Physics 410: Computational Physics

This quarter we will focus on acquiring the tools and skills needed to tackle a range of interesting problems in computational physics. The basic building blocks of computational physics include the numerical solution of initial value problems for ordinary differential equations, numerical integration, root finding and linear algebra. We will study these topics, as we need them. One of the goals of the course is to enable you to use computation to tackle problems that are of current interest in physics. Although our primary focus will be on numerical computation, I will define computation broadly to include both symbolic calculations and the graphical representation of data. A long list of numbers is generally worthless, but a carefully designed graphical representation of that list may give new insight into the physics.

We will emphasize writing efficient code, debugging code and code validation. The primary language will be *Mathematica* but the principles of good programming are largely the same, regardless of the language used. We will also pay careful attention to how algorithms scale with respect to both memory and CPU time as the size of the problem increases. In addition we will be a small amount of programming in C.

Text

There is no formal text for the course (However, there is a list of references at the end of this document). I will supply *Mathematica* notebooks for the material we cover. All the course materials are in the Computational Physics\_2012 folder on Hauck.

Philosophy of the course

There are two views (at least) of what students should learn in a computational physics course. One view holds that it is important to write all of your own algorithms, never use canned (existing) code, know the details about how numbers are represented in computers and in general understand all of the nitty-gritty details. And there is something to be said for this view. The trouble is that if you adopt this view you don’t have time to address very many interesting physics problems in a one-quarter course. Professionals often use canned code.

Accordingly, we will often use the built–in routines in *Mathematica*, the trick is to know when you can trust the result and when you can’t. Usually you don’t need to know all the details of how an algorithm is implemented but sometimes you do and one thing that distinguishes good scientific programmers from bad ones is that good ones know when this matters. A constant theme will be: How do we know this result is right? How accurate is it? How accurate do we need it to be? Blindly trusting results just because they came from a computer is a grave sin. Don’t commit it.

Grading

There are no exams. Grades will be based on homework (50%) and on a project (50%). I expect that you will choose your own project (subject to my approval) but I can offer advice, suggestions and possible topics. You must have a preliminary project proposal by the end of the third week of the course. Projects require both a paper and a ten-minute oral presentation. I expect the papers to be well written and proofread. Long lines of computer code without explanatory text are unacceptable. The same guidelines also apply to the homework. Oral presentations should be well organized and practiced beforehand. See the guidelines for giving a good talk in the references section.

Pace of the course

This may be too optimistic and is subject to change.

Week 1 Initial and boundary value problems for ODE’s

Week 2 Eigenvalue problems for ODE’s

Week 3 Solving the wave equation

Week 4 Applications to wave packet dynamics

Week 5 Solving elliptical PDE’s on a grid, eigenvectors and eigenvalues of matrices

Week 6 Applications to electrostatics, drums and quantum mechanics

Week 7 Solution of elliptical PDE’s by eigenfunction expansions

Week 8 FDTD methods in electrodynamics

Week 9 GPU Programming in *Mathematica*

Week 10 Topics of class interest and project presentations

Exam Week Project presentations

Useful references

[1] *Numerical Recipes in C*, Willian Press, Saul Teukolsky, William Vetterling and Brian Flannery, Cambridge

[2] *An Introduction to Computational Physics*, Tao Pang, Cambridge

[3] *Computational Physics*, Rubin Landau, Manuel J. Páez and Cristian C. Bordeianu, Wily

[4] *Mathematica for Scientists and Engineers*, R. Gass, Prentice Hall

[5] *Numerical Computation Using C*, R. Glassey, Academic Press

[6] There are numerous C tutorials on the web. One is http://einstein.drexel.edu/courses/Comp\_Phys/General/C\_basics/

[7] John Wilkins’ guides to giving good talks

 a) <http://www.physics.ohio-state.edu/~wilkins/onepage/terribletalk.ps>

 b) <http://www.physics.ohio-state.edu/~wilkins/onepage/talk.ps>