Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.
- You should include tests of your algorithms. This could be represented by unit tests and/or tests of mathematical aspects of the algorithm.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use your github address to hand in your projects.
- Make a folder for each project. For each project you should have three folders: one for the code files, one for the report and finally a folder with specific benchmark calculations. The latter can be in the form of output from your code for a selected set of runs and input parameters.

Finally, we encourage you to work two and two together. Optimal working groups consist of 2-3 students. You can then hand in a common report.

Project 3, building a model for the solar system, deadline April 1

The aim of this project is to develop a code for simulating the solar system. Parts of this code can reused in two of the versions of project 4 (Galaxy model and Molecular Dynamics simulation). In the first part however, we will limit ourselves (in order to test our Verlet Runge-Kutta solvers) to a hypothetical solar system with one planet, say Earth, which orbits around the Sun. The only force in the problem is gravity. Newton's law of gravitation is given by a force F_G

$$F_G = \frac{GM_{\mathrm{sun}}M_{\mathrm{Earth}}}{r^2},$$

where M_{sun} is the mass of the Sun and M_{Earth} is the mass of Earth. The gravitational constant is G and r is the distance between Earth and the Sun. We assume that the sun has a mass which is much larger than that of Earth. We can therefore safely neglect the motion of the sun in this problem. In the first part of this project, your aim is to compute the motion of the Earth using different methods for solving ordinary differential equations.

We assume that the orbit of Earth around the Sun is co-planar, and we take this to be the xy-plane (you can extend your code to three dimensions as well). Using Newton's second law of motion we get the following equations

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{\mathrm{Earth}}},$$

and

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{\mathrm{Earth}}},$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force.

a) Rewrite the above second-order ordinary differential equations as a set of coupled first order differential equations. Write also these equations in terms of dimensionless variables. However, as an alternative to the usage of dimensionless variables, you could also use so-called astronomical units (AU as abbreviation). This is a common approach in such simulations. If you choose the latter set of units, one astronomical unit of length, known as 1 AU, is the average distance between the Sun and Earth, that is 1 AU $= 1.5\times 10^{11}$ m. It can also be convenient to use years instead of seconds since years match better the solar system. The mass of the Sun is $M_{\mathrm{sun}} = M_{\odot} = 2\times 10^{30}$ kg. The mass of Earth is $M_{\mathrm{Earth}} = 6\times 10^{24}$ kg. The mass of other planets like Jupiter is $M_{\mathrm{Jupiter}} = 1.9\times 10^{27}$ kg and its distance to the Sun is 5.20 AU. Similar numbers for Mars are $M_{\mathrm{Mars}} = 6.6\times 10^{23}$ kg and 1.52 AU, for Venus $M_{\mathrm{Venus}} = 4.9\times 10^{24}$ kg and 0.72 AU, for Saturn are $M_{\mathrm{Saturn}} = 5.5\times 10^{26}$ kg and 9.54 AU, for Mercury are $M_{\mathrm{Mercury}} = 2.4\times 10^{23}$ kg and 0.39 AU, for Uranus are $M_{\mathrm{Uranus}} = 8.8\times 10^{25}$ kg and 19.19 AU, for Neptun are $M_{\mathrm{Neptun}} = 1.03\times 10^{26}$ kg and 30.06 AU and for Pluto are $M_{\mathrm{Pluto}} = 1.31\times 10^{22}$ kg and 39.53 AU. Pluto is no longer considered a planet, but we add it here for historical reasons.

Finally, mass units can be obtained by using the fact that Earth's orbit is almost circular around the Sun. For circular motion we know that the force must obey the following relation

$$F_G = \frac{M_{\mathrm{Earth}} v^2}{r} = \frac{G M_\odot M_{\mathrm{Earth}}}{r^2},$$

where v is the velocity of Earth. The latter equation can be used to show that

$$v^2 r = G M_\odot = 4\pi^2 \mathrm{AU}^3/\mathrm{yr}^2.$$

Discretize the above differential equations and set up an algorithm for solving these equations using the so-called Verlet and Runge-Kutta 4 (RK4 hereafter) methods discussed in the lecture notes, chapter 8.

b) Write then a program which solves the above differential equations for the Earth-Sun system using the RK4 method and the Verlet method. Your code should now be object-oriented. Try to figure out which parts and operations could be written as classes and generalized (hint: one possibility is to write a class which returns the distance between the various objects. Or, you could write a planet class which contains relevant data about different planets). Your task here is to think of the program flow and figure out which parts can be abstracted and reused for many types of operations. Object orientation, with examples that apply to this project will be discussed right after spring break.

c) Find out which initial value for the velocity that gives a circular orbit and test the stability of your algorithm as function of different time steps Δt. Make a plot of the results you obtain for the position of Earth (plot the x and y values) orbiting the Sun.

   Check also for the case of a circular orbit that both the kinetic and the potential energies are constants. Check also that the angular momentum is a constant. Explain why these quantities are conserved.

d) Consider then a planet which begins at a distance of 1 AU from the sun. Find out by trial and error what the initial velocity must be in order for the planet to escape from the sun. Can you find an exact answer?

e) We will now study the three-body problem, still with the Sun kept fixed at the center but including Jupiter (the most massive planet in the solar system, having a mass that is approximately 1000 times smaller than that of the Sun) together with Earth. This leads us to a three-body problem. Without Jupiter, Earth's motion is stable and unchanging with time. The aim here is to find out how much Jupiter alters Earth's motion.

   The program you have developed can easily be modified by simply adding the magnitude of the force betweem Earth and Jupiter.

   This force is given again by

$$F_{\mathrm{Earth-Jupiter}} = \frac{G M_{\mathrm{Jupiter}} M_{\mathrm{Earth}}}{r^2_{\mathrm{Earth-Jupiter}}},$$

   where M_{Jupiter} is the mass of the sun and M_{Earth} is the mass of Earth. The gravitational constant is G and $r_{\mathrm{Earth-Jupiter}}$ is the distance between Earth and Jupiter.

We assume again that the orbits of the two planets are co-planar, and we take this to be the xy-plane. Modify your first-order differential equations in order to accomodate both the motion of Earth and Jupiter by taking into account the distance in x and y between Earth and Jupiter. Set up the algorithm and plot the positions of Earth and Jupiter using the fourth-order Runge-Kutta method. Discuss the stability of the solutions using your Verlet and RK4 solvers.

Repeat the calculations by increasing the mass of Jupiter by a factor of 10 and 1000 and plot the position of Earth. Study again the stability of the Verlet and RK4 solvers.

f) Finally, using our Verlet and RK4 solvers, we carry out a real three-body calculation where all three systems, Earth, Jupiter and the Sun are in motion. To do this, choose the center-of-mass position of the three-body system as the origin rather than the position of the sun. Give the sun an initial velocity which makes the total momentum of the system exactly zero (the center-of-mass will remain fixed). Compare these results with those from the previous exercise and comment your results. Extend your program to include all planets in the solar system (if you have time, you can also include the various moons, but it is not required) and discuss your results. Try to find data for the initial positions and velocities for all planets.

g) The perihelion precession of Mercury. This part is optional and gives you an additional score of 30 points (on top of the maximum of 100, that is you can obtain 130 points).

An important test of the general theory of relativity was to compare its prediction for the perihelion precession of Mercury to the observed value. The observed value of the perihelion precession, when all classical effects (such as the perturbation of the orbit due to gravitational attraction from the other planets) are subtracted, is $43''$ (43 arc seconds) per century.

Closed elliptical orbits are a special feature of the Newtonian $1/r^2$ force. In general, any correction to the pure $1/r^2$ behaviour will lead to an orbit which is not closed, i.e. after one complete orbit around the Sun, the planet will not be at exactly the same position as it started. If the correction is small, then each orbit around the Sun will be almost the same as the classical ellipse, and the orbit can be thought of as an ellipse whose orientation in space slowly rotates. In other words, the perihelion of the ellipse slowly precesses around the Sun.

You will now study the orbit of Mercury around the Sun, adding a general relativistic correction to the Newtonian gravitational force, so that the force becomes

$$F_G = \frac{GM_{\text{Sun}}M_{\text{Mercury}}}{r^2}\left[1 + \frac{3l^2}{r^2c^2}\right]$$

where M_{Mercury} is the mass of Mercury, r is the distance between Mercury and the Sun, $l = |\vec{r} \times \vec{v}|$ is the magnitude of Mercury's orbital angular momentum per unit mass, and c is the speed of light in vacuum. Run a simulation over one century of Mercury's orbit around the Sun with no other planets present, starting with Mercury at perihelion on the x axis. Check then the value of the perihelion angle θ_{p}, using

$$\tan\theta_{\text{p}} = \frac{y_{\text{p}}}{x_{\text{p}}}$$

where x_{p} (y_{p}) is the x (y) position of Mercury at perihelion, i.e. at the point where Mercury is at its closest to the Sun. You may use that the speed of Mercury at perihelion is 12.44 AU/yr, and that the distance to the Sun at perihelion is 0.3075 AU. You need to make sure that the time resolution used in your simulation is sufficient, for example by checking that the perihelion precession you get with a pure Newtonian force is at least a few orders of magnitude smaller than the observed perihelion precession of Mercury. Can the observed perihelion precession of Mercury be explained by the general theory of relativity?