Astrophysical Exercises: Orbital perturbations

Joachim Köppen Strasbourg 2011

1. Physical Background

The motion of planets is ruled by the gravitational attraction of the Sun, described by Newton's inverse-square law. However, among the members of our solar system, there are Jupiter and Saturn whose large masses makes their gravitational influence on other planets, asteroids, and interplanetary probes non-negligeable. The presence of these additional forces causes the orbits of planets to deviate from Keplerian orbit, one speaks of orbital perturbations. The planet Neptune was discovered from the perturbations it causes to the orbit of Uranus. Another striking effect is visible in the structure of Saturn's rings which show some gaps. Also, the asteroids are not evenly distributed on orbits between Mars and Jupiter. If the ratio of the period of the perturbing body – here: Jupiter – and of the perturbed object is a rational number close to 1 – such as 3:2 or 2:1 or 3:1 – then the perturbation occurs at regular intervals at the same orbital position and in the same sense, one speaks of a resonance. By this steady accumulation of perturbations, every minor bodies is pushed away from the resonant orbit, and this orbital regime is left empty.

With our computers we can easily and accurately compute the orbits, due to any forces. In this exercise, we shall write a simplified program that simulates the flight of a planet around a star under the influence of gravity, but we also allow for the presence of a second planet which perturbs the orbit of the first one. In our simulations, we can compare the perturbed trajectory directly with the unperturbed orbit, and thus explore the characteristics and the evolution of the perturbations.

2. The Equations

Let us consider the orbit of a planet about a star. Since the star is much more massive than the planet, we may neglect that the planet also acts on the star, and hence both bodies perform orbits about their centre of mass. It makes life much easier, if we neglect this, but we do not lose any significant aspect: It can be shown that the treatment of the proper two-body problem can be separated into the motion of the centre of mass and the relative motion of the two bodies...

The star of mass M will thus be placed in the origin of the coordinate system and it will be at rest. At a given instant of time t, the planet of mass m shall be at

position $\mathbf{r} = (x, y, z)$ and moving with velocity $\mathbf{v} = (v_x, v_y, v_z)$. The force experienced by the planet is (written component-wise, marked by the indices):

$$F_i = -\frac{GMm}{r^2}\frac{r_i}{r} \tag{1}$$

where $r = |\mathbf{r}| = \sqrt{x^2 + y^2 + z^2}$ and G is the gravitational constant. The first term is Newton's law of gravity for the total force, the second term is just the projection of the force onto the direction to the star.

Here, it is useful to make another simplifying assumption: with any central force law, the motion of the planet is restricted to a plane. Thus, we may solve the equations in two dimensions only, by placing the orbital plane in the (x, y) plane. This means that the computational effort is reduced by 30 percent, and the simulation will run faster!

The planet experiences an acceleration $\mathbf{a}$ due to this force:

$$a_i = F_i/m = -\frac{GM}{r^2}\frac{r_i}{r} \tag{2}$$

which changes the velocity and the position of the particle:

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

These are all the equations we need to solve for one (unperturbed) planet. This shall be our reference planet, and we shall indicate its variables with the index "ref".

Now let us add two more planets to this system: First, the perturbing planet shall be treated in the same way as before, and we shall designate its variables with the index d.

Finally, we describe the perturbed planet: its equation of motion will include both the forces from the Sun and from the perturbing planet: here written for the x-component only

$$F_x = -\frac{GMm}{r^2}\frac{x}{r} - \frac{Gm_pm}{d^2}\frac{x - x_p}{d} \tag{4}$$

with the distance r to the Sun, as before, and the distance to the perturber $d = \sqrt{(x - x_p)^2 + (y - y_p)^2}$. The resulting acceleration (again only the x component:

$$a_x = F_x/m = -\frac{GM}{r^2}\frac{r_i}{r} - \frac{Gm_p}{d^2}\frac{x - x_p}{d} \tag{5}$$

We shall launch the perturbed planet together with the unperturbed one, so their initial positions and velocities will be identical.

Please note that we have neglected any perturbations of the perturber's orbit due to the gravitational influence by our planet ... so our model pertains to a small asteroid being influenced by Jupiter. It may not apply to the perturbations between Uranus and Neptune. We have made this approximation so that we can study the

simple perturbations by one body on another one. We have to keep this in mind along with the other assumption, whenever we apply the program and interpret results!

2.1 Normalized units

The first thing we notice in the above equations is that the planet's mass m completely vanishes from the final expressions for that planet. Also, that GM always appear as this product. This allows us to formulate the problem in normalized or adapted units: if we used m, kg, and s as units, we would get numerically large numbers for the astronomical dimension, masses, and times. This may cause numerical difficulties ... Instead it is better to have the numerical values of all the variables close to unity. If we chose such a unit of mass that $GM = 1$ and as unit of length that the initial distance planet-star becomes $r = 1$, and as unit of time that the speed for a circular orbit at radius 1 also becomes $v_c(r = 1) = 1$, we have a much better numerical situation. As the circumference of a circle of unity radius is 2π, this will be the value of the circular period.

Then, we recognize that our problem has only one parameter: the mass ratio of perturber and the Sun. Evidently if the perturber's mass is sufficiently small, the perturbations will be negligeable.

For our three bodies we have to specify their initial conditions, which are the position and velocity at time 0.0. If we place reference and perturbed planets on the same circular orbit, we might use these conditions

- $x_{\rm ref} = x = -1$ and $y_{\rm ref} = v_x = 0$
- $v_{x\rm ref} = y = 0$ and $v_{y\rm ref} = v_y = 1$

Let us assume that the perturbing planet is also on a circular orbit but with distance r_d from the Sun. Then we have

- $x_d = -r_d$ and $y_d = 0$
- $v_{xd} = 0$ and $v_{yd} = 1/\sqrt{r_d}$

There is another benefit: the general problem of a small body around a very massive one is reduced to this simple view. Irrespective of the real masses and the real dimensions, we get a circular orbit, if the small body has circular speed as its initial speed. This applies to any system: the circular speed for Earth satellites at low alitude is about 7.9 km/s, the circular speed around the Sun is 30 km/s at the place of the Earth, and about 5.5 km/s at Neptune's distance, and about 200 km/s for the Sun in the Milky Way Galaxy.

2.2 How to Compute it

At a given time t we compute the force, and thus the acceleration. 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. Thus the x-component of the velocity of the planet will change during this time step:

$$v_x(t + \Delta t) = v_x(t) + a_x \Delta t \tag{6}$$

Of course, during this time interval, the acceleration changes, as the planet flies to another position. 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 planet's new position 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 x-position is

$$x(t + \Delta t) = x(t) + v_x(t + \Delta t)\Delta t \tag{7}$$

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.....

The method described here must be applied to all three planets at the same time. In the program, one treats one body after the other, but this means that we must make sure that if one equation needs the positions of another planet, these must be the ones corresponding to the same instant of time!

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 single a planet which has circular velocity: 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, and recover Kepler's laws. 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? How does it depend on the length of the time step?

3. Things to Explore

3.1 Numerical Effects

Since orbital perturbations usually are relatively weak, the first thing you have do is to make sure that the numerical solution of the physics equations do not enter any undesired effects, such as lack of accuracy or unphysical behaviour.

Although our method (konown as leap frog) conserves angular momentum with machine accuracy, this does not apply to the energy! There is also a numerical precession of elliptical orbits: the major axis slowly turns around the central body. Both aspects can only be limited below some acceptable value, if the time step is below a certain small value. Thus, you have to find out by tests, which value to choose and which errors you could tolerate. As this decision depends on the type of the orbits you want to study, in particular their eccentricities, you need to do this testing for every configuration. In our exercise, we can use the orbit of the unperturbed planet to check for any deviations from the laws of physics.

You might be forced to take such a small time step that the overall computing time becomes too long to get results efficiently. Then you need to use a more accurate numerical method: For instance, we can modify our simple method by using a halfstep in time:

1. compute forces
2. get new velocity for half time step $v_x = v_x + a_x \Delta t/2$
3. get new positions at full step $x = x + v_x \Delta t$
4. compute forces at new position
5. get new velocity for next half time step $v_x = v_x + a_x \Delta t/2$

Since on an elliptical orbit the speed is maximal close to the central body, we could add some automatic ajustment of the time step, by demanding that the change in position and/or in speed during one step remains below a certain tolerance ϵ:

$$\epsilon > \Delta|v| \approx |a|\Delta t$$

gives

$$\Delta t \approx \frac{\epsilon}{|a|}$$

Or we could demand that the relative change $\Delta v/|v|$ in speed should remain below the tolerance... Similar arguments could be applied to the change in position ... There is no “best” solution!

Lastly, we could use a different and more sophisticated method: the well-known 4th order Runge-Kutta method would look promising, but it does not conserve angular

momentum with machine accuracy ... Fortunately, there is a symplectic version of RK4 - "symplectic" means that it obeys one conservation law. Written here for the x coordinate only:

- compute acceleration(x)
- $sv_1 = b_1\, a_x\, \Delta t$; $x_1 = c_1\, sv_1\, \Delta t$
- compute acceleration(x_1)
- $sv_2 = sv_1 + b_2\, a_x\, \Delta t$; $x_2 = x_1 + c_2\, sv_2\, \Delta t$
- compute acceleration(x_2)
- $sv_3 = sv_2 + b_3\, a_x\, \Delta t$; $x_3 = x_2 + c_3\, sv_3\, \Delta t$
- compute acceleration(x_3)
- $sv_4 = sv_3 + b_4\, a_x\, \Delta t$; $x_4 = x_3 + c_4\, sv_4\, \Delta t$
- result: $vx = sv_4$; $x = x_4$

With these coefficients

$$c_1 = c_4 = (2.0 + 2^{1/3} + 2^{-1/3})/6.0$$

$$c_2 = c_3 = (2.0 - 2^{1/3} - 2^{-1/3})/6.0$$

$$b_1 = b_4 = 0$$

$$b_2 = 1.0/(2.0 - 2^{1/3})$$

$$b_3 = 1.0/(1.0 - 2^{2/3})$$

In the applet http://astro.u-strasbg.fr/~koppen/OrbitPerturb/ the methods Leap Frog, Leap Frog 2nd order, Symplectic 4, as well as some others are implemented.

3.2 Two Planets on Circular Orbits

Obviously, the simplest case is to consider both perturbed and perturbing planets being on circular orbits. For convenience, we'll place the perturbed planet at distance $r = 1$ from the Sun. This gives essentially two free parameters: the mass m_p/M of the perturber, and its distance from the Sun r_p.

Evidently, the perturbations will increase with the perturber's mass. Choose this mass sufficiently large so that any effects on the other planet are noticeable ... but still obey $m_p \ll M$!

To show the perturbations more clearly, it is better to follow the evolution of the difference between the perturbed orbit and its artificially unperturbed reference twin. There are several possibilities to do this ... maybe you design and try out your own. In the applet we plot the difference in heliocentric angle as a function of the difference in heliocentric distance. One can also plot a zoomed view ... In these representations one notes that the perturbation occurs in radial direction quite rapidly and shows period increase and decrease in amplitude, reminiscent of the beats between two harmonic oscillators. Furthermore, the planet also oscillates in angular position, but also moves slowly away in angle from its unperturbed position.

If one places the perturber on an orbit with a different heliocentric distance, the amplitude and the shape of these periodic oscillations will be different. One can always use the perturber mass to make them larger or smaller.

Try to put the perturber in an orbit whose period is a multiple of our planet's period, where the factor is a rational number with small integers in numerator and denominator, such as 2:1 or 3:2 ... Then you will observe that the amplitude of the radial oscillations grows steadily until eventually it saturates at some value. The closer one gets to such a "resonance" the larger will the amplitudes be, but they still grow and fade like a beat. Also, the period of the beat increases ...

- What happens to its angular position?
- Which resonances can you detect with our simple program?
- What happens to the behaviour of the radial and angular amplitudes when you change the perturber's mass?
- What happens if you alter the initial heliocentric angle of the planets?
- Of course you should try out what happens when the perturber's mass is large, even if it might be too large for our simplified scheme to be still physically correct ... what happens to the perturbed planet?

3.3 Planets on Other Orbits

You could also place one or even both planets on elliptical orbits ...

3.4 Perturbation of the Orbital Elements

As long as perturbations do not upset the orbit completely, or the planets collide, one can characterize an orbit by matching it with a Keplerian orbit. In this way, one determines the length of the semimajor axis, the eccentricity, and other parameters. Then one can follow the evolution of the real trajectory by the evolution of these parameters.

Since we have full information about the position and velocity at any time, we may compute from these information the instantaneous orbital parameters.

Knowing the heliocentric distance r, the speed v, and the angle α between the velocity vector and the radius vector to the Sun, we compute (in our adapted units) the angular momentum (per unit planetary mass)

$$L = rv \sin\alpha = rv \sin(\arccos \frac{\mathbf{r}.\mathbf{v}}{rv})$$

and the (total) energy

$$E = \frac{v^2}{2} - \frac{1}{r}$$

Then we get the length of the semimajor axis

$$a = -\frac{1}{2E}$$

the orbital period (in units of the circular orbit at $r = 1$)

$$T = a^{3/2}$$

and the eccentricity

$$\epsilon = \sqrt{1 + 2L^2E}$$

If we want, we can also compute aphel and perihel

$$r_{a,p} = a(1 \pm \epsilon)$$

With these parameters we can study whether the perturbations affect more the shape of the orbit (its eccentricity) than the length of the major axis, and hence the orbital period.

3.5 A more consistent approach

If we want to see what may really occur when the masses of the planets are no longer negligeable in comparison to the Sun, we have to solve the equations of motion for all bodies and containing all mutual forces. From the information in the first sections you can formulate the equations easily yourself.

But several complications exist now:

— the Sun is no longer at the origin of the coordinate system, but the centre-of-mass of the system

— instead it orbits the centre-of-mass. Its motion is determined by the planets, i.e. the perturber and the perturbed planet. It is now more difficult to define the orbit of an unperturbed reference planet, because its orbit will be influenced by the motion of the Sun ...

— the initial positions and velocities of the Sun and the relevant planets need to be given in such a way that they conform with the condition that the centre-of-mass shall be at rest in the origin. This requirement along with the masses and the intended initial heliocentric distances gives relations from which we can work out the initial positions and velocities for each body.

— how one can display the results in a sensible way might also be a bit more difficult: shall we refer to the Sun or to the centre-of-mass?

4. 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.
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.
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. If you have a few variables that switch on these print-outs. Then, you can always check every part of the program, even after it has been considered finished. If later you 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 warning messages if something goes abnormal, e.g. if the automatic stepwidth goes below or above

specified values. This becomes 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, especially when results look reasonable ... when you get weird results, negative temperatures, NaN, orbital catastropies, then you should be happy: only then you really know that there in an error in the program!
7. When you make tests, and later run 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".