Level 5 Computational Physics - Lecture 1

Dr. Simon Hanna

October 13, 2012

1 Introduction

The course is taught through a set of exercises, of gradually increasing complexity, supplemented by occasional lectures and weekly drop-in sessions.

There are many useful textbooks in the library. A few are:

- "Numerical Recipes" – Press et al.
- "Computer simulation methods" – Gould, Tobochnik, Christian (3rd ed).
- "Computational Physics" – Pang.

1.1 Course Objectives

- Develop your understanding of numerical techniques
- Gain experience in a computer language
- Gain experience in use of computers to solve physics problems
- Develop report-writing and analysis skills
- Carry out a substantial quantity of self-guided work

1.2 What is Computational Physics?

- Covers four key areas:
  - Symbolic & logical manipulation
  - Numerical analysis
  - Numerical simulation
  - Data acquisition, processing and analysis
- Why are we teaching it?
  - Allows the solution of otherwise intractable problems
  - Bridges the gap between theory and experiment
- Well-known applications
  - Fluid dynamics (e.g. design of vehicles, climate studies)
  - Classical and quantum dynamics (e.g. chaos, many body problem)
  - Particle physics & astrophysics simulation (e.g. hadronisation, galaxy formation)
  - Condensed matter problems (e.g. material properties from electronic structure)

1.3 Numerical Analysis

- The solution of mathematical problems using algorithms
  - Algorithm = "well-defined set of repeated steps for manipulating data"
  - Some problems can only be solved this way – there is no analytical method
  - e.g. find the roots of a quintic equation
- Typical topics for numerical analysis
  - Evaluating equations
  - Finding solutions to large sets of simultaneous equations
  - Solving differential equations; Integration
  - Optimisation i.e. finding the minimum or maximum value of a multidimensional function
- Application to physics problems
  - Physical principles may be well-known, but may have an analytically insoluble set of equations
  - We can often obtain an approximate solution by numerical methods

1.4 Numerical Simulation

- Predicting the behaviour of physical systems
  - Given a set of microscopic physical laws, how will a complex system behave?
  - What is the sensitivity of a system to initial conditions?
  - What can we learn about the physical laws from observing a large-scale system?
- Why is this useful?
  - Learn about systems which are impossible to construct experimentally
  - Study the sensitivity of systems to choice of parameters
  - Learn about the emergent collective behaviour of large systems e.g. fluid flow from motion of molecules (not obvious from the underlying simple equations)
  - Apply basic physics to the design of real, complex, systems

1.5 Course Schedule

- Lectures:
  - 9 a.m. Friday, weeks 1, 3, 5, 7
  - 9 a.m. Tuesday, week 9
- Drop-in sessions, 11 a.m. to 1 p.m. every Friday, beginning week 2, in Room 1.14 (Physics)
- Deadlines:
  - Ex. 1: Sun 21st Oct. 2012, at midnight
  - Ex. 2: Sun 11th Nov. 2012, at midnight
  - Ex. 3: Sun 2nd Dec. 2012, at midnight
  - Ex. 4: Sun 16th Dec. 2012, at midnight

1.6 Learning style

- Work independently
- But, do not work in isolation
  - Come to drop-in sessions
  - Exchanging ideas with other students is fine
  - Exchanging ideas with demonstrators is essential
  - Use textbooks – there are plenty in the library
- If you are in trouble, ask a demonstrator

1.7 Programming

- Approach:
  - Think about your program design before implementing anything
  - Much of your effort should be spent before touching a computer
- Code quality:
  - If your code is unintelligible, you will lose marks
  - Comment code (within reason), explaining the role of each section
  - Demonstrators will advise you on style and debugging tips as we go on

1.8 Reports

- Style and content:
  - Describe your understanding of the problem (can be very brief)
  - Outline your solution method at each step of the exercise
  - Show your results in a concise and appropriate form i.e. tables and graphs
  - Comment on your results, answering any explicit questions in the script
  - Identify any difficulties, problems, issues
- Length and format:
  - For exercises, recommend no more than one A4 page per section
  - Try to be concise; you are marked on understanding, not word count
  - Documents should be in Word or pdf format

1.9 Assessment & Feedback

- Assessment
  - Your report and code should be submitted via Blackboard
  - Exercises are assessed by your demonstrator
  - You will be assessed within three weeks of submission (provided work is handed in on time)
- Feedback
  - This is essential, and is a two-way process i.e. you should discuss it with your demonstrator
  - You can expect to receive feedback on your code at any drop-in session
  - You will receive a formal feedback form after each assessment

1.10 Marking Scheme

- Breakdown between exercises:
  - Summer Exercises: 20% of the total marks
  - Exercise 1: 20% of the total marks
  - Exercise 2: 20% of the total marks
  - Exercise 3: 20% of the total marks
  - Exercise 4: 20% of the total marks
- Allocation of marks:
  - Report: 50%
  - Computer code: 50%

1.11 Plagiarism

- Don't do it, you will be caught.
- The School of Physics has a zero-tolerance policy on this matter.
- If you are not sure what constitutes plagiarism
  - Look in the student handbook
  - Look at: www.bristol.ac.uk/library/support/findinginfo/plagiarism

2 Basic numerical problems

Here we briefly survey three numerical issues of importance when solving physics problems with a computer: numerical errors, series approximation and equation solving (root finding).

2.1 Sources of errors

- Rounding error (imprecision)
  - Computers (usually) use finite-precision arithmetic
  - A real number is stored as a single precision (32-bit) or double precision (64-bit) floating-point value in memory
  - This leads to rounding errors, sometimes avoidable, sometimes not
  - Single precision = roughly 7 digits of accuracy
  - Double precision = roughly 15 digits of accuracy
  - e.g. be careful with $x = a - b$ when $a \simeq b$ but $x \ll a$.
- Discretisation (or sampling) error
  - Results from approximating continuous systems by a set of sampled values
- Truncation error
  - Iterative algorithms must stop at some point, if you want an answer...
  - This will nearly always result in some error
- Inaccuracies in the model / algorithm
  - Are we solving / simulating the right thing? Does the algorithm work?!?

Numerical errors are inevitable. The skill is in estimating them, and avoiding forward propagation of errors.

2.2 Series expansions

- Many mathematical functions are approximated by series expansions.
- The expansions used in modern computer languages are very efficient.
- Simple Taylor expansions can be unreliable. e.g.

  $$e^x \simeq 1 + x + x^2/2! + x^3/3! + \ldots$$

  converges rapidly for small x:

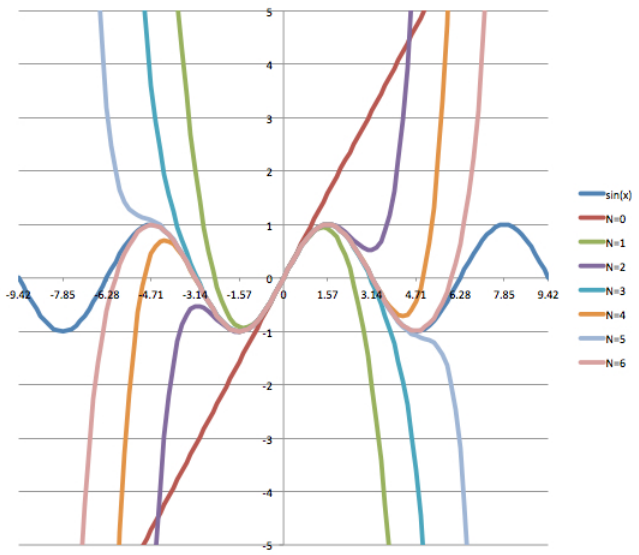
  $$e^{0.01} \simeq 1{+}0.01{+}0.00005{+}0.0000001667{+}\ldots$$

  but slowly for larger x:

  $$e^2 \simeq 1{+}2{+}2{+}1.333{+}0.667{+}0.267{+}0.089{+}\ldots$$

- Sometimes a series may fail to converge for some input values. e.g.

  $$\sin x = \sum_{n=0}^{N} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad \text{where} \quad N \longrightarrow \infty$$



This is an example of truncation error.

2.3 Root finding

You are familiar with the formula for finding the roots of a quadratic equation, and it is a simple matter to write a computer program to solve this. However, higher-order polynomial equations (quintic and above) cannot be solved in this way, and so some form of numerical technique is required.

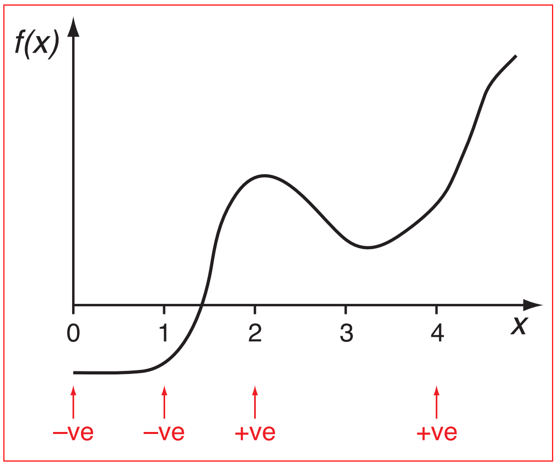
There are two basic methods you should know about:

- The bisection method;
- The Newton-Raphson method.

2.3.1 Bisection method

The method of bisection will reliably converge on the roots of all reasonable functions; however, it is very inefficient and only has a simple implementation in one dimension.

The bisection method is summarised as follows:



- The function $f(x)$ is initially bracketed by $[0, 4]$ i.e. $f(0) < 0$ and $f(4) > 0$.
- The first bisection is to $x = 2$. $f(2) > 0$, so the root must lie in the range $[0, 2]$.
- The second bisection is to $x = 1$. Since $f(1) < 0$, we adjust the range to $[1, 2]$, and so on until we achieve the accuracy required.

2.3.2 Newton-Raphson method

Need pictures for this and examples.

- The Newton-Raphson method uses more information (both the value and differential of the function) to converge more quickly, and works in many dimensions.

- The Newton-Raphson approach relies on expanding the function as a Taylor series:

$$\boxed{f(x) \approx f(a) + (x-a)f'(a)} \tag{1}$$

where a is our initial guess at the root, so that $(x-a)$ is small.

- We want to find x such that $f(x) = 0$. Therefore, rearranging Eq. (1) gives:

$$\boxed{x = a - \frac{f(a)}{f'(a)}} \tag{2}$$

This solution is only exact if $f(x)$ is linear.

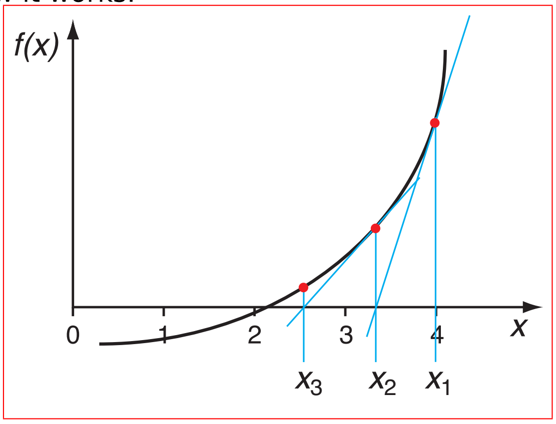
- In general Eq. (2) yields an improved guess, but we need to repeat the process:

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \tag{3}$$

iteratively until we have the required accuracy.

- The Newton-Raphson method may also be generalised to complex roots.

How it works:



- It is important to note a general issue with root finding. It is not usually possible to determine all roots of a function algorithmically, even if the parameters are bounded.
- The methods given above will only converge to local points of interest.
- You cannot always be sure how many or which roots you will find.
- A useful tip is to visualise your function by graphical means before starting.
- What happens to Eq. (3) if $f'(x_i) = 0$?