PHYS 3437 PROJECTS

Rob Thacker, January 2007
(revised from D. Clarke document)

Your term project is worth 30% of your grade, and will require at least as much effort on your part as preparing for a midterm and a final exam in classes where such examinations are part of the assessment. There are essentially three parts to your project, including selecting a topic, preparing your oral presentation, and preparing your written final report.

Selecting your topic

I have listed several possible projects, and give either details of the algorithm or references where you can find them at the end of this document. You may select one of these projects, or define one of your own. In either case, I need to know what problem you will be doing by the **first class in February**.

Midterm Report

Unlike previous years you are not required to supply a mid-term report. While this reduces your work load, it also requires that you be thoughtful in planning out your time allocation to the project. Leaving things until the last three weeks of term will almost certainly ensure you do not do very well.

Oral Presentation

Your oral presentation is worth 50% of your project grade (and thus 15% of your course grade), and will be made in class during the **last two weeks of classes**. You will have approximately $300/N$ minutes where N is the number of students in the class, to give your "colloquium", in which you will define your problem, indicate what physics you are using, indicate what numerical techniques or algorithms you used, show enough results from your tests to convince us your program and algorithms are working as needed, present your results, and finally draw some conclusions. Your talk should take on the form of a formal colloquium that professors and scientists give all the time to their colleagues when they wish to share the latest results of their research. Thus, you might want to come to the Departmental colloquia (Fridays at 3:00pm), if you are not already doing so, to get a feel for how these presentations are given. *I strongly advise you to practise your talk several times before giving it to the class to make sure your presentation fits into the allotted time.*

As in previous years an invitation to the department to attend projects will be extended, although in practice few take up this offer. We will also maintain the practice of having classmates grade presentations.

Final Report

Your final, written report, worth 50% of your project grade (and thus 15% of your course grade), is due on the **last day of classes**. The report should be written in paper format, using LaTeX (the same software that generated this document) using 12 point text, single spacing, and 1" margins on 8.5" × 11.0" paper (regular letter stock). The length of the main report should be between 10-20 pages although code listings can go beyond the 20th page. However, I as in previous years if the body (*i.e.* excluding bibliography and appendices) of the report extends beyond the 20th page, I will stop at page 20! You need to ensure the following points are covered:

- ABSTRACT: an abstract to give a quick overview of your paper;
- INTRODUCTION: a clear definition of the problem you are going to solve, including all the physics, initial conditions, *etc.*;
- METHOD: a discussion of the various algorithms available; which algorithm you have chosen, and why (along with any tests you have done to help you make your choice);
- RESULTS: the numerical solution to your problem, displayed in whatever format you think best illustrates it;
- DISCUSSION: limitations of your solutions (are there any numerical artifacts or features which concern you?); any suggestions you may have for improving the methodology;
- CONCLUSIONS;
- BIBLIOGRAPHY;
- APPENDIX: a program listing.

It is difficult to imagine how you would be able to accomplish all this with less than 10 pages.

A note on written English

In your report you must write in a style in keeping with the image you are trying to portray; that of a serious scholar. Therefore, you must write using grammatically correct English, you must avoid colloquialisms, slang, awkward sentence and paragraph structure, and you must spell correctly! **Be warned: a poorly written midterm or final report will not receive a passing grade.** If you have never learned how to write properly in English, it is high time you did!

As a guide, you might want to open a few professional journals to get a feel for the style of language used (or check out the paper I have placed on the web page). I have also created a link to a simple latex skeleton that you can use for your paper. If you want a gentler introduction to formally written English, read through some of the B.Sc. honours (blue) and M.Sc. theses (maroon) in the ICA reading room. As you already know, the written language is not the same as the spoken one in large part because all the reader has to go on is what appears in type in front of him or her. The lack of body language, gestures, facial expressions, *etc.*, means that the written language needs to be much more precise and efficient than the spoken one, and among other things, this demands strict adherence to the rules of grammar.

Suggestions for Term Projects

1. Solar System Dynamics.

Question: Could a binary star of masses 0.6 $M_\odot$ and 0.4 $M_\odot$ separated by say 0.1 AU host a stable planetary system? If for example, there were a few terrestrial planets within an AU of the centre of mass of the binary, and a Jovian planet at a few AU, would one or more of the planets be ejected from the system (or collide with one of the stars) in a time less than the lifetime of our own solar system?

— or —

Question: How does the presence of the outer planets affect the collision rate of comets falling in from the Oort cloud with the Earth? That is, do the Jovian planets help protect the Earth from such collisions, or make them more likely?

Numerical Techniques: Use a "leap-frog" style "N-body solver" to integrate Newton's law of gravitation for as long as necessary to obtain the desired solution(s). Problems involving planetary dynamics and stability require a relatively few number of particles, but can require integration over tens or hundreds of millions of orbits. Other applications of N-body solvers (*e.g.*, galactic dynamics as suggested in project 2) may require many thousands of particles, but integration over a relatively short period of time. In either case, criteria are needed in order to determine the optimal time step for integration, and to safeguard against two particles coming too close together (*e.g.*, softening parameters).

The equations of motion are from PHYS 1210:

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i, \quad \text{and} \tag{1}$$

$$\frac{d\vec{v}_i}{dt} = \vec{a}_i(r_{i,1}, r_{i,2}, \dots, r_{i,N}). \tag{2}$$

where $\vec{r}_i$ is the position of the ith particle (planet or star) of which there are N, $\vec{v}_i$ is the velocity, and $\vec{a}_i$ is the acceleration which depends upon the distances, $r_{i,j}$, $i \neq j$, between the ith particle and each of the other $N-1$ particles, namely,

$$\vec{a}_i = \sum_{j \neq i} \frac{Gm_j}{{r_{i,j}}^2} \hat{r}_{i,j}$$

where $\hat{r}_{i,j}$ is the unit vector pointing from the ith particle to the jth particle.

As these are vector equations, the scheme is a little complicated by having to keep track of three components. For simplicity, my discussion will be limited to 1-D.

Equations (1) and (2) are solved numerically by "differencing" them. Thus, we write:

$$\frac{r_i^{n+1} - r_i^n}{\delta t} = v_i^m, \quad \text{and} \tag{3}$$

$$\frac{v_i^{m+1} - v_i^m}{\delta t} = a_i^l(r). \tag{4}$$

where the superscripts represent the number of time steps from $t = t_0$, and δt is a small (but not infinitesimal as dt is) time step. Thus, r_i^{n+1} is the value of r_i at the $(n+1)$th time step (advanced) while r_i^n is the value of r_i at the nth time step (present), *etc.* In equation (3), presumably m (the superscript on v_i) represents some time between time step n and $n+1$.

Solving equations (3) and (4) for the advanced time steps, we get:

$$r_i^{n+1} = r_i^n + v_i^m\,\delta t, \quad \text{and} \tag{5}$$

$$v_i^{m+1} = v_i^m + a_i^l(r)\,\delta t. \tag{6}$$

So what time steps are best to choose for m and l? If $l = m = n$ for example, equations (5) and (6) become what is known as an *explicit integrator*, since everything on the right hand side are known explicitly (present values of r_i and v_i), and all we need to do to get the new values of r_i and v_i is the arithmetic.

However, you will find if you do this (and you should) that a simple circular orbit such as the earth moving around the sun will spiral outward, regardless of how small a time step you choose! You should draw a diagram to show how the explicit integrator advances an object moving in a circular orbit to convince yourself that a spiraling outward orbit is the expected result.

We may argue, then, that to fix this spiral-outward problem, we need to use a value of v_i in equation (5) that is somehow in between the nth and $(n+1)$th time steps. Surely $m = n + \frac{1}{2}$ is a reasonable guess. Then, by the same token, a_i in equation (6) should be evaluated half way between $m+1$ and m, or equivalently, half way between $n + \frac{3}{2}$ and $n + \frac{1}{2}$. Thus, take $l = n+1$. Therefore, equations (5) and (6) become:

$$r_i^{n+1} = r_i^n + v_i^{n+1/2}\,\delta t, \quad \text{and} \tag{7}$$

$$v_i^{n+3/2} = v_i^{n+1/2} + a_i^{n+1}(r'\text{s})\,\delta t. \tag{6}$$

Notice in this scheme that r_i and v_i are known at different times; they are forever "staggered" by half a time step relative to each other. This is the so-called "leap-frog" method, and when applied to the simple closed orbit, eliminates the spiraling out problem. The scheme still commits numerical errors (circular orbits become slightly elliptical), but the orbits are at least closed.

2. Galaxy Dynamics

Question" How is one elliptical galaxy (assembled from N_1 "stars") affected by the passage of another galaxy (assembled from N_2 "stars")?

This is the Toomre & Toomre problem whose seminal paper in 1972 entitled "Galactic bridges and tails" (ApJ, 178, 623) changed the way astrophysicists approached galactic dynamics. Like the first project on solar system dynamics, this is an application for an N-body solver for which the "leap-frog" method described above is an example. An ambitious student might want to search the web for more efficient N-body algorithms, including *tree codes* (look for papers by Lars Hernquist in the 1990s on the NASA ADS server) to allow the use of many more particles than a leap-frog method would allow. However, it would be quite sufficient if the student were to reproduce all that Toomre and Toomre published more than a generation ago.

As pointed out in project 1, this project will require relatively few integration steps (perhaps a few thousand rather than many millions), but with absolutely as many particles as you can get away with without grinding our computing resources into the ground. Even still, you will not be able to have anywhere near enough particles, even with sophisticated "tree-codes", to keep track of as many stars as there are in even a dwarf elliptical galaxy. Thus, each of your "stars" may well be thousands of solar masses, and you will rightly ask yourself: 'What does this mean for the realism of the simulation?', 'What should be done if two "stars" get too close, collide, go into orbit about each other', *etc.*

3. Fluid Dynamics

Question: What is the role of viscosity when a fluid, trapped between two rotating, concentric cylinders (Couette flow) with different angular velocity, undergoes the transition between laminar and turbulent motion?

— or —

Question: How are accretion discs formed around protostars?

— or —

Question: What happens when two jets collide?

References: I can provide some elementary texts on fluid dynamics, as well as give you a crash course on the fundamentals. You will use an existing hydrocode for this project. Rather than writing your own code, the onus will be on you to understand some of the principles behind a working, research-calibre code (ZEUS-3D), and to gain enough of an understanding of the physics to teach the class some elementary fluid dynamics. You may have to make some relatively minor modifications to one or two of the subroutines of the code, and perhaps write your own subroutine to set up the initial flow. I will give you lots of guidance here.

4. Fluid Dynamics: Bondi Flow

Question: What happens when spherically-symmetric accretion (known as *Bondi flow* after Sir Hermann Bondi who first published the analytical solution in the MNRAS in 1952, v. 112, p. 195) is perturbed with rotation? magnetic fields? in both 2-D and 3-D?

It is widely known that in 2-D, Bondi accretion with rotation results in a new steady-state solution characterised as a "fat accretion disc". This is similar to the structures formed around all proto-stellar objects before enough material can be accreted to the central mass to ignite hydrogen fusion. As with project 3, you would be using an existing hydrocode (ZEUS-3D) and picking up the project where previous students (*e.g.*, Joel Tanner) have left off.

5. Time-independent Schrödinger equation

Question: What are the electron energy levels in the helium atom? Lithium? Boron? *etc.*

References: "Computational Physics", by Stephen E. Koonin, Chapter 3.

6. Time-dependent Schrödinger equation

Question: How does a wave packet (say a Gaussian) reflect off a barrier? How much of a wave packet reflects off a wall of finite height and thickness, and how much tunnels through?

References: "Computational Physics", by Nicholas Giordano, Chapter 10.

It would be very instructive for you to make an animation from a series of "snapshots" of your simulation at closely spaced time intervals, so we could watch how the packet interacts with the various barriers, pits, *etc.* you subject it to. Can you show us what happens when two wave packets collide?

7. Poisson Solver

Question: What is the gravitational (electric) potential of an arbitrary distribution of matter (charge)?

Reference: "Numerical Recipes", by Press *et al.*, §19.5; pages 854-860).

Part I: Find the density distribution, $\rho(r)$, for a spherically symmetric isothermal cloud of gas in hydrostatic equilibrium. (The solution is called the *Bonnor-Ebert sphere*.) To do this, consider the governing equations:

$$\nabla^2\phi = 4\pi G\rho; \qquad \text{Poisson's equation}$$

$$\frac{1}{\rho}\nabla p + \nabla\phi = 0; \qquad \text{hydrostatic equilibrium}$$

$$p = c\rho; \quad c = \frac{kT}{\langle m \rangle}. \qquad \text{isothermal equation of state}$$

The density ρ could be a function of all three coordinate variables, and thus so could ϕ and p. For the Bonnor-Ebert sphere, however, we take $\rho = \rho(r)$ only (spherical symmetry assumed), and likewise for ϕ and p.

Manipulate these equations to obtain one second-order, ordinary differential equation for $\rho(r)$, then use your Runga-Kutta solver modified for 2nd order ODEs to solve $\rho(r)$ over a reasonable domain of r using the Cauchy boundary conditions ($\rho(0) = 1$, $\rho'(0) = 0$). Note that by setting $\rho(0) = 1$, your solution is scaled to the central density, which can be left as a free parameter.

Combine the equation of state with the equation for hydrostatic equilibrium to show that

$$\phi = -c \ln\left(\frac{\rho}{\rho_\infty}\right); \quad \text{where} \quad \rho_\infty = \lim_{r\to\infty} \rho(r).$$

Part II: Develop a computer program based on the *Successive Over-Relaxation* (SOR) method to solve $\phi(x, y, z)$ given an arbitrary density distribution, $\rho(x, y, z)$. Note: The *Numerical Recipes* algorithm is for 2-D only. The extension to 3-D is easy if you rewrite their equation (19.5.24) as:

$$\rho_{\text{Jacobi}} = \frac{\frac{1}{\Delta x^2}\cos(\pi/J) + \frac{1}{\Delta y^2}\cos(\pi/L)}{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}$$

for a $J \times L$ ($x \times y$) grid. Then, in 3-D, this generalises trivially to:

$$\rho_{\text{Jacobi}} = \frac{\frac{1}{\Delta x^2}\cos(\pi/J) + \frac{1}{\Delta y^2}\cos(\pi/L) + \frac{1}{\Delta z^2}\cos(\pi/M)}{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}$$

where, for $J = L = M = N$ and $\Delta x = \Delta y = \Delta z$, we have

$$\rho_{\text{Jacobi}} = \cos(\pi/N).$$

You should test your algorithm by making sure it can compute the correct gravitational potential (which you can compute analytically) for:

a) a single point mass at the centre of a 32^3 grid;

b) several point masses scattered about a 32^3 grid;

c) a uniform sphere (with a diameter half the width of a 32^3 grid);

d) your Bonnor-Ebert density distribution resolved on a 32^3 grid;

e) any arbitrary, non-spherically symmetric density (charge) distribution you can come up with.

You might examine both the *residual* and *error* of your SOR solver for the single point mass [test problem a) above] on a 32^3 grid. The residual is the absolute value of the difference between the present trial solution and the previous trial solution, averaged over all 32^3 grid points. The error is the absolute value of the difference between the present trial solution and the analytical solution, averaged over all 32^3 grid points. On a log-log scale, plot both the residual and error as a function of iteration for say 128 iterations. You should find that both quantities, after some initial transient fluctuations, settle onto a monotonically decreasing straight line, but then asymptote so the quality of the solution as measured by the residuals and errors cease to improve with iteration. Can you explain this asymptotic behaviour?

You might also test your algorithm for efficiency. Compute ϕ for a single point mass for four different grid resolutions, N^3, where $N = 16$, 32, 64, and 128. On a log-log scale, plot the CPU time it takes for your SOR solver to converge to the same error (say 10^{-6}) as a function of N, and show that the convergence time goes as N^4.

Note that the convergence time for the "last word" on Poisson solvers—a technique known as *Full Multi-Grid* (FMG; also outlined in Press *et al.*)—is $N^3 \ln(N)$ which in practise is enough of a difference to make SOR unusable for any large (M)HD simulation, but FMG eminently practical. Because an FMG solution is just as good as a SOR solution, SOR is rarely used anymore. Its only advantage is it is far far simpler to program than FMG.

8. Formulate your own project

In addition to the two texts of the same name ("Computational Physics"), one by Giordano, the other by Koonin, you have DeVries, and Garcia has a book called "Numerical Methods for Physicists" I can lend you. These books are all packed with ideas for projects such as these, and you are welcome to come up with your own.