Computer Physics Communications 185 (2014) 1339–1342

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

High precision framework for chaos many-body engine☆

I.V. Grossu [a,*], C. Besliu [a], D. Felea [b], Al. Jipa [a]

[a] University of Bucharest, Faculty of Physics, Bucharest-Magurele, P.O. Box MG 11, 077125, Romania

[b] Institute of Space Science, Laboratory of High Energy, Astrophysics and Advanced Technologies, Bucharest-Magurele, P.O. Box MG 23, 077125, Romania

ARTICLE INFO

Article history:
Received 2 April 2013
Received in revised form
16 November 2013
Accepted 10 December 2013
Available online 31 December 2013

Keywords:
Gravitation
Chaos
Many-body
C#
Butterfly effect
Intermittency

ABSTRACT

In this paper we present a C# 4.0 high precision framework for simulation of relativistic many-body systems. In order to benefit from the, previously developed, chaos analysis instruments, all new modules were integrated with Chaos Many-Body Engine (Grossu et al. 2010, 2013). As a direct application, we used 46 digits precision for analyzing the "Butterfly Effect" of the gravitational force in a specific relativistic nuclear collision toy-model.

Program summary

Program title: Chaos Many-Body Engine v04.1
Catalogue identifier: AEGH_v4_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEGH_v4_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Microsoft Public License (Ms-PL)
No. of lines in distributed program, including test data, etc.: 307938
No. of bytes in distributed program, including test data, etc.: 11953299
Distribution format: tar.gz
Programming language: Visual C# Express 2010.
Computer: PC.
Operating system: .Net Framework 4.0 running on MS Windows.
RAM: 100 Megabytes
Classification: 6.2, 6.5.
External routines: BigRational structure provided by Microsoft
Does the new version supersede the previous version?: yes
Nature of problem:
high precision simulation of relativistic many-body systems.
Solution method:
high precision calculations based on BigInteger .Net Framework 4.0 new feature.
Reasons for new version:
development of a high precision framework
Summary of revisions:

- high precision framework based on the new BigInteger .Net Framework 4.0 structure
- high precision relativistic many-body engine
- concrete application: using 46 digit precision for analyzing the gravitational Butterfly Effect in a specific relativistic nuclear collision toy-model

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail addresses: ioan.grossu@brahms.fizica.unibuc.ro, iv_grossu@yahoo.com (I.V. Grossu).

http://dx.doi.org/10.1016/j.cpc.2013.12.024

- CMBE Reactions Module Bug Correction: in the particular case of two identical particles head-on collision, reactions were not treated in earlier versions of CMBE.
- Chaos Analysis: implementation of a new measure "Average Y" for computing the average of any function loaded in this module.
- Chaos Analysis: implementation of the phase space distance between two many-body systems, as a function of time.
- Chaos Analysis: Implementation of a decimal version of the Chaos Analysis module.
- Chaos Analysis: Implementation of some usual relativistic formulas for facilitating processing of Monte Carlo log files (Analysis\Relativistic Formulas XLS).

Additional comments:

XCopy deployment strategy.

Running time:

Quadratic complexity, around 2 h for one C+C event, 50 Fm/c, on a dual core @ 2.0 GHz CPU

1. Introduction

In [1] we presented Chaos Many-Body Engine, a C# 2.0 application for chaos analysis of relativistic many-body systems. Although the double value type (8 octets, 15–16 digits precision) is sufficient for a wide class of simulations, it involves unacceptable limitations [2] in some specific problems encountered in physics (e.g. in the natural system of unities, the magnitude order of the gravitational constant is 10^{-45} MeV^{-2}). Considering the high instability in fluctuations and perturbations of non-linear systems, neglecting such small quantities might result in the important loss of information. In this context, we considered the opportunity of developing a high precision version of Chaos Many-Body Engine.

2. Program description

The "Chaos Many-Body Engine" library is designed as a flexible, general numerical solution for the simulation of three-dimensional relativistic many-body systems. In order to benefit from the latest technologies' advantages, the code was first migrated to .Net Framework 4.0 [3].

Math.dll. The *BigInteger* structure is a new feature in .Net Framework 4.0 [4]. For big rational numbers, Microsoft provides also an additional open source structure (*BigRational*), which is designed as a two *BigInteger-s* ratio.

Trying to use *BigRational*, one first problem comes from a growing precision effect (involved especially by multiplications), which leads to an unacceptable performance deprecation. Thus, starting from *BigRational*, we created the *BigDecimal* structure, its precision being limited (the *LimitPrecision* method) to the number of digits specified by the *Decimals* static property.

Another difficulty comes from the fact that .Net Framework 4.0 does not provide common mathematical functions for *BigRational-s*. In this context, we developed the static *BigMath* class, which implements a *halving interval programming technique* for computing the square root and the exponential functions for *BigDecimal* numbers.

The *BigVector* class is used for abstracting high precision three-dimensional vectors. We used operator overloading for implementing the sum, scalar product, and vectorial product operations.

AutomaticTests.dll. As Visual C# 2010 Express Edition does not include unit test capabilities, we created a simple framework for supporting automatic tests.

The *AutomaticTest* attribute represents the metadata used for identifying test methods. The *BaseAutomaticTest* class is designed as the base class for all automatic tests classes. The *AutomaticTestsEngine* contains a generic list of *BaseAutomaticTest* objects, and uses *reflection* [4] for dynamically calling all corresponding test methods (methods with the, previously mentioned, *AutomaticTest* attribute).

Detailed automatic tests were implemented for the most important classes of the Math.dll library (*BigDecimal, BigMath, BigVector, BigRelativity*), in order to minimize the impact of further changes or refactoring processes.

ParallelProgramming.dll. As working with high precision numbers might seriously affect the performance, an important attention was paid to both code optimization and algorithm parallelization. The *ParallelProgramming* library is a basic framework for executing methods on different execution threads [4,5]. The *IParallelObject* interface is used for defining the "contract" (*Start* and *Stop* methods) that a class must conform in order to be parallelizable. The *ParallelTask* class is used to create a new thread for executing the *Start* method of an *IParallelObject* instance, received as a constructor parameter (dependency injection). The *ParallelTaskList* class is based on a generic list of *ParallelTask* objects, and is used for calling multiple methods in parallel. Further analyses along parallelization techniques are currently in progress.

Engine.dll. The high precision many-body library was developed by analogy with the existing many-body engine [1,3]. Thus, the *BigParticle* class abstracts a material point and provides some basic scalar and vectorial properties (rest mass, movement mass, electric charge, velocity, force etc.). *BigNBody* is a generic list of *BigParticle* instances and provides also the main simulation algorithms. The *BigNBody.Next* method uses a second order Runge–Kutta algorithm [2] for computing the following system of equations (derived from the second Newton's law):

$$\begin{cases} \dfrac{d\vec{p_i}}{dt} = \displaystyle\sum_{i\langle\rangle j} \vec{F_{ij}} = \vec{F_i} \\ m_i \dfrac{d\vec{r_i}}{dt} = \vec{p_i} \\ m_i = \dfrac{m_{0i}}{\sqrt{1 - \left(\frac{v_i}{c}\right)^2}} \end{cases} \tag{1}$$

where p_i is the momentum, r_i the position, v_i the velocity, m_{0i} the rest mass, m_i the movement mass of the constituent i, t is the time, F_{ij} the bi-particle force, and c is the velocity of light in vacuum.

The *BigNBody.EnergyTest* property is used for implementing the energy conservation assessment [1,2]:

$$P_\epsilon = -\log_{10}\left|\frac{E(t) - E(t=0)}{E(t=0)}\right|. \tag{2}$$