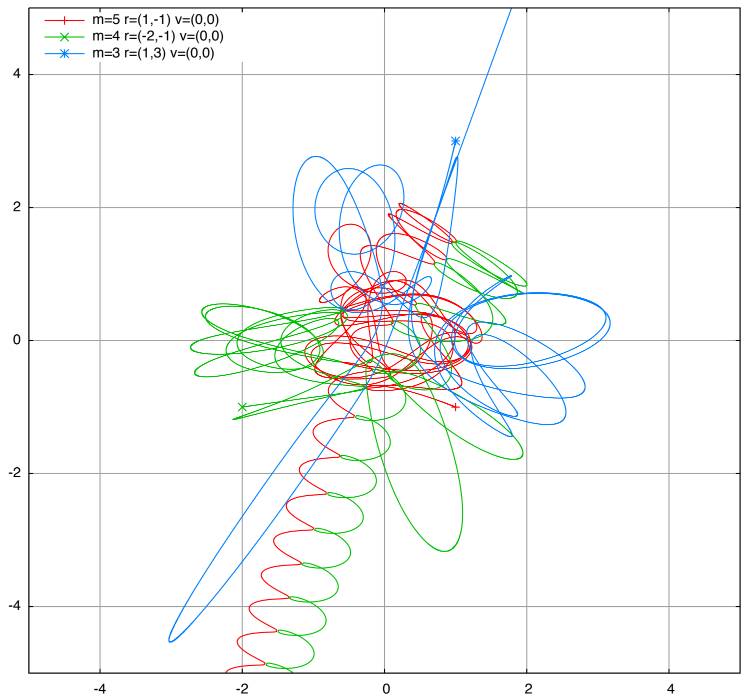
Astrophysical Simulations

Project Assignment

Planar 3-body problem with close encounters

September 2014

Peter Camps
peter.camps@ugent.be



1 Assignment

These are the project's key objectives:

1. Write a C++ program to accurately simulate the time evolution of planar 3-body systems that involve close encounters.
2. Implement Runge-Kutta-4 integration, variable time steps, and regularisation.
3. Study energy conservation and accuracy of the calculated orbits.
4. Solve Burrau's problem (see figure) and other 3-body systems of your choice.
5. Report on your results and experiences in a formal slide show and live presentation.

The following sections provide a more detailed list of the tasks to be accomplished.

2 Mathematics

Basic equations

Consider N point-like bodies $i = 1 \dots N$ with masses m_i gravitating about each other. Denoting the respective position vectors as $\boldsymbol{r}_i(t)$ the equations of motion can be written as

$$\ddot{\boldsymbol{r}}_i = - \sum_{j=1, j\neq i}^{N} \frac{G m_j (\boldsymbol{r}_i - \boldsymbol{r}_j)}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3}$$

where G is the gravitational constant. The total energy of the system is given by

$$\mathcal{E} = \sum_{i=1}^{N} \frac{1}{2} m_i \, |\dot{\boldsymbol{r}}|^2 - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{G m_i m_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}$$

For the purposes of this project, $N = 3$ and the initial conditions $\boldsymbol{r}_i(0)$ and $\dot{\boldsymbol{r}}_i(0)$ are such that the motion is restricted to the xy-plane.

Regularization

When the separation $|\boldsymbol{r}_i - \boldsymbol{r}_j|$ between two bodies i and j becomes very small, the corresponding term in the above equations grows very large, substantially affecting the accuracy of the numerical integration. Decreasing the time step helps, but is not sufficient to handle extreme close encounters. The solution is to transform the problem from physical to regularized coordinates, which are designed so that the singularity vanishes from the equations.

In physical coordinates, the two close encounter bodies i and j can be characterized by their relative position vector

$$\boldsymbol{r} = \boldsymbol{r}_i - \boldsymbol{r}_j$$

and their center of mass

$$\boldsymbol{r}_{\mathrm{C}} = \frac{m_i \boldsymbol{r}_i + m_j \boldsymbol{r}_j}{m_i + m_j} \, .$$

With the definition

$$\boldsymbol{F}_{i,j} = - \sum_{k=1; k\neq i,j}^{N} \frac{G m_k (\boldsymbol{r}_i - \boldsymbol{r}_k)}{|\boldsymbol{r}_i - \boldsymbol{r}_k|^3}$$

the tidal acceleration caused by the other bodies on the close encounter pair is given by

$$\boldsymbol{F} = \boldsymbol{F}_{i,j} - \boldsymbol{F}_{j,i}$$

and the acceleration on the center of mass due to the other bodies becomes

$$\ddot{\boldsymbol{r}}_{\mathrm{C}} = \frac{m_i \boldsymbol{F}_{i,j} + m_j \boldsymbol{F}_{j,i}}{m_i + m_j} \, .$$

The regularized time coordinate τ is defined through

$$\frac{\mathrm{d}}{\mathrm{d}\tau} = |\boldsymbol{r}| \, \frac{\mathrm{d}}{\mathrm{d}t} \, .$$

For planar problems, the transformation of the relative position vector $\boldsymbol{r} = (x, y)$ and its time derivative $\boldsymbol{v} = \dot{\boldsymbol{r}}$ to regularized coordinates $\boldsymbol{u} = (u_1, u_2)$ and the corresponding time derivative $\boldsymbol{w} = \mathrm{d}\boldsymbol{u}/d\tau$ can be written as

$$
\begin{aligned}
\boldsymbol{u} &= \left(\operatorname{sign}(y)\sqrt{\frac{|\mathbf{r}| + x}{2}}\,,\ \sqrt{\frac{|\mathbf{r}| - x}{2}} \right) \\
\boldsymbol{w} &= \frac{1}{2}\mathcal{L}^{\mathrm{T}}(\boldsymbol{u})\boldsymbol{v}
\end{aligned}
$$

and the inverse transformation as

$$
\begin{aligned}
\boldsymbol{r} &= \mathcal{L}(\boldsymbol{u})\boldsymbol{u} \\
\boldsymbol{v} &= \frac{2}{\boldsymbol{u}^2}\mathcal{L}(\boldsymbol{u})\boldsymbol{w}
\end{aligned}
$$

with the linear transform

$$
\mathcal{L}(\boldsymbol{u}) = \begin{pmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{pmatrix}
$$

and the sign function

$$
\operatorname{sign}(y) = \begin{cases} -1 & y < 0 \\ \ \ 1 & y \geq 0 \end{cases}.
$$

Using an accent to denote differentiation with respect to regularized time τ, the relative equation of motion for the close encounter pair i and j transforms to

$$
\begin{aligned}
t' &= \boldsymbol{u}^2 \\
\boldsymbol{u}' &= \boldsymbol{w} \\
\boldsymbol{w}' &= \frac{1}{2}\left[\mathcal{E}\boldsymbol{u} + \boldsymbol{u}^2\mathcal{L}^{\mathrm{T}}(\boldsymbol{u})\boldsymbol{F}\right] \\
\mathcal{E}' &= 2\boldsymbol{w}\cdot\mathcal{L}^{\mathrm{T}}(\boldsymbol{u})\boldsymbol{F}
\end{aligned}
$$

where the energy is given by

$$
\mathcal{E} = \frac{2\boldsymbol{w}^2 - G(m_i + m_j)}{\boldsymbol{u}^2}.
$$

3 Guidelines

Structure of the code

There are many ways to structure your code (many good ones and plenty more bad ones...). Here are some hints:

- There is no obligation to use C++ classes for this project. However it is recommended to create at least a class to represent a two-dimensional vector with x, y components and overloaded operators for basic vector arithmetic (addition, multiplication with a scalar, norm...). Your code will be lot easier to read.
- You don't need pointers for this project. If you need a variable-length array, use the `std::vector<>` class template rather than plain C++ arrays.
- Make your source code easy to read for yourself and for others: use vertical and horizontal white spacing in a consistent manner, properly indent function bodies and compound statements, provide meaningful variable and function names, and add *concise* comments to explain what the code is doing and why.

One step at a time

When tackling this project, take one step at a time. For example, use the steps outlined in the section below as "milestones". Compile often, and thouroughly test your code after each milestone. Make a backup after each milestone so you can revert to a working version if things go awry.

If you try to complete the assignment in one go, you will surely fail, and it will take a miracle to locate the problems in your code.

4 Implementation steps

Program

1. Write a program that solves the planar two-body system with the fourth-order Runge-Kutta integrator, using a fixed time step, without regularisation. Provide proper output and visualization tools to show orbits and track the evolution of the relative energy error. Verify the results for various initial conditions.

2. Extend your program and visualization tools to handle three bodies, still using fixed time steps, without regularisation. Verify the results for various initial conditions. Test the limits of the program: can you solve Burrau's problem?

3. Allow the time step h to vary (for all three bodies simultaneously) in function of the smallest separation $d_{\min}$ between the bodies, using a simple scheme such as $h \propto d_{\min}$. Study the effect on performance and accuracy of the results. Can you solve Burrau's problem?

4. At each time step, determine the two bodies with the smallest separation. If the separation is smaller than some threshold, integrate the close encounter two-body system using its relative position and center of mass, still in physical coordinates, and switch back to normal mode when the seperation is again large enough. While this shouldn't affect the results (except for numerical rounding differences), it allows you to test the "close encounter" mechanism and part of the coordinate conversions.

5. Add the transformations back and forth to regularized coordinates, and the Runge-Kutta integration of the regularized variables while in "close encounter" mode. Verify the results for various initial conditions. Monitor the evolution of the energy error, especially during close encounters. Adjust the time step and smallest separation cutoff for optimal results.

6. Solve Burrau's problem. Compare the calculated orbits to the reference solution on the first page of this text. Find other ways to convince yourself that your solution is correct (the reference solution might not be available for other problems).

7. Create a movie animating your solution of Burrau's problem. After all, this is why you were going through so much trouble!

8. Invent and solve other interesting three-body problems. Study the effect of the time step and other relevant simulation parameters.

Presentation

1. Prepare to discuss your results and experiences in a live presentation of *no more than 15 minutes*. Focus on the important issues. Show your results, your experiments and tests. Explain the motivations for the choices you made.

2. Create a formal slide show to serve as the backbone for your live presentation. Include a lot of visual material, such as your plots or diagrams. Movies are always very much appreciated! Don't show source code in the presentation, unless you have a *very* good reason to do so (for example, to point out an original way of solving some particular problem).

3. Be prepared to answer questions after your presentation. It is very important that you understand why your code operates the way it does. A nice result is not always a physically correct result. On the other hand, it is not necessarily a major disaster if your code does not produce the expected results. At the very least show that you explored various avenues and explain why and how you got stuck.

Deliverables

1. A week or so before the live presentation, send us your C++ code for the project. Provide any special instructions for using it. Make sure the code at least compiles!

2. During the live presentation bring your slide show and any related visual material. Preferably use your own laptop (don't worry about projecting, we'll just look at your screen). You're free to use any presentation software running on any operating system. If you want a backup in case your laptop fails at the wrong moment, bring a PDF copy of your presentation on a USB stick.

The deadline for handing in the C++ code and the time schedule for the live presentation will be announced via Minerva.