Orbits & Simulation

Part 6

Symplectic Integrators

In part 3 of these notes we described how the total energy in a gravitational system should (barring collisions) remain unchanged. A good integrator should maintain the total energy, being the sum of Potential and Kinetic Energy.

There is another way of looking at it – let us make the total energy a determining factor rather that simply using it as a measure of quality in the integrator.

Those who are happy to steep themselves in Hamiltonian Systems and Symplectic Manofolds are invited to look up these topics on the internet or study the papers listed in the bibliography by Yoshida, Chin and Chen.

The basic thrust of the story is that a dynamic system (like a gravitational system) that has a function whose value must preserved (like energy) and the function involves a couple of values that are related (like kintetic energy and potential energy) can be represented as a Hamiltonian function. There is a single set of solutions to those equations (called a symplectic manifold).

The learned mathematicians show that integrators (like we have used so far) always create a growing amount of energy error – i.e. they stray from the single set of solutions. This is due to the imperfections introduced by inexact estimates of position, velocity, acceleration as well as large time steps and rounding errors.

They show that some modifications to the methods can be introduced that maintain the energy and produce better results with larger timesteps (allowing for faster, longer simulations). Sympletic versions of many integrators are described in the literature but here I want to look at the Yoshida scheme for Leapfrog.

Yoshida Integrators

Yoshida (1990) shows that the energy level of the Leapfrog method can be maintained by splitting each time step into a series of short steps – some of them positive (i.e. forward in time) and some of them negative. Overall, the forward movements are greater than the backward movements by one time step but the steps are carefully selected so that the energy errors from the forward and backward mini-steps act to cancel each out out.

The maths in his paper are totally beyond me but the procedure is simply to execute the Leapfrog multiple times each with the normal time step multiplied by a factor. Implementation is described in easy steps by Hut and Makino (2003-2007).

For a 4th order method, execute 3 Leapfrogs with the time step multiplied by:

1. 1.351207191959657
2. -1.702414383919315
3. 1.351207191959657

For a 6th order integrator, execute the Leapfrog seven times with the time step multiplied by:

1. 0.784513610477560e0
2. 0.235573213359357e0
3. -1.17767998417887e0
4. 1.315186320683910e0
5. -1.17767998417887e0
6. 0.235573213359357e0
7. 0.784513610477560e0

For an 8th order method, execute the Leapfrog 15 times with the time step multiplied by:

1. 0.104242620869991e1
2. 0.182020630970714e1
3. 0.157739928123617e0
4. 0.244002732616735e1
5. -0.716989419708120e-2
6. -0.244699182370524e1
7. -0.161582374150097e1
8. -0.17808286265894516e1
9. -0.161582374150097e1
10. -0.244699182370524e1
11. -0.716989419708120e-2
12. 0.244002732616735e1
13. 0.157739928123617e0
14. 0.182020630970714e1
15. 0.104242620869991e1

Yoshida shows that there are several different sets of time-step-multipliers you can choose from, those shown above are the first in his set of options.

The big advantage of this method is that it is very easy to program and it seems to work very well.

Results

With 1 day time step over 100 years:

- 4th order, error 2×10^{-9} with visible precession error
- 6th order, error 5×10^{-13} no visible precession
- 8th order, error 4×10^{-12} no visible precession

With 0.1 day time step over 100 years:

- 4th order, error 6×10^{-14} with slight precession error visible
- 6th order, error 1×10^{-13} no visible precession
- 8th order, error 2×10^{-13} no visible precession

It looks like there is little advantage in using the 8th order method compared to the 6th order method and it is probably limited by rounding errors. The 6th Order method, with 0.1 day time step will run the Solar System at about 25 years per second on my i5 processor while maintaining an energy error of the order 10^{-13} for thousands of years. I have adopted Yoshida 6th order as my method of choice.

The integrator is written in Java and running on a 64bit runtime. It is much too fast to demonstrate with a gif image, but a there is a Javascript version of it running here that is reasonably fast.

To be continued…