Astrophysical Exercises: Self-gravitating N-body Systems

Joachim Köppen Strasbourg 2002

1. Physical Background

The big astronomical objects, such as galaxy clusters, galaxies, and stellar clusters are composed of many smaller objects (galaxies, stars, gas clouds) which interact gravitationally. This determines the structure and the evolution of these objects. With modern computers one can easily follow the evolution of such a system of a large number of particles, and thus study not only galaxies, stellar clusters, and planetary systems, but often one applies these techniques to similar problems, such as the evolution of the clumps that make up an interstellar gas cloud. Also, one can follow the collision of interacting particles and study how galaxies collide, or stars or even subatomic particles. In this exercise, we write a simplified program that computes the movements of a large number of mass points under the influence of their gravitational attraction. It is easy to generalize this program to other, additional forces.

2. The Equations

We consider a system of n points of mass m_i with positions $\mathbf{r}_i$ and velocity $\mathbf{v}_i$ at time t. Thus the force experienced by the i-th point is the (vector) sum of all the forces exerted by all the other particles:

$$\mathbf{Force}_i = \sum_{k=1}^{n}{}' \frac{G m_i m_k}{d_{ik}^3} \mathbf{r}_{ik} \tag{1}$$

The quote on the summation sign means that we sum over all $k \neq i$, since the mass point does not attract itself. G ist the gravitational constant, and $\mathbf{r}_{ik} = \mathbf{r}_i - \mathbf{r}_k$ is the radius vector from particle i to k, with the distance $d_{ik} = |\mathbf{r}_{ik}|$.

Each mass point experiences an acceleration $\mathbf{a}_i$ due to this force:

$$m_i \mathbf{a}_i = \mathbf{Force}_i \tag{2}$$

which changes the velocity and the position of the particle:

$$\frac{d^2}{dt^2}\mathbf{r}_i = \frac{d}{dt}\mathbf{v}_i = \mathbf{a}_i \tag{3}$$

These are all the equations that rule the movements of each particle, and thus the evolution of the whole n-body system.

2.1 How to Compute it

The equations (1) to (3) are vector equations, so we write them down for each component separately. For simplicity, we use cartesian coordinates $\mathbf{r}_i = (x_i, y_i, z_i)$. This gives for the x-component of the acceleration of the i-th particle:

$$a_{x,i} = \sum_{k=1}^{n}{}' \frac{Gm_k}{d_{ik}^3}(x_i - x_k) \tag{4}$$

To do the integration of time, we take a constant time step Δt, during which we assume that the acceleration at the 'old' time t is constant in each component and for each particle. Thus the x-component of the velocity of the i-th particle will change during this time step:

$$v_{x,i}(t + \Delta t) = v_{x,i}(t) + a_{x,i}\Delta t \tag{5}$$

Of course, during this time interval, the acceleration changes, as the mass point flies to another position. So what one really needs for the acceleration is a suitable mean value for each time step. But in order to compute this, we need to know the new position, ... which is what we are just going to compute. So, such a more accurate method (which is called *implicit* since it uses information about the new position) needs an iteration for each time step and it is more complex to program. For the time being, we shall use the simpler method, but please remember, it is less accurate. It is called an *explicit* method, as it uses only information from the old time.

To compute the new position of the mass point at the new time $t + \Delta t$ we assume that the velocity is constant. However, the same question as before is raised: which velocity shall we use? We have two: the old one $v(t)$, or the new one $v(t + \Delta t)$, and we could even take some average value. We recommend to use the *new* velocities. The advantage of doing this is the following: One can simply show that the change of the angular momentum during a time step will be zero, if we use this mixed method (Please check this by calculating analytically $\Delta\mathbf{L} = \mathbf{r}(t + \Delta t) \,\mathtt{X}\, \mathbf{v}(t + \Delta t) - \mathbf{r}(t) \,\mathtt{X}\, \mathbf{v}(t)$ from Eqs. (5) and (6), and also by observing the components of $\mathbf{L}$ in the numerical calculations). So the new positions are

$$x_i(t + \Delta t) = x_i(t) + v_{x,i}(t + \Delta t)\Delta t \tag{6}$$

Since we now use some information from the new time, the method is implicit. These computations are done for all the points, and after all the new positions are ready, one repeats the operation for the next time time step.......

2.2 Presentation of results

While a direct plot of the instantaneous positions of the point masses, projected into some plane – say the (x, y)-plane – is helpful to do the first checks with very simple models, and to get experiences with these systems, the plot quickly becomes quite filled with a mass of lines.

Often it is helpful to plot just the distance of each point from the centre-of-mass, whose x-coordinate is:

$$x_0 = \sum_{i=1}^{n} m_i x_i / \sum_{i=1}^{n} m_i \tag{7}$$

The other coordinates are computed analoguously.

2.3 Checks of the Method

Before one applies one's program to complex problems, one must check whether it really does what it was designed to do. The ideal way is to run it for simple problems that have an analytical solution. For example:

1. compute the orbit around the sun of a planet which has circular velocity: take two masses, a big one at the centre ($x = y = z = 0$ and $v_x = v_y = v_z = 0$) and a very small one at ($x = r$, $y = z = 0$) with initial velocity ($v_x = v_z = 0$, $v_y = v_c$). v_c is the velocity of a circular orbit with radius r. The planet should not only make a perfect circle around the sun, and come back to the initial position (within the orbital period – that you had computed from the basic formulae!), but also keep staying on the circle for as many cycles as possible. The accuracy depends on the time step you had chosen.
2. check other initial velocities. They should give elliptic orbits, and if one exceeds the escape velocity, the planet should never come back!
3. compute the kinetic and potential energies of the whole system, as well as the modulus or the components of the angular momentum vector. Plot them as a function of time. Since we do not feed in any energy or angular momentum, they must stay constant. Do they ???? How well does your program conserve energy and momentum? Dependence on time step.

3. General Remarks, Hints, and Kinks (I know you won't read this!!!)

0. Before actually writing the program, do make a flow chart diagram in order to understand the sequence of what is computed; a diagram of the program structure and the data structure to find out, how the loops and iterations are nested, which data from earlier parts you need at each section, which kind of vectors and arrays you are going to need. This may seem bureaucratic, boring or even old fashioned, but **don't start typing anything, before you are absolutely clear about what you plan to do**. Otherwise you may really end

up wasting much time in trying to find the logical errors, loopholes, and cul-de-sacs of your hasty programming. Save yourself the frustration, disappointment, and anger!

1. General Program Planning: It is a good idea to lay out the program as general as possible. This makes it easier to include other effects, or to try out other situations. Expanding the program to other force laws, e.g. repelling, as for the a cluster of protons instaed of stars, or completely different dependence on distance.
2. Modular Construction: It is also a good idea to break up the program into mathematically or physically sensible units. This allows a better testing of these individual modules — and most of the time is spent in tracing an error — a more flexible use of them for other purposes, and their exchange against improved or alternative methods, improved data, other physical processes, etc. For example, if the time integration is contained in a separate unit, one simply exchanges this against a more sophisticated method, if need arises, but without the trouble of having to change the program at a dozen places. For testing, a simple main program has to be written which supplies the necessary input data to this unit. When adapting the program to a different problem, one merely re-arranges the modules. The main program may then be just a control program to call the subprograms in the particular order, and gets and supplies data from and to each part.
3. Check Everything by Hand: Often, we underestimate our ingenuity to make small logical mistakes or simple typing errors, which may cause faulty results. The worst kind of mistakes are those which produce results that look as one would expect them to be. Do take the trouble of check everything the program does, until you are sure it does only what you want it to do. In programs about physical things, basic physics must be obeyed: conservation of particles, energy, etc. Also, all the simple and limiting cases which we do understand, must be reproduced accurately.
4. Provide Error Messages and Tracing: Especially during testing, you will print out everything that is useful to judge on what the program does and what decisions it makes. For example, have a print out of the energies and angular momentum of the system. It is recommendable to define one or more variables that can be used to switch on these print-outs. In this way, one can always check every part of the program, even after it has been considered finished. If you later want to try out some modification, and the results are either complete rubbish (because you made some error) or you just want to understand how the solution behaves, this option is very useful. Provide written messages if something goes not normal, e.g. if the automatic stepwidth goes below or above specified values. This become more important as the program grows in size and complexity.
5. Take Time For Comments: Don't be lazy with putting comments into the program. Not only for the general description what each subprogram does, what data it needs, and what variables it changes. But also if you change just a line for a test. Often one forgets after a few days about it, and is quite surprised about

the results. Save yourself the panic! Plenty of comments are vital, if you find some time later that you might use it for something, but you can't remember about its inner workings. Don't wait for them after "the program is finished". It never happens, or you won't have the time.

6. Be highly skeptic of anything the program produces.
7. When you are making tests, and later running the program for various situations and parameters, try to keep a careful written record of what you do, noting input parameters and results. This will make it easier for you later to compare results with earlier ones, in case you have to hunt for an error that has crept in yesterday when you "just changed a few things, almost nothing — but the program doesn't work any more".

4. Applications

4.1 Dangerous close encounters

When one computes the evolution of a cluster of stars, sometimes two mass points come quite close to each other, and the accelerations may become very large, hence the velocities, so that the particle moves within one time step over a long distance, and the assumption that the acceleration does not change, becomes very poor. Sometimes even the program will crash, as the numbers become too large for the computer.

One simple practical – but not physically correct – cure is to limit the force by using a **'softening' parameter** in the gravitational force:

$$\mathbf{Force} = {\sum_{k=1}^{n}}' \frac{G m_i m_k}{d_{ik}^2 + \epsilon^2} \cdot \frac{\mathbf{r}_{ik}}{d_{ik}} \tag{8}$$

In this way, one can safely do the computations. Check how the value for ϵ changes the evolution of simple systems, and how it affects the conservation of energy and angular momentum.

Another approach is to increase the accuracy of the program, by using better integration methods e.g. Runge-Kutta. You are invited to experiment with this.

An important tool to keep the accuracy during close encounters is to have an **adaptive time step width**. Here one would choose a time step so that the all the velocity components of all the particles do not change by more than a specified amount, e.g. 1 percent. Try it out!

Though this method will permit to follow through a close encounter quite accurately, it makes the program to slow down enormously every time two particles come rather close. Professional programs use various clever tricks, and they are very efficient. An example is the code of Aarseth which is described in Binney and Tremaine's book "Galactic Dynamics". There is a listing of a simple version of this program in the back of the book. But there **is** a bug in the program! Why don't you get some idea from this program, and try out this method in some simple way?

4.2 Collisions of asteroids and comets

In recent years, we have become aware of the fact that there are a great number of asteroids flying about in the Solar System, and that these bodies have sometimes collided with the Earth and caused great damage, probably including the extinction of the dinosaurs. What could we do, if we find out that a large asteroid (say 10 km diametre) will be colliding with us sometime in the future threatening to wipe out humanity? If one wanted to deflect the asteroid on its orbit, the energy required would be so huge that even all our nuclear weapon available today might not be sufficient. Apart from the question how to bring all the explosives to the asteroid...

My students at the International Space University in 2001/02 followed up the idea that it would be much more efficient if we could find a smaller asteroid which would come close to the offending asteroid, change its orbit a bit so that it would collide with the large asteroid, and thus change its orbit. In doing so, one would use the huge kinetic energy of the orbital motion.

Detailed studies of asteroids have shown that many of them do not consist of solid rock, but are rather a 'rubble pile' (une amas des cailloux, on peut dire). This means that a collision of two asteroids will not be elastic, like billard balls, but inelastic, and some part of the kinetic energy will go into 'heating up' the large asteroid, even cause it to break up.

We can study this by the n-body code: we shall assume that each asteroid is modelled as a cloud of particles which are gravitationally bound, so that they always remain together within a certain size. This is a simplification, because in a real asteroid there are also forces of cohesion which make the dust particles and rocks stick together.

Thus the first step is to make a stable, gravitationally bound cluster of particles (or stars or galaxies ...). This is the same problem which one deals in stellar dynamics and cinematics of galaxies! The book "Galactic Dynamics" by Binney and Tremaine (available in the DEA library) has a whole chapter devoted to autogravitating configurations of matter. We know that the density distribution of a gas with a velocity dispersion σ in a (spherically symmetric) potential $\Phi(r)$ is given by

$$\rho(r) = \rho_0 \exp(-\frac{\Phi(r)}{\sigma^2})$$

with a central density ρ_0 which is determined by the total mass

$$M_{\text{total}} = \int_0^\infty 4\pi r^2 \rho(r) dr$$

If the potential is due to the matter itself,

$$\Phi(r) = const. - G \int_0^r 4\pi r^2 \rho(r) dr$$

In principle, we should solve these equations consistently....

But let us use our code to find the equilibrium configuration by trial and error: The density will have a maximum at the centre, so let us simply assume a Gaussian form

$$\rho(r) = \rho_0 \exp(-(r/R)^2)$$

with a characteristic radius R which would be of the same order of magnitude as the asteroid's true radius. We shall create particle coordinates so that the particles' space density is equal to this density: With a random number generator (Numerical Recipes) we draw three numbers (distributed uniformly between 0 and 1); the first one is transformed into the interval 0 ... 2π for the azimutal angleϕ, the second one is transformed (Numerical Recipes: transformation method to convert a uniformly distributed random number into one which is distributed according to a given function) into angle θ which should be distributed like $\cos(\theta)$ – this ensures that we create random orientations uniformly distributed over the entire sphere. The third number is transformed into a random r which must be distributed like $r^2\rho(r)$, since we demand spherical symmetry.

We also need to assign an initial velocity vector: again, we draw three random numbers, again we create random angles ϕ_v and θ_v in the same way as before, but for the modulus v of the velocity vector we shall assume that it is distributed like a Maxwell-Boltzmann distribution: $v^2 \exp(-(v/\sigma)^2/2)$, with a free parameter σ.

We select a suitable softening parameter: ϵ should certainly be less than one-tenth of the asteroid's radius in order to see the details of the collision, but not too small because then the timestep must be made too small.

Then we start the simulation and observe whether the particles stream towards the centre, i.e. whether the initial configuration collapses. This means that the velocity dispersion σ had been too small to balance the gravitational attraction. If on the other hand, the particles move away from each other, and the asteroid explodes or evaporates, the initital kinetic energy was too large. A few tries may give a reasonably stable configuration. You might observe some oscillations of the entire cloud of particles. There will probably be a simple relationship between R, σ, and the mass of the asteroid.

After you've found how to make a stable rubble pile, take a large one (10 km diametre, mean density 3 g/cm^3). It is convenient to observe everything from the system of coordinates in which the big asteroid is at rest – we shall neglect its orbital motion about the Sun and also the gravitational fields of the Sun and the planets.

The smaller 'striker' asteroid (say 1 km diametre) comes in from some distance (not too far, otherwise we waste computing time) with some initial speed and direction. And then let them collide, and let's see what happens: under which conditions does the big asteroid remain intact? if it remains intact, how much is its speed changed? what is the fate of the smaller asteroid – does it merge with the big one? ... If one marks the particles belonging to the two bodies with different colours, one can see how the matter of each body is distributed in the resulting configuration.

The principal parametres are the mass ratio M_2/M_1, the initial speed difference $\Delta v = v_2 - v_1$, and the impact parameter that is the distance of their centres of mass at closest approach, if the asteroids were single point masses.

Suggestion: For the case of central collision – zero impact parameter – application of the conservations of momentum and energy before and after the collision (your first year in physics!) gives you simple formulae for the change of velocity of the big asteroid and all other quantities of the problem, both for the perfectly elastic and inelastic collisions.

4.3 Perturbations of planetary orbits

Consider the following situation: a planet is on a circular orbit around a massive star. Put another planet at some other distance from the star, also on a circular orbit. If the second planets' mass is sufficiently large (how large??) the first planet will no longer remain on a circular orbit. As one can see from a plot of the distances from the centre-of-mass as a function of time, it changes its distance to the sun periodically: it has an elliptic orbit. If the mass of the second planet become comparable to the stellar mass, the star also orbits the centre-of-mass! Investigate what can happen to the orbit of the first planet, and find out under which conditions (ratio of the orbital axes) interesting effects occur.

Now put a planet on a circular orbit around the star. Also place a number of **very** small mass points on circular orbits on various distances from the star, both between the planet and the star and further away. Make the masses very small indeed – or even put them to zero, so that they do not interact with each other. As the planet and the 'asteroids' circle the star, the planet will cause changes to the orbits of the test particles. Some of them even will be ejected from the planet system. This is what has happened to Saturn's rings and to the asteroids, which are influenced by Jupiter. How close to a planet can a stable orbit for an asteroid exist?

4.4 Stellar clusters: collapse and evaporation

For the following task, it is sufficient to put in the softening of the force. We start with a spherical cloud of randomly distributed stars, with some random initial velocities. Note: you find a random number generator program in the “Numerical Recipes” by Press et al. Let this stellar cluster evolve. Try out different numbers of stars, different initial conditions (especially the velocities, which determine the initail kinetic energy), time step widths, and softening parameters. What general behaviour can you see?

You will have noticed that there are a number of stars which escape from the cluster. Hénon (1969, Astron.Astrophys. **2**, 151) has computed the rate of stars escaping from the system. With our program we can measure the escape rate ....

4.5 Collision with a double star

First, make a double star, with two stars of equal mass, orbiting each other in circular orbits, say in the (x, y)-plane. Next, place another star – same mass – at a large distance from the double star, either in the orbital plane, or perpendicular to it. Let this star have some initial velocity, and positioned in a way that it flies towards the binary, and if there were no interaction, it would hit the double star at some distance from the centre. See what happens!

Let us first consider if the ‘bullet’ star comes perpendicular to the orbital plane: In this collision there are three parameters: the initial velocity of the star, where it hits the double star system – the impact parameter, i.e. the distance from the centre, if it would fly in a straight line – and the phase of the double star at the time of the collision – which we can control simply by placing the bullet nearer or farther. Play with these parameters, and observe what happens to the binary star. Under which conditions the double star remains unperturbed, when does the orbit change, when does the star pair break up?

Next, what happens when the bullet flies in the orbital plane? And: what happens when the three masses are not equal?