http://demonstrations.wolfram.com/TwoStepAndFourStepAdamsPredictorCorrectorMethod/HTMLImages/index.en/11.gif(predictor step: two-step Adams–Bashforth)

http://demonstrations.wolfram.com/TwoStepAndFourStepAdamsPredictorCorrectorMethod/HTMLImages/index.en/12.gif(corrector step: two-step Adams–Moulton)

The four-step Adams predictor-corrector method uses the four-step Adams–Bashforth and Adams-Moulton methods together:

http://demonstrations.wolfram.com/TwoStepAndFourStepAdamsPredictorCorrectorMethod/HTMLImages/index.en/13.gif(predictor step)   
http://demonstrations.wolfram.com/TwoStepAndFourStepAdamsPredictorCorrectorMethod/HTMLImages/index.en/14.gif(corrector step)

Several years ago I thought of writing a solar system simulator.

Newton’s law of gravity is simple to integrate, so this should be a simple program.

Just set an initial position for all the planets, then run the integrate loop.

The main issue I encountered was that I could not find the initial positions of the planets :)

I thought a little about this problem, and realized that theoretically it should be possible to recover the 3d position from the aparent positions of the planets on the earth sky (which I could find in some astronomic calendars). But this was not an easy task. Basically to integrate the solar system I needed the precise location of the planets in some coordinate reference system, and their speed. To get this from the 2D projection on the sky should be possible: make some estimation for the 3d position and speed, then apply corrections to get the same 2d timeseries from the astronomical calendar. An interesting problem, but not quite the one I wanted to solve.

So in the end I did nothing more, and the idea was shelved.

Recently I googled for a solar system simulator, and after a couple of false starts I found <http://www.moshier.net/ssystem.html> which already does what I intended. From my point of view, the gem was that it contains a header file with exactly the information I was missing (the 3d positions and speed for all planets at a specific moment in time). As it turns out, there were a lot more things that I was missing, but more on this later.

From there I recovered the following story.

JPL needed to simulate the solar system for their missions and to provide that data to other projects.

So they made a detailed simulation (I was barely born around that time), and then published the resulted efemerides tables for all planets for a couple of kyears.

The Moshier guy wanted to reproduce the same data, 20 years later.

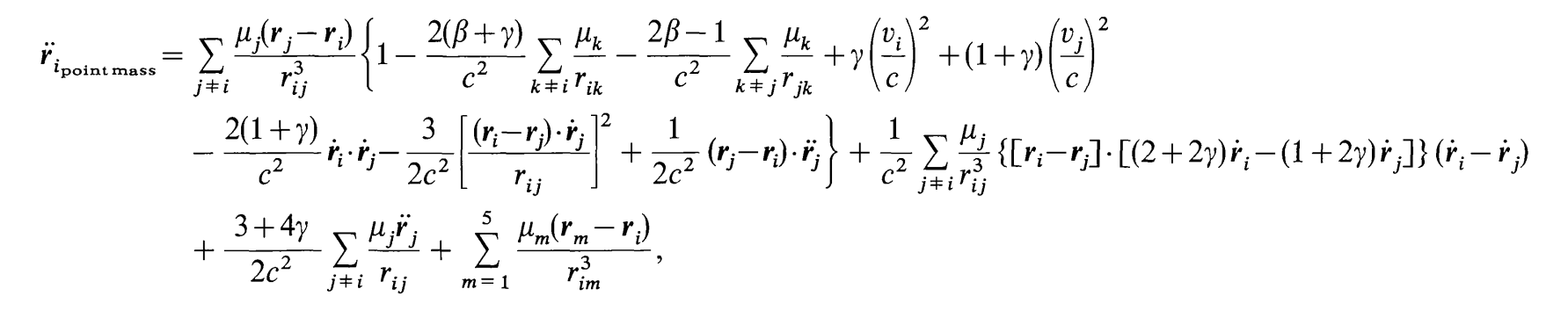
So I took a look at his code, which contained some stuff I did not expect.

There were 2 things that stand out:

1. The newton law of gravitation is not enough for a realistic simulation; relativistic corrections need to be applied.
2. The integrator used needs to be quite precise. A basic Runge-Kutta is not enough.

# 2-Body gravitation law

The equation used by the Jpl simulation has the following form:



I tried to find out what it means, but so far the results have been inconclusive.

It seems to be the PPN (post newton) approximation, with beta and gamma set to 1 (for general relativity).

# Integration

The jpl simulation uses a predictor-corrector integrator or order up to 19 (Adams-Bashfort-Moulton).

I found an easy way to generate the coefficients (solving an linear eq system), but more probably I can just take precomputed coefficients.

Still not sure what is the advantage of a 4 order predictor corrector (8 terms total) vs a RK of similar order.

The plot thickens when reading about N-body simulations I found that the modern implementations use a Hermite implementation, which apparently is faster and easier to implement. To check the stability of the integration the total energy of the system is used (i.e it must remain constant).

I found some details about the 4th order Hermite, and about 6 and 8th order which should be more precise.

Because some formulations of the Hermite interpolation are time symetric, it is possible to apply the corrector step several time to obtain better results. This is called the P(EC)n method.

In retrospect, I do not understand how this sophisticated integration methods work. Basically I am not able currently to deduce them on my own from what I know about how they work...

DE 102: a numerically integrated ephemeris of the Moon and planets spanning forty-four centuries - 1982 – Newhall JPL