2G03 Project 2024

When to start thinking about it: Now (see project1 lecture)
Consult with TA/Lecturer about idea: ASAP
When to start writing code: Anytime, as soon as you get a proposal approved
Deadlines:
1) **Written Proposal: Oct 7 deadline Sooner is ok – first come first served**
we give you detailed feedback as they come in
2) In class program demo: **mid November**
3) Final Write-up Due: **last week of class (Dec)**

Overview

Each student will complete an individual project: designing and writing a program that demonstrates the numerical solution of a scientific problem. The individual projects will be developed over the course of the term and consultation and advice will be readily available. The project consists of several parts:

1) Project proposal (short description)

Firstly the student will select a (scientifically) interesting system or situation that can be solved via a set of ordinary differential equations and then write a proposal that outlines the problem, what the equations are, what the variables in the equations mean, how they might solve it and how they might test their computational version of the system to see if it behaves correctly (i.e. like the real system).

It is expected that projects will identify a set of Ordinary Differential Equations (ODEs) that can model the chosen system found in a reliable reference. We expect you to write out these equations in your proposal, e.g.

$$\begin{aligned} \frac{dG}{dt} &= -a\,G\,R + b\,G & \text{Rate of change of Grass, } G \\ \frac{dG}{dt} &= -b\,W\,R + c\,RG & \text{Rate of change of Rabbits, } R \\ \frac{dW}{dt} &= -e\,W + f\,W\,R & \text{Rate of change of Wolves, } W \end{aligned}$$

You are *strongly* encouraged to talk with lecturers in other courses to identify potential projects in research areas related to those courses. Part of the goal of 2G03 is to prepare you for computer based research work and this contact could translate into a research opportunity or a summer job.

The proposal only needs to be a page and must contain at least two and preferrably three or more equations. A typical way to present the proposal is to describe a system you wish to model with a reference (e.g. to a web site, journal article, etc...), write down the standard equations and variables, discuss how you'll solve the mathematical model on a computer and what you will look for when you test it. A key requirement is how you will extend it into a unique project (e.g. add one more equation or make the equations more complex). We will help you refine this part but please make a first suggestion.

Note that it is external code or libraries to solve the equations or to write program that simply plots analytical solutions. Your program must actually solve the equations iteratively (numerically).

You are welcome to discuss your proposal with other students. However, the normal rules about plagiarism apply. Students who work together should choose different project areas. You are expected to ultimately present a project that is different from that of all other students. The preliminary parts may be similar (such as the first demo versions and tests you do).

Proposals are worth participation marks, about 10% of the project grade. If you don't hand one in or hand it in late you lose marks.

Example project research areas

The following suggested areas are very broad. You should select a more specialized topic within one of the following areas for your project. Note that the further down this list you go, the trickier the project is likely to be – consult with us! For example, some may not be easy to express simply using ODEs. If there is a choice – you should go with the ODE way of doing it.
- Population growth/infections (epidemiology and populations)
- Solar system/Orbit integration (astronomy)
- Kinematics, Collisions, Projectile Motion (physics)
- Electrical Circuit Modeling (engineering/physics)
- Electrostatics/Gravity (physics, astrophysics)
- Structure models for planets or stars (astrophysics)
- Reactions/Phase diagrams (thermochemistry)
- Molecular Dynamics (biochemistry)
- Evolution of traits (evolutionary biology)
- Nerve signals/Neurons (psychology, neuroscience)
- Diffusion models (materials, physics) [PDE]
- Economic Models/Stock Market (finance)
- Crystal growth (materials)
- Models of condensed matter systems (physics)

2) Demo Program

The goal of the project is to create a C++ program on phys-ugrad to demonstrate the solution of the model problem and show that the program approximates the behaviour of the real physical system in some way. You should do this in a directory called **project** in your home directory on phys-ugrad. You should generate a **Makefile** to make your runnable project in that directory.

We strongly recommend including some real-time plotting with your project so you can see it working and make figures for the write-up. The pgplot library is ideal for this – we will help you use it. See the example in **/home/2G03/ploty** for some basic plotting. See the example fake project in **/home/2G03/lorenz** for some ideas on how to use plotting in a project. Note that it is not required that your plots are interactive or clickable.

The best way is to make progress is to gradually improve your program – make sure it works first. When it does work on a basic case such as a simple test, save that version for your demo, e.g. **cp -r ˜/project ˜/demo** . This copies all the current files in your project to a new directory called **demo**. It should literally be possible to go to this directory, type **make** and then run the demo. You can have one program that does several things if you wish (e.g. asks the user).

The most basic demo would include a basic test with a known answer (e.g. circular orbit) or reproducing a plot in a paper or something similar. Other tests include proving that the steps (in time or space) that you take are small enough to be accurate. Accuracy means the results stay similar. Ideally you can also show energy or momentum is conserved or some similar conservation law for your system is obeyed. It could also include simple cases where the behaviour is predictable (e.g. doubling the food leads to faster population growth). The exact tests depend on the system you are looking at.

Demos are worth particpation marks, about 20% of the project grade. If you don't have a runnable demo by the Nov. due date you lose marks.

3) Extended Program

Once you have a demo that reproduces standard results you should extend it in a unique way for your project. Typical choices include exploring the parameter space to identify interesting behaviour and cases, adding a tweak to the equations to compensate for an obvious shortcoming of the model, adding in some extra science (e.g. another equation or two) that leads to a more complex behaviour or running bigger versions (such as versions with more objects, species, planets etc...).

Students sometimes have code problems with the extended project code so it doesn't work as planned or has a difficult to find bug. There will be class time and help to resolve these issues. If the problem is never resolved, the project can still be acceptable, especially if the demo worked. You can describe your plan in the write-up even if the code never quite did what you wanted.

You should also test your extended version. It should also converge (give the same plot) with smaller steps over the same interval. If something is conserved, it should show that. The extension is often designed to fix a short-coming of the basic system. You should show that the answer is now better or at least explore why it didn't work. In general, we grade you on how well you explored the system rather how realistic the extension is.

4) Final Write-up

The write-up should start with a brief introduction to the area you chose and the model that your program uses from that area. You can also describe more advanced approaches (that you did not explore with code). You should include references to papers, books and/or well-regarded web-sites (e.g. scientifically sound wiki-pedia entries).

In the method section, you show the equations you solved and the kind of solver you used. In many cases it will be leap-frog or Runge-Kutta (or another method presented in lectures) and you will not need to provide very much detail. If you something unusual, give a reference to where the method came from and also add comments in your code about it.

In the tests section, you describe how you tested your program to be sure the model is working as it should **with figures**. This usually includes the demo's plot(s). For orbits this might be a simple orbit. For populations or disease models it might be reproducing a figure from the original paper or source. You should include several figures in the write-up. Ideally you clearly describe the expected answer and the origin of any discrepancies. For time evolution, you might show that the answer is the same for smaller steps forward in time over the same interval or that a conserved property does not change significantly (e.g. within round-off errors). In general more tests are better, especially if they test something different to the previous tests and every test should have a figure rather than just a number or text saying "it worked".

In the results section, you evaluate the model include any extensions you made to it. This might include changing parameters or cases or modifying the basic model a bit. For this part you need to do something different to other students. It should be working correctly at this point but may not be a good approximation of the real world. You should show figures of how the model system behaves and compare it to real world outcomes or expectations. You can be as critical of the model as you like (the original and any modifications or ideas you tried).

Finally, you should put forward conclusions that you feel are justified by the exploration you did and discuss the possible alternative approaches, future work if you had time and so on.

The final working C++ code for your project should be present in the directory **project** in your home directory. **You must include your code in your project hand-in (all content of source files compiled into your final project)**. It is strongly discouraged to develop code off phys-ugrad or in any other programming language except C++. If you present a project (e.g. results and plots) without C++ code on phys-ugrad that could produce it, it may lead to problems with *academic* *dishonesty.* The best approach is to work direclty on phys-ugrad and get all your results by running code there. The grader will look for working code on phys-ugrad in your directory.

It is not acceptable to re-use work you or others have submitted for another course or project. You must develop a new project and write new C++ source code based on a description of the appropriate algorithm(s). Converting or using another code is considered academic dishonesty.

It *is* acceptable to use another code to test that your own code works by comparing results but you must explicitly say you did that. It is a nice way to verify. That other code could have been written by you for another course, etc...

Program Testing

For the programming part, you must demonstrate that the program works and that the results are correct. There are two parts to this: verification and validation.

Verfication indicates that the code compiles, runs and is actually doing what you say it should – i.e. no bugs. Verification typically comes first. This would typically show basic good behaviour: solutions increasing in the right direction, giving the same answer for smaller timesteps, conserving some quantity for a short interval.

Validation is showing that the model, as implemented in your program, is a realistic model of a real system. This includes showing the results replicate known cases where applicable. It is possible the code is right and model just does badly on some cases. You first make sure there isn't a bug and then go ahead and critique it!

If your problem and solution describe a physical system, you must test if the numbers coming out are physically reasonable. For example, a program to integrate the orbit of the Earth should have results where the Earth's orbital period is one year. Most scientific problems have parameters. If the results for your system are supposed to vary in a particular way as you change the parameters you need to show this happens with your program. As noted below, it is acceptable to use analytical solutions, another code or other source to independently generate correct answers for comparison. A detailed reference must be given in this case. Note that your program may not be able to model the system very well. This can be ok as long as you can describe how it fails (in some cases) and have a go at explaining why.

Your write-up should include figures showing results from your program. This should include at least a few figures/plots showing tests of basic code correctness (verification). It should also include at least one figure (usually several) showing how the model responded to changes and showcasing the unique direction you took the project at the end. The grader is not going to run your program and try to recreate the interesting cases – you need to include plots that demonstrate each result!

Key steps to completing the project

1. Choose a research area or interesting problem and think up some ideas
2. **REQUIRED**: 5 minute proposal discussion with T.A. or instructor. You must tell the T.A. your chosen research area and get feedback on where to look for background material. You can also bounce ideas off the T.A.'s or the lecturer at other times.
3. **REQUIRED by Oct 7th**: Hand in proposal (do it earlier if you can)
4. Feedback : individual feedback to each student and discussion in class (slides)
5. **REQUIRED**: 10 minute feedback session with T.A. or instructor. In this session, the T.A. will help you set the programming goal for your project. Note that you are free to consult further on the program as you work on it.
6. Code up your program and test it (class time will be available for this and help to get it to work)
7. **REQUIRED by mid Nov (Date TBA)**: Demo! Demonstrate a working program in tutorial/class time. The program can be a work in progess but it must at least run
8. **FINAL HAND IN, Dec**: Hand in write-up for the program

Final Write-up and Assessment

70 % of the project assessment will be a combined grade based on the quality of the final write-up, the program itself and how well you explored the model.

We will look at: the writing quality, particularly the introduction about the research area; your explanation of the equations and terms in them; working basic code that makes plots; basic testing to show your code is correct (plots); exploration of your model and what parameters do (plots), including a scientific interpretation of different cases; the unique extension of your model including justification and testing of it (more plots!). The bottom line is that you should attempt to confirm that it is a good model by testing it. Note that it is not your model: if it is not good because they made bad or unrealistic assumptions you show that and talk about it. This is part of a good write-up. Do not assume we will run your code to see any results. Include the plot in your write-up if you want to get credit for it.

30 % of the project grade will be a fixed participation grade which includes handing in a proposal and doing the in-class demo.