C++ Assessment, March 2018

This is the second C++ assessment for 2017 / 2018. It is worth 40% of the total mark available for this module. The submission date is in 30 April 2018 by 12:00.

Simulating N-body interaction

An N-body problem refers to modelling the interaction of N astronomical bodies in three dimensions through time. The assessment task is to write a general purpose N-body simulation programme and run it under three scenarios. Assume a Newtonian universe in which relativistic effects can be ignored.

Two different simulation methods must be implemented and their effectiveness, in terms of run-times and accuracy, compared. The methods are

1) A naive Euler discretization.
2) A Runge-Kutta discretization.

Details of these discretizations are given in the appendix. (Note: neither of these discretization methods are accurate over periods even as brief as hundreds of thousands of years.)

You must produce simulation results for the following scenarios

i) A system containing only Sun-Earth (based on the solar system).
ii) A system containing only Sun-Venus-Earth-Mars-Jupiter (based on the solar system).
iii) A system containing only Sun-Venus-Earth-Mars perturbed by a passing neutron star.

Initial data and evolution equations are given in the appendix. You need to be able to output a representation of the positions of the bodies in the system at intervals you can specify.

For scenarios:

i) You must comment on the effectiveness of the discretizations, and the stability of the system, if possible, over millions of years, in this simplest case.
ii) You must comment on the effectiveness of the discretizations, and the stability of the system.
iii) You should alter the initial velocity (speed and direction) of the neutron star. Draw conclusions about how destructive the passage of the neutron star is.

The application must be object oriented with a clear client interface and input validation. The code you submit must to be able to run without addition or modification with the DevCpp installation on the machines in the teaching area.

The report

Your report must include a clear description of your application and an assessment of its performance. You will be assessed on

a) The quality of your code in terms of readability, maintainability, clarity, sophistication, generalizability, and general presentation.
b) The degree to which your code has the required functionality.
c) Your assessment of the efficacy of your implementation.
d) The presentational quality of the assessment as a whole.

Group work is not permitted

The code you submit must be yours alone. Clear similarities in code, in the write-up or in results, will be taken as evidence of group work (for instance if identical or very similar screen shots as given, or if code is clearly shared.)

Code closely modelled on that from other sources is not permitted

Code to perform components of this exercise could be found in various places and might, for instance, be downloadable from the web. Use of code taken from, or altered from, such sources is not permitted. You must devise your code yourself.

Specifically *excluded* from this prohibition is code I have declared that you may use. In particular you may use the library code I have made available, without attribution.

Nick Webber

Appendix: an N-body system interacting under gravity

Let there be N (point) bodies, $\{B_i\}_{i=1,\dots,N}$ interacting under gravity. Each body $B_i = \{m_i, P_i, V_i\}$ has a mass m_i, a position $P_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, and a velocity $V_i = (u_i, v_i, w_i) \in \mathbb{R}^3$. Bodies interact under the influence of Newtonian gravity. They do not spin, and experience no tidal forces. They behave like point masses.

The equations of motion under Newtonian gravity are

$$\frac{dP_i(t)}{dt} = V_i(t), \qquad i = 1, \dots, N,$$
$$\frac{dV_i(t)}{dt} = A_i(t), \qquad i = 1, \dots, N,$$

where $A_i(t)$ is acceleration due to gravity,

$$A_i(t) = \sum_{\substack{k=1,\dots,N,\\ k\neq i}} G\frac{m_k}{|P_k(t) - P_i(t)|^3}\big(P_k(t) - P_i(t)\big),$$

and (in MKS units) $G = 6.674 \times 10^{-11}$ $m^3kg^{-1}s^{-2}$ is the constant of gravitational attraction.
Here $P_k - P_i \in \mathbb{R}^3$ is a 3-vector, and $|P_k - P_i|$ is the ordinary Euclidian distance,

$$|P_k - P_i| = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2}.$$

Table 1 gives initial data. This is based on approximate data for the solar system.

Table 1. Initial data

	m_i (kg)	P_{i0} (m)	V_{i0} (m/s)
Sun	1.989×10^{30}	$(0,0,0)$	$(0,0,0)$
Venus	4.867×10^{24}	$(-1.082 \times 10^{11}, 0, 6.411 \times 10^{9})$	$(0, -3.502 \times 10^{4}, 0)$
Earth	5.972×10^{24}	$(1.496 \times 10^{11}, 0, 0)$	$(0, 2.978 \times 10^{4}, 0)$
Mars	6.417×10^{23}	$(0, 2.279 \times 10^{11}, 7.359 \times 10^{9})$	$\big(-2.401 \times 10^{4}, 0, 0\big)$
Jupiter	1.898×10^{27}	$(0, -7.785 \times 10^{11}, -1.77 \times 10^{10})$	$(1.307 \times 10^{4}, 0, 0)$
Neutron star	4.0×10^{30}	$(0, 0, 1.0 \times 10^{15})$	$(1.2 \times 10^{2}, 0, -1.0 \times 10^{5})$

If using MKS units you may need to take $\Delta t = \frac{3600\times24\times365.25}{100{,}000}$, one hundred thousandth of a year (in seconds), in the Euler discretization. In the Runge-Kutta discretization a larger time step might be possible.

It may be convenient to use one year as the unit of time, instead of seconds, in which case $\Delta t = \frac{1}{100{,}000}$, and the astronomical units, $AU = 1.495978707 \times 10^{11}$ m, as the unit of length, instead of the metre. (One AU is roughly the average distance between the Earth and the Sun.) In AU-kg-year units the gravitational constant is $G = 1.9853 \times 10^{-29}$ $AU^3\,kg^{-1}\,yr^{-2}$.

Appendix: the Euler and Runge-Kutta discretizations

Consider a system of equations of the form

$$\frac{dy_i}{dt} = f_i(t, y_1, \dots, y_m), \qquad i = 1, \dots, m.$$

Set $\boldsymbol{y} = (y_1, \dots, y_m)'$, $\boldsymbol{f} = (f_1, \dots, f_m)'$, so in vector notation one can write

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t, \boldsymbol{y}).$$

In our case $m = 2N$, $y_i \in \mathbb{R}^3$, and

$$y_i = \begin{cases} P_i, & i = 1, \dots, N, \\ V_i, & i = N+1, \dots, 2N, \end{cases}$$

$$f_i(t, y_1, \dots, y_m) = f_i(P_1, \dots, P_N, V_1, \dots, V_N) = \begin{cases} V_i, & i = 1, \dots, N, \\ \sum_{\substack{k=1,\dots,N, \\ k \neq i-N}} G \frac{m_k}{|P_k - P_{i-N}|^3}(P_k - P_{i-N}), & i = N+1, \dots, 2N. \end{cases}$$

Note that the fact that $y_i \in \mathbb{R}^3$ causes no complications; one looks at each component of y_i individually.

Time evolves in ticks of length $h = \Delta t$ starting from time $t_0 = 0$, so that $t_j = t_{j-1} + \mathrm{h} = jh$. Write $\boldsymbol{y}_j$ and $\boldsymbol{f}_j$ for the values of $\boldsymbol{y}$ and $\boldsymbol{f}$ at time t_j.

A naive Euler discretization sets

$$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + h\boldsymbol{f}_j.$$

In our case, writing P_{ij} and V_{ij} for the position and velocity of body B_i at time t_j, the Euler discretization sets

$$P_{i,j+1} = P_{ij} + hV_{ij}, \qquad i = 1, \dots, N,$$

$$V_{i,j+1} = V_{ij} + h \sum_{\substack{k=1,\dots,N, \\ k \neq i}} A_{ik}, \quad i = 1, \dots, N,$$

where

$$A_{ik} = G \frac{m_k}{|P_{kj} - P_{ij}|^3}(P_{kj} - P_{ij}).$$

Note that the time step must be very small for this approximation to be useful.

A Runge-Kutta discretization is more accurate than the Euler discretization. The standard fourth order Runge-Kutta method sets

$$\begin{aligned}
\boldsymbol{d}_1 &= h\boldsymbol{f}(t_j, \boldsymbol{y}_j),\\
\boldsymbol{d}_2 &= h\boldsymbol{f}\left(t_j + \tfrac{1}{2}h, \boldsymbol{y}_j + \tfrac{1}{2}\boldsymbol{d}_1\right),\\
\boldsymbol{d}_3 &= h\boldsymbol{f}\left(t_j + \tfrac{1}{2}h, \boldsymbol{y}_j + \tfrac{1}{2}\boldsymbol{d}_2\right),\\
\boldsymbol{d}_4 &= h\boldsymbol{f}(t_j + h, \boldsymbol{y}_j + \boldsymbol{d}_3),
\end{aligned}$$

and finally sets

$$\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + \tfrac{1}{6}\boldsymbol{d}_1 + \tfrac{1}{3}\boldsymbol{d}_2 + \tfrac{1}{3}\boldsymbol{d}_3 + \tfrac{1}{6}\boldsymbol{d}_4.$$

(See, for example, Numerical Recipes.) One expects a larger time step can be used with a Runge-Kutta method to achieve the same accuracy as the Euler method.

In our case

$$d_{1,i} = \begin{cases} hV_{ij}, & i = 1, \dots, N,\\ h\sum\limits_{\substack{k=1,\dots,N,\\ k\neq i-N}} G\dfrac{m_k}{\left|P_{kj} - P_{i-N,j}\right|^3}\left(P_{kj} - P_{i-N,j}\right), & i = N+1, \dots, 2N, \end{cases}$$

$$d_{2,i} = \begin{cases} h\left(V_{ij} + \tfrac{1}{2}d_{1,i+N}\right), & i = 1, \dots, N,\\ h\sum\limits_{\substack{k=1,\dots,N,\\ k\neq i-N}} G\dfrac{m_k}{\left|P_k^2 - P_{i-N}^2\right|^3}\left(P_k^2 - P_{i-N}^2\right), & i = N+1, \dots, 2N, \end{cases} \quad \text{where } P_k^2 = P_{kj} + \tfrac{1}{2}d_{1,k},$$

$$d_{3,i} = \begin{cases} h\left(V_{ij} + \tfrac{1}{2}d_{2,i+N}\right), & i = 1, \dots, N,\\ h\sum\limits_{\substack{k=1,\dots,N,\\ k\neq i-N}} G\dfrac{m_k}{\left|P_k^3 - P_{i-N}^3\right|^3}\left(P_k^3 - P_{i-N}^3\right), & i = N+1, \dots, 2N, \end{cases} \quad \text{where } P_k^3 = P_{kj} + \tfrac{1}{2}d_{2,k},$$

$$d_{4,i} = \begin{cases} h\left(V_{ij} + d_{3,i+N}\right), & i = 1, \dots, N,\\ h\sum\limits_{\substack{k=1,\dots,N,\\ k\neq i-N}} G\dfrac{m_k}{\left|P_k^4 - P_{i-N}^4\right|^3}\left(P_k^4 - P_{i-N}^4\right), & i = N+1, \dots, 2N, \end{cases} \quad \text{where } P_k^4 = P_{kj} + d_{3,k},$$

and finally,

$$\begin{aligned}
P_{i,j+1} &= P_{ij} + \tfrac{1}{6}d_{1,i} + \tfrac{1}{3}d_{2,i} + \tfrac{1}{3}d_{2,i} + \tfrac{1}{6}d_{2,i}, && i = 1, \dots, N,\\
V_{i,j+1} &= V_{ij} + \tfrac{1}{6}d_{1,i+N} + \tfrac{1}{3}d_{2,i+N} + \tfrac{1}{3}d_{2,i+N} + \tfrac{1}{6}d_{2,i+N}, && i = 1, \dots, N.
\end{aligned}$$