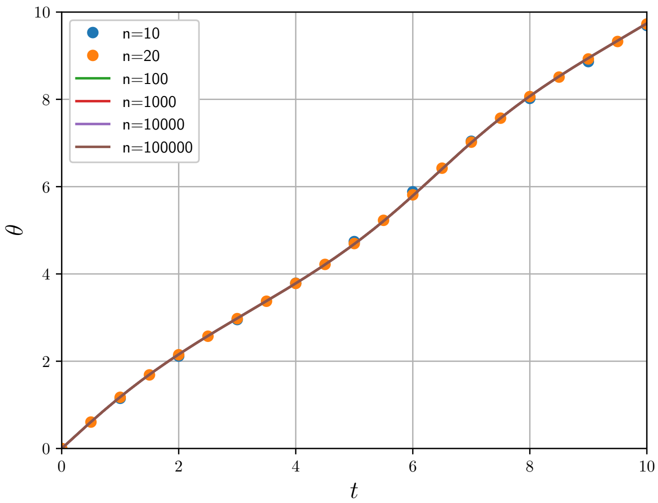


Figure 3.2 Convergence plot: double precision

However in double precision, it converges to exact solution.

Chapter 4

Trajectory

We know that

$$x(t) = \zeta(t)\cos\theta(t) \qquad (4.1)$$
$$y(t) = \zeta(t)\sin\theta(t)$$

Using above relation (4.1) , we can draw trajectory plot.

4.1 Trajectory Plot

- Initial Condition

$$\zeta(0) = 0.9$$
$$\zeta'(0) = 0$$
$$\theta(0) = 0$$

- Initial time: 0
- Final time: 10

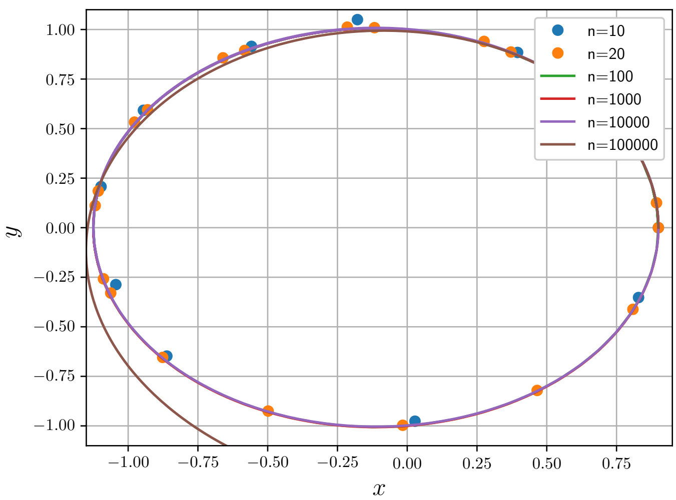


Figure 4.1 Convergence plot: single precision

Due to floating point tuncation error, it diverges at $n = 10^5$ (i.e. $\Delta t = 10^{-4}$).

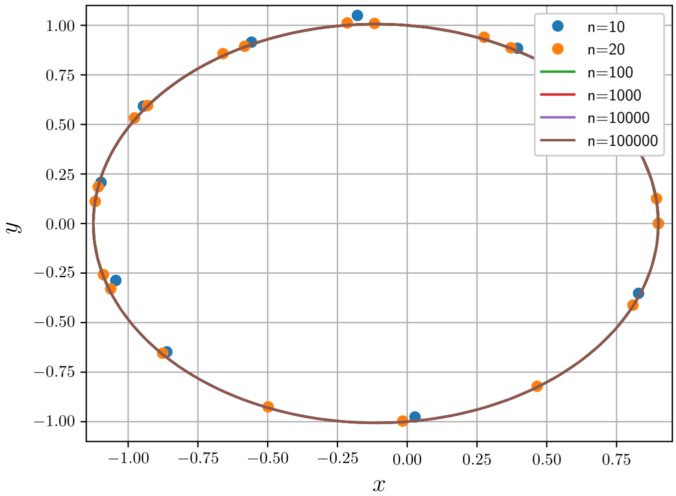


Figure 4.2 Convergence plot: double precision

However, in double precision, it converges to exact epllise trajectory.

4.2 Trajectory Error analysis

Since trajectory approximated by finite difference method is converges to exact epllise trajectory, when double precision, I assume $n = 10^5$ with double precision result as exact path. Then we can estimate error at the selected points $(t = 1, 2, ..., 10)$.

$$\text{error} = \sqrt{(x(t) - x_{ref}(t))^2 + (y(t) - y_{ref}(t))^2} \quad (4.2)$$

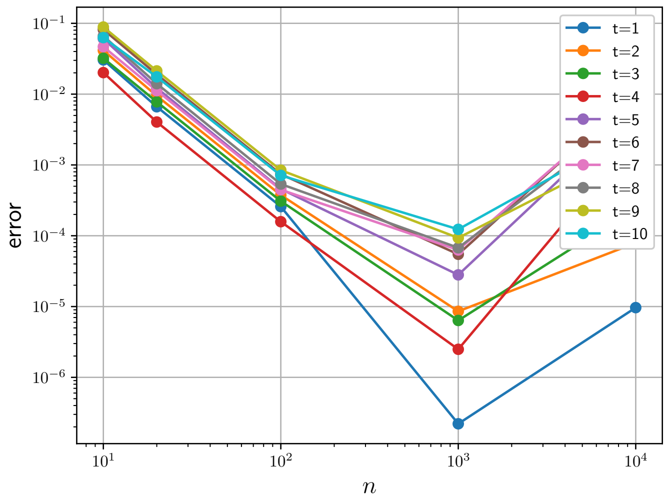


Figure 4.3 Error Analysis: single precision

As you can see due to tuncation error, accuracy is worsen when $n > 10^3$.

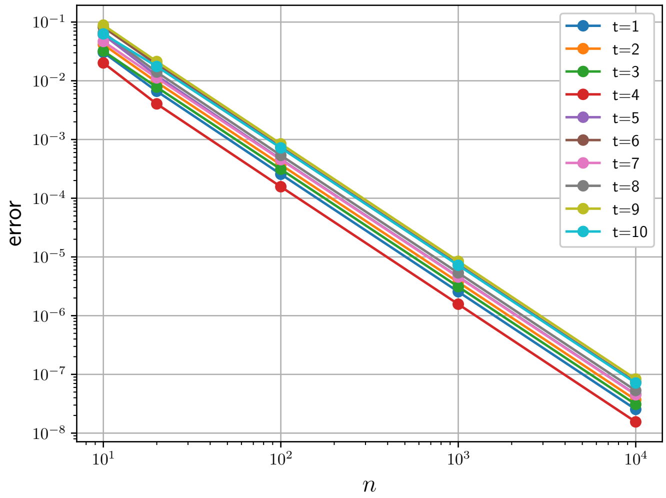


Figure 4.4 Error Analysis: double precision

However, when double precision, accuracy is better and better as n increases. Therefore we can estimate the order of error using above plot. Order of error is estimated to

$$O(n^{-2}) \tag{4.3}$$

Practical error bound is same as estimated error bound $O(n^{-2})$.

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

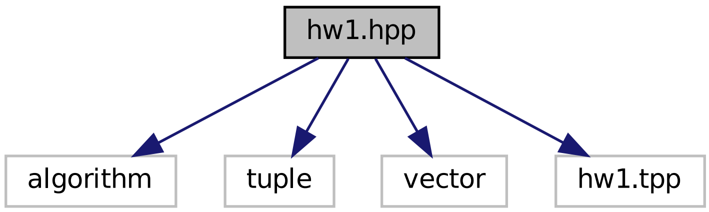
File Documentation

6.1 hw1.hpp File Reference

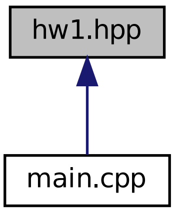
Header file for homework1 of Computer1 class in Yonsei University Use finite difference method to solve Kepler problem.

```
#include <algorithm>
#include <tuple>
#include <vector>
#include "hw1.tpp"
```

Include dependency graph for hw1.hpp:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename T >
  std::tuple< std::vector< T >, std::vector< T >, std::vector< T > > HW1 (T t0, T t1, int n, T y0, T y0p, T theta0)

  HW1: Solve Kepler problem via finite difference Method Behavior of HW1 is undefined when type T is not equal to one of float, double, long double.

6.1.1 Detailed Description

Header file for homework1 of Computer1 class in Yonsei University Use finite difference method to solve Kepler problem.

Author

pistack (Junho Lee)

Date

2021. 11. 3.

6.1.2 Function Documentation

6.1.2.1 HW1()

```
template<typename T >
std::tuple<std::vector<T>, std::vector<T>, std::vector<T> > HW1 (
            T t0,
            T t1,
            int n,
            T y0,
            T y0p,
            T theta0 )
```

HW1: Solve Kepler problem via finite difference Method Behavior of HW1 is undefined when type T is not equal to one of float, double, long double.

Parameters

t0	initial time
t1	final time
n	number of gird points to evaluate
y0	initial condition for zeta
y0p	intial condition for derivative of zeta
theta0	initial condition for theta

Returns

tuple of time, zeta and theta

See also

Finite difference Method

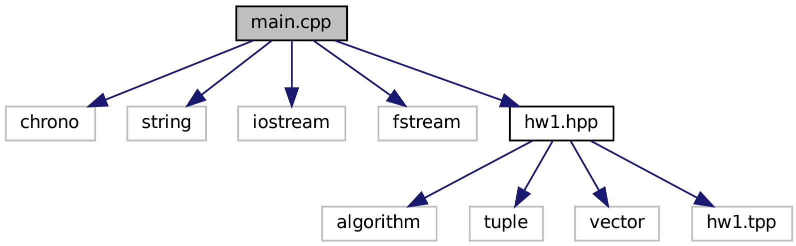
Theta

6.2 main.cpp File Reference

main program for homework1 of Computer1 class in Yonsei University Interactively reads inital condition, number of gird points to evaluate and output file name then computes and saves solution.

```
#include <chrono>
#include <string>
#include <iostream>
#include <fstream>
#include "hw1.hpp"
```

Include dependency graph for main.cpp:



Macros

- #define **PRECISION** float
- #define **DIGITS** 6

Functions

- int **main** (void)

6.2.1 Detailed Description

main program for homework1 of Computer1 class in Yonsei University Interactively reads inital condition, number of gird points to evaluate and output file name then computes and saves solution.

Author

pistack (Junho Lee)

Date

2021. 11. 3.

Index