

PHY 410-505 Computational Physics I

Chapter 1: A First Numerical Problem

Lecture 1

Monday August 25, 2008

Lecture Outline

Course Information	3
Computational Physics	5
Computation and Algorithms	5
A First Numerical Problem	7
Real world phenomenon	7
Mathematical Model	7
Computational problem	7
Numerical algorithm	8
Computer Programs in Mathematica and C++	9
First “Hello world” programs	9
Mathematica decay program	10
C++ decay program	11
Maple decay program	13
Display data using Gnuplot	13

Course Information

- Computational Physics, Giordano and Nakanishi, Second Edition
- Chapters 1–7, approximately 2 weeks per chapter
- Math prerequisites: differential equations, linear algebra
- Physics: undergraduate mechanics, electrodynamics, statistical physics
- Numerical analysis: Textbook appendices, Numerical Recipes
- Programming language: C++. (Can use Fortran or Java, but you are on your own.)
- Computational systems: Mathematica or Maple – use optionally as fancy calculator – available on UB computers and UBMicro
- Homework, exams, grades:

Weekly Homework Assignments (~ 3 textbook problems)	50%
Mid-term Exam (October 19)	15%
Final Exam	25%
Class Participation	10%

- Assignments will involve writing codes, running simulations, and analyzing results
 - Learn to code by studying example programs and making changes
 - Not difficult, but finding and correcting bugs requires time and patience!
 - Discuss assignments with classmates, but coding, simulation and writeup must be your own work.
 - Pass = 50%, A \approx 85%
-
- Operating systems and software
 - You can use Windows, Mac OS X, or Linux (ubuntu or UBLinux)
 - All needed software is available for free
 - Windows:** Microsoft Visual Studio Express 2008 or Dev-C++
 - Mac OS X:** Xcode
 - Linux:** GCC plus Emacs or Vi; KDevelop
 - Gnuplot for graphics, OpenGL for graphics and animation

Computational Physics

Physical System: In Science and Engineering we study physical phenomena in the real world – usually an isolated self-contained system – e.g., pendulum oscillations under gravity

Mathematical Model: Approximate the system using a mathematical model – e.g., an ordinary differential equation

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L} \sin \theta$$

Computer Simulation: Exact analytical solution may not be available – use a digital computer to generate an approximate numerical solution

Code Verification: Is the numerical solution accurate? Error analysis

Model Validation: Does the model and numerical solution agree with the real world – compare with experiment – data analysis

Computation and Algorithms

- Solutions of a mathematical model are functions

- A function is a map from a set of input points (domain of the function) to a set of output points (range of the function)
- A computation is a process that generates an output point from a given input – a computation evaluates the function
- An algorithm is a procedure (or recipe) that transforms an input point into an output point in a sequence of steps
- In a digital computation, the domain and range are discrete sets, and the algorithm terminates after a finite number of steps
- The computational complexity of an algorithm is important – it must give a result in a reasonable number of steps (time complexity) and using a reasonable amount of resources, e.g., computer memory (space complexity)

A First Numerical Problem

Real world phenomenon

- A collection of radioactive nuclei are observed to decay
- Measure decay times, measure mean lifetime τ

Mathematical Model

- Approximate discrete number of nuclei by real variable $N(t)$
- Differential equation

$$\frac{dN}{dt} = -\frac{N}{\tau}$$

- Exact solution

$$N(t) = N(0)e^{-t/\tau}$$

- Model validation: N must be large

Computational problem

- Given $N(0)$, τ , t find $N(t)$

Numerical algorithm

- Use Euler's algorithm §1.2 and Appendix A
- Discretize t using a time step Δt
- Taylor series

$$N(t + \Delta t) = N(t) + \frac{dN(t)}{dt}\Delta t + \dots$$

dots represent error in this approximation

- Use

$$\frac{dN}{dt} = -\frac{N}{\tau}$$

- Iterative (marching) algorithm:
 - Given $N(0)$ find $N(\Delta t) = N(0) - N(0)\Delta t/\tau$
 - Using $N(\Delta t)$ find $N(2\Delta t) = N(\Delta t) - N(\Delta t)\Delta t/\tau$
 - Repeat as often as needed
 - Monitor (...) errors which accumulate in each step
- Code verification: Compare numerical results with exact solution

Computer Programs in Mathematica and C++

First “Hello world” programs

- Mathematica: create a file `hello.m` with the following code:

```
Print["Hello, world!"]
```

- Start Mathematica and load the code (press Shift-Enter):

```
In[1]:= << hello.m
```

- C++: create a file `hello.cpp` with the following code:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

`hello.cpp`

■ Compile and execute the code:

```
> g++ -o hello hello.cpp  
> hello
```

Mathematica decay program

Type the following code directly in Mathematica, or load it from a file `decay.m`.

`decay.m`

```
(* Get input from user *)  
n = Input["Enter initial number of nuclei: "]  
tau = Input["Enter decay time constant tau: "]  
dt = Input["Enter time step dt: "]  
tMax = Input["Enter time to end simulation: "]  
  
(* Initialize variables and output list *)  
t = 0  
result = {{t, n}}  
  
(* Calculate results and store in list *)  
While[t < tMax,  
  n -= n / tau * dt;
```

```
t += dt;
result = Append[result, {t, n}]
]

(* Plot output and exact solution *)
euler = ListPlot[result, DisplayFunction->Identity]
exact = Plot[100 Exp[-t], {t, 0, 5}, DisplayFunction->Identity]
Show[euler, exact, DisplayFunction->$DisplayFunction]
```

C++ decay program

decay.cpp

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

int main() {

    // Get input from user
    cout << "Enter initial number of nuclei: ";
    double n;
    cin >> n;
    cout << "Enter decay time constant tau: ";
```

```
double tau;
cin >> tau;
cout << "Enter time step dt: ";
double dt;
cin >> dt;
cout << "Enter time to end simulation: ";
double tMax;
cin >> tMax;
cout << "Enter name of output file: ";
string name;
cin >> name;

// Initialize variables and open output file
double t = 0;
ofstream file(name.c_str());
file << t << '\t' << n << '\n';

// Calculate results and store in file
while (t < tMax) {
    n -= n / tau * dt;
    t += dt;
    file << t << '\t' << n << '\n';
}

// Close file
```

```
file.close();  
cout << "Results stored in " << name << endl;  
  
}
```

Maple decay program

Display data using Gnuplot

- C++ does not have standard graphics built in.
- Use Gnuplot (or Mathematica, Excel, or any plotting program) to display the results.

