Preface

These notes are intended for introductory and advanced courses in numerical methods and numerical analysis for physics students. The objective is for students to learn where approximations methods come from and to take them *behind-the-scenes* of problem-solving environments such as Mathematica, Maple, Matlab etc... and disect (and understand) the algorithms and numerics used. For example, the choice of Fortran95 as our primary language is to show the student the power of a compiled language in forcing you to deal directly with fundamental algorithms in computational physics.

The Philosophy

```
Know the numerical method/technique & understand the related algorithm.
Find (and if necessary, develop) the tool/code (e.g. F90, Mathematica).
Apply to the physics problem at hand.
```

Introductory Computational Physics (Sections 1-7) deals with the basic mathematical, numerical, and conceptual elements needed for using a computer as a virtual physical science laboratory. Each element will be studied in the context of simple projects, thus permitting the students to work independently and to understand at their own rate each element's virtues, areas of applicability, and limits. We start off by teaching students the basic tools (*Fortran 95, Makefiles, Gnuplot ...*), then we get on to the basics of computing: algorithms, precision, efficiency, and verification. From there we move to some necessary numerical analysis and its associated approximation and round-off error. Physical applications are used, including interpolation, extrapolation, least square fitting, cubic spline fitting, Runge-Kutta etc .... In the second stage of introductory computational physics, the students learn Mathematica and more programing in Fortran 95. During this learning stage they are given notebooks or codes to run and then modify (or debug) them as they follow along through worksheets. Eventually in later chapters, students will have to write their own Fortran routines, Mathematica notebooks, making Gnuplot figures using scripts, and presenting their assignments and exams in postscript or PDF format using Latex.

Advanced Computational Physics (Sections 8-11) focuses on realistic physics and astrophysics problems which apply and extend the techniques learned in the preceding course. The students are exposed to the concept of random numbers in physics and computation before introducing them to the world of Monte-Carlo techniques. We follow with more detailed learning of differential equations with specialization on *finite differences*. In advance computational physics, the students will be asked to develop and write their own codes and use the IDL (Interactive Data Language) tool which is crucial for visualization of Monte Carlo simulation results and Finite-difference simulations.

The ultimate goal is to teach students:

• How to break down a project into sub-problems; implementation issues (e.g., program design, code conventions, makefiles); use of graphics for visualisation; verification
• Good programming practices; how to test and debug a code (Mathematica, Fortran); how to tune a code to run faster.
• Errors and uncertainties in computations. E.g., one should understand how to analyse whether a calculation is limited by the algorithm or round-off error.
• Random numbers, Monte Carlo integration and simulation, matrix computing, and so on.

The numerical techniques and methods covered in these new courses, with a very high likelihood, will be utilised to study other complex systems found in science and engineering. Indeed students who in past years have taken such courses and worked on computational physics problems are now leaders in such diverse field as financial mathematics, petroleum exploration, traffic engineering, climate change, and medical imaging. By teaching students how to tackle physics and astrophysics numerically, we provide them with skills that are so necessary in today's IT world.

Computational Physics

`A multidisciplinary subject combining:`
`(i) Physics; (ii) Applied Mathematics; (iii) Computer Science`
The aim: `Solving realistic physics problems.`

Acknowledgements We first would like to thank Hugo Graumann for assistance during the early stages of development of the computer laboratory at the University of Calgary (Physics and Astronomy). Hugo has been instrumental in setting up Linux, Fortran 95, Mathematica, IDL and other softwares in the machines that allowed us to teach this course and run the labs with ease. Our thanks also go to Jan E. Staff and Brian P. Niebergal for their help and input during the development of this course and notes. Finally, we thank Anson Lam for going over a few sections checking for typos and missing documents.

Rachid Ouyed & Woflgang Dobler.