Computer Physics Communications 183 (2012) 1055–1059

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Code C# for chaos analysis of relativistic many-body systems with reactions ☆

I.V. Grossu [a,*], C. Besliu [a], Al. Jipa [a], E. Stan [b], T. Esanu [a], D. Felea [b], C.C. Bordeianu [a]

[a] University of Bucharest, Faculty of Physics, Bucharest-Magurele, P.O. Box MG 11, 077125, Romania
[b] Institute of Space Sciences, Laboratory of Space Research, Bucharest-Magurele, P.O. Box MG 23, 077125, Romania

ARTICLE INFO

Article history:
Received 16 September 2010
Received in revised form 8 December 2011
Accepted 9 January 2012
Available online 13 January 2012

Keywords:
Chaos theory
Many-body
Nuclear relativistic collisions
Lyapunov exponent
Nuclear billiards
C#

ABSTRACT

In this work we present a reaction module for "Chaos Many-Body Engine" (Grossu et al., 2010 [1]). Following our goal of creating a customizable, object oriented code library, the list of all possible reactions, including the corresponding properties (particle types, probability, cross section, particle lifetime, etc.), could be supplied as parameter, using a specific XML input file. Inspired by the Poincaré section, we propose also the "Clusterization Map", as a new intuitive analysis method of many-body systems. For exemplification, we implemented a numerical toy-model for nuclear relativistic collisions at 4.5 A GeV/c (the SKM200 Collaboration). An encouraging agreement with experimental data was obtained for momentum, energy, rapidity, and angular π^- distributions.

Program summary

Program title: Chaos Many-Body Engine v02
Catalogue identifier: AEGH_v2_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEGH_v2_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 184 628
No. of bytes in distributed program, including test data, etc.: 7 905 425
Distribution format: tar.gz
Programming language: Visual C#.NET 2005
Computer: PC
Operating system: Net Framework 2.0 running on MS Windows
Has the code been vectorized or parallelized?: Each many-body system is simulated on a separate execution thread. One processor used for each many-body system.
RAM: 128 Megabytes
Classification: 6.2, 6.5
Catalogue identifier of previous version: AEGH_v1_0
Journal reference of previous version: Comput. Phys. Comm. 181 (2010) 1464
External routines: Net Framework 2.0 Library
Does the new version supersede the previous version?: Yes
Nature of problem: Chaos analysis of three-dimensional, relativistic many-body systems with reactions.
Solution method: Second order Runge–Kutta algorithm for simulating relativistic many-body systems with reactions. Object oriented solution, easy to reuse, extend and customize, in any development environment which accepts .Net assemblies or COM components. Treatment of two particles reactions and decays. For each particle, calculation of the time measured in the particle reference frame, according to the instantaneous velocity. Possibility to dynamically add particle properties (spin, isospin, etc.), and reactions/decays, using a specific XML input file. Basic support for Monte Carlo simulations. Implementation of: Lyapunov exponent, "fragmentation level", "average system radius", "virial coefficient", "clusterization map", and energy conservation precision test. As an example of use, we implemented a toy-model for nuclear relativistic collisions at 4.5 A GeV/c.
Reasons for new version: Following our goal of applying chaos theory to nuclear relativistic collisions at 4.5 A GeV/c, we developed a reaction module integrated with the Chaos Many-Body Engine.

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

* Corresponding author.
E-mail address: ioan.grossu@brahms.fizica.unibuc.ro (I.V. Grossu).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2012.01.009

Summary of revisions:

1. In the previous version, inheriting the Particle class was the only possibility of implementing more particle properties (spin, isospin, and so on). In the new version, particle properties can be dynamically added using a dictionary object.
2. The application was improved in order to calculate the time measured in the own reference frame of each particle.
3. We developed a reaction module for treating the following processes:
   - two particles reactions: $a + b \to c + d$,
   - decays: $a \to c + d$,
   - stimulated decays,
   - more complicated schemas, implemented as various combinations of previous reactions.
4. Following our goal of creating a flexible application, the reactions list, including the corresponding properties (cross sections, particles lifetime, etc.), could be supplied as parameter, using a specific XML configuration file.
5. The simulation output files were modified for systems with reactions, assuring also the backward compatibility.
6. We propose the "Clusterization Map" as a new investigation method of many-body systems.
7. The multi-dimensional Lyapunov Exponent was adapted in order to be used for systems with variable structure.
8. Basic support for Monte Carlo simulations was also added.

Additional comments: Windows forms application for testing the engine. Easy copy/paste based deployment method.
Running time: Quadratic complexity.

1. Introduction

In a previous work [1], we presented the "Chaos Many-Body Engine" code library for chaos analysis of relativistic many-body systems. Inspired by existing studies on Fermi nuclear systems [2–5], we tried to apply chaos theory to the more complex case of nuclear relativistic collisions at 4.5 A GeV/c. In this domain of energies it is possible to apply the many-body representation, as the nucleon de Broglie wavelength is less than the average internucleonic distance, and the nucleon mean free path is shorter than the target radius [6–10]. On the other hand, the two-nucleon reactions are a key point in understanding the processes involved in nuclear collisions. In this context we motivate our efforts of developing a C# [11] reaction module, integrated with "Chaos Many-Body Engine".

2. Two particles reactions and decays

The second version of "Chaos Many-Body Engine" library is designed as a general numerical solution for the simulation of three-dimensional relativistic many-body systems with reactions. We implemented: two particles reactions, decays, and stimulated decays.

***a + b → c + d* reactions.** For simplicity, we considered that only the momentum component oriented towards the direction between the two bodies is changing during the collision. Working in a Cartesian system in which this direction corresponds to abscise, by applying the relativistic energy and momentum conservation laws, written in the natural system of units, we obtained the following second degree equation [12–14]:

$$\left(\alpha^2 - 1\right)E_c^2 + 2\alpha\beta E_c + \beta^2 + p_{ya}^2 + p_{za}^2 + m_c^2 = 0 \tag{1}$$

with:

$$\alpha = \frac{E_a + E_b}{p_{xa} + p_{xb}} \tag{2}$$

$$\beta = \left[m_d^2 + p_{yb}^2 + p_{zb}^2 - (E_a + E_b)^2 + (p_{xa} + p_{xb})^2 - p_{ya}^2 - p_{zb}^2 - m_c^2\right]\left[2(p_{xa} + p_{xb})\right]^{-1} \tag{3}$$

where m_a, m_b, m_c, m_d are the rest masses, p_{xa}, p_{xb}, p_{xc}, p_{xd}, p_{ya}, p_{yb}, p_{yc}, p_{yd}, p_{za}, p_{zb}, p_{zc}, p_{zd} are the Cartesian components of momentum, and E_a, E_b, E_c, E_d are the particle energies.

The momentums of the two resultant particles are given by:

$$p_c = \pm\sqrt{E_c^2 - m_c^2} \tag{4}$$

$$\vec{p}_d = \vec{p}_a + \vec{p}_b - \vec{p}_c \tag{5}$$

Only the physical solutions of Eqs. (1)–(5) are chosen. We impose also two additional conditions:

- The two colliding particles must approach each other.
- The distance between the two particles must be less than a specific value, $r_{\max}$, calculated in agreement with the cross section σ:

$$r_{\max} = \sqrt{\frac{\sigma}{\pi}} \tag{6}$$

***a → c + d* decays.** For simplicity, we assumed that each decay takes place after the corresponding lifetime, measured in particle's own reference system. Thus, the program was improved in order to calculate each particle lifetime, according to the corresponding instantaneous velocity:

$$t_2 = t_1 + \frac{dt}{\sqrt{1 - \beta^2(t_1)}} \tag{7}$$

where t is the time, $\beta = v/c$, and dt is the integration interval.

In the frame of decaying particle, the two new particles are emitted on a random direction, with the following momentum:

$$p = p_c = p_d = \frac{1}{2m_a}\sqrt{\left[m_a^2 - (m_c + m_d)^2\right]\left[m_a^2 - (m_c - m_d)^2\right]} \tag{8}$$

The result in the simulation reference frame is obtained by applying a Lorenz transformation.

Stimulated decays are decays triggered by collisions. At least one participant must have at least one decay schema. This kind of

reactions is implemented by simply setting the particles lifetime values to zero on collision events.

It is important to notice that more complex schemas could be considered by employing various combinations of the previous described basic reactions. For example, for $a+b \to c+d+e$, one can use a virtual particle "f", with zero lifetime, which decays into "d" and "e". In this case, the engine will process the following sequence: $a+b \to c+f$; $f \to d+e$.

3. Program description

Our main goal was to create a highly configurable reaction module, integrated with "Chaos Many-Body Engine". Thus, for storing the list of all reaction schemas which applies to the many-body system of interest, we created the *DsReactions* typed dataset. The main advantages of using datasets [15] are related to XML serialization, and very easy filtering capabilities, based on SQL-like query strings.

The *DsReactions* dataset contains three datatables. The *DsReactions.Particle* table is used for defining all necessary particle types. It contains the unique identifier (*IdParticle*), and the most common particle properties (mass, charge, and lifetime). The *DsReactions.ParticleProperties* datatable is in a many-to one relation with *DsReactions.Particle*, and is used for dynamically defining particle properties (spin, isospin, color charge and so on). One new *Particle* class instance can be created, based on information stored in the *DsReactions.Particle* and *DsReactions.ParticleProperties* datatables, using the *NBody.NewParticle* method, which expects the particle type (*IdParticle*) as parameter.

The *DsReactions.Reactions* datatable is used for defining the list of all possible reactions for the specific system of interest. The *IdParticleIn01* and *IdParticleIn02* columns represent the unique identifiers of the input particles, while *IdParticleOut01* and *IdParticleOut02* are the reaction output identifiers. The *MaxDistance* column is directly related to the cross-section parameter (Eq. (6)). For avoiding also any unwanted effect related to infinity values near origin, we considered a minimum distance (the *MinDistance* column), under which the reaction is forbidden. For decays, this property has a different significance. It represents the initial distance between the two, new generated, particles. When more channels are possible, the reaction probability can also be specified. The same datatable is used for all reaction types. Thus, the decays are specified by setting *IdParticleIn02* = *null*, while, for stimulated decays, one will set both *IdParticleOut01*, and *IdParticleOut02* to *null*.

The list of all allowed reactions can be supplied/modified by simply creating/changing the XML file used as data source for the *DsReactions* dataset. The user should carefully define each reaction, in agreement with all specific conservation laws (charge, baryonic number and so on), and avoiding also circular references.

Based on a second order Runge–Kutta algorithm, the *Nbody.Next* method is used for "moving" the many-body system into its next status [1]. We improved this method for finding also all particles susceptible to participate to reactions. Thus, we fill one generic list of *Reaction* objects with all constituent pairs for which the mutual distance is less than the maximum cross-section radius, Eq. (6), and another one, with all particles for which the time, measured in the own reference system, is greater or equal with the corresponding lifetime. The two lists are used as input for the reactions engine. For optimization reasons, the reaction module is called with a frequency which can be specified as parameter. It is important to notice that the program will automatically increase this frequency in certain circumstances (the number of particles susceptible to participate to reactions changed, and/or, at least one reaction took place in the current iteration).

The *ReactionEngine* class contains the main processing routines. The *GetReactions* method assures that one constituent will participate to only one reaction. The decays are treated in priority. The *CheckReaction* method is used for finding all reactions which apply to the input data (the previous described generic lists), according to the information stored in the *DsReactions* dataset. The *GetReactionOutput* and *GetDecayOutput* methods select only the physical solutions which satisfy the energy and momentum conservation laws (Eqs. (1)–(5)). If there are more possible reaction schemas for the same input, one solution is randomly chosen, according to its probability (the *DsReactions.Reactions.Probability* column).

The *Nbody.ProcessReactions* function is used for applying the processing result to the *NBody* instance. It removes the old particles, and adds the new ones from/to the *mParticles* generic list. The *OnReactions* event is raised if at least one reaction took place. As the collisions are inelastic, the kinetic and potential energy variables are also adjusted after each reaction event, in order to avoid any impact on the energy conservation precision test [16]:

$$-\log_{10}\left|\frac{E(t)-E(t=0)}{E(t=0)}\right| \tag{9}$$

The structure of the simulation output files was changed according to the new necessities, trying to assure also the compatibility with the previous version. Thus, in the header file we added the creation time of each particle (the *Time* column), an integer (*Particle*) which uniquely identifies each particle instance, and the particle type (*Id*), which corresponds to the *IdParticle* column defined in the *DsReactions* dataset. As the structure of the system could change in time, the unique instance identifier field (*Particle*) was added also to the data file.

4. Chaos analysis of many-body systems

In [1] we implemented the multi-dimensional Lyapunov Exponent [17,18]:

$$L \overset{\text{def}}{=} \lim_{t\to\infty}\frac{1}{t}\ln\frac{d(t)}{d(0)} = \lim_{t\to\infty} L(t) \tag{10}$$

where $d(t)$ represents the phase space distance between the two systems:

$$d = \sqrt{\sum_{i=1}^{n}\left((\vec{r}_{i1}-\vec{r}_{i2})^2 + (\vec{p}_{i1}-\vec{p}_{i2})^2\right)} \tag{11}$$

where r_{i1}, p_{i1}, are the position, respectively the momentum, of the particle "i" from the first system, and r_{i2}, p_{i2} are the analogous coordinates of the second system.

As the structure of one many-body system with reactions could change in time, the previous distance cannot be applied any more. We propose instead a "global" approach:

$$d = \sqrt{\left(\sum_{i=1}^{n_1}\vec{r}_{i1} - \sum_{i=1}^{n_2}\vec{r}_{i2}\right)^2 + \left(\sum_{i=1}^{n_1}\vec{p}_{i1} - \sum_{i=1}^{n_2}\vec{p}_{i2}\right)^2} \tag{12}$$

The *BaseSystemData.GlobalLyapunovExponent* static method is implementing the Lyapunov Exponent based on this distance definition.

Inspired by the Poincaré section [18,19], we propose also a new intuitive investigation method. Thus, for one many-body constituent we considered the following set:

$$M_i = \left\{\left(x_i(t), y_i(t)\right) \;\middle|\; \frac{dp_{xi}}{dt}(t)=0 \vee \frac{dp_{yi}}{dt}(t)=0 \vee \frac{dp_{zi}}{dt}(t)=0\right\} \tag{13}$$

We defined the "Clusterization Map" as:

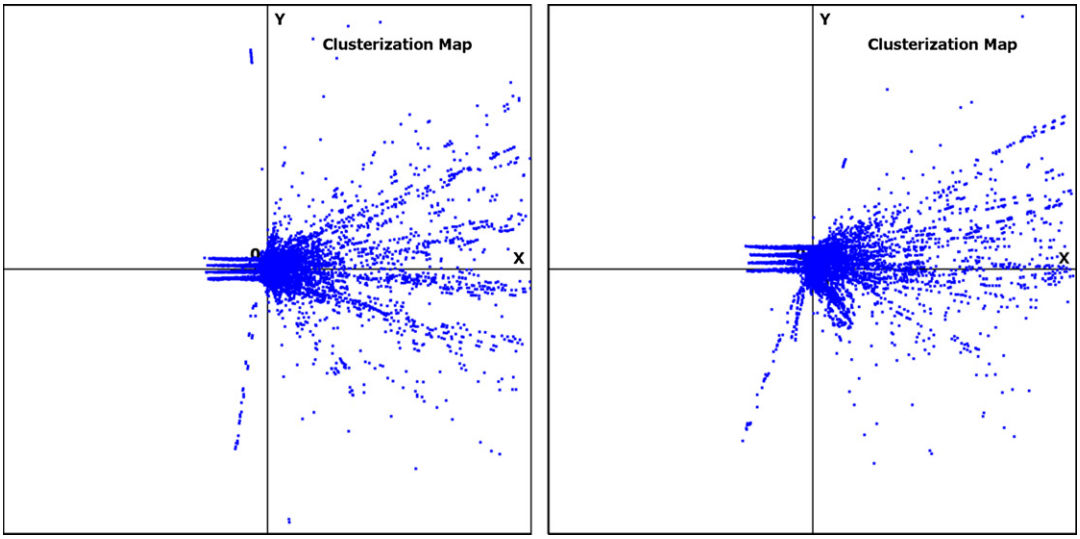

Fig. 1. Simulation of Cu + Cu at 4.5 A GeV/c nuclear collision. Left: Clusterization map for a central collision. Right: Clusterization map for a peripheral collision.

$$M = \bigcup_{i=1}^{n} M_i \tag{14}$$

Using this technique, one obtains a "bi-dimensional projection" of all clusters created inside the system. For a better precision, the *BaseSystemData.ClusterizationMap* static method is searching all points for which the signs of derivatives from Eq. (13) are changing. That way, the free particles are excluded from the set.

5. Application to the C–C nuclear collision at 4.5 A GeV/c

As an example of use, we implemented a simplified model for nuclear collisions at 4.5 A GeV/c (the SKM200 Collaboration [8, 20–22]). (See Fig. 1) The colliding nuclei are represented as two distinct sets of nucleons, initially placed in the vertices of a cubical network, with the edge calculated in agreement with the nucleus radius $r = r_0 A^{1/3}$. The target is initially at rest, while the incident momentum of the projectile constituents could be specified as parameter. For simplicity, the Lorenz contraction is ignored.

We employed a finite depth Yukawa potential well, together with a coulombian, and a short distance repulsive term [23]:

$$V(r_{ij}) = \begin{cases} -k\ln(r), \\ \quad r_{ij} \leqslant 1.134 \text{ Fm} \\ -V_0 \frac{e^{-\frac{r_{ij}}{a}}}{\frac{r_{ij}}{a}} + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} - k\ln(r), \\ \quad 1.134 \text{ Fm} < r_{ij} < 2.2 \text{ Fm} \\ -V_0 \frac{e^{-\frac{r_{ij}}{a}}}{\frac{r_{ij}}{a}} + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}, \\ \quad r_{ij} \geqslant 2.2 \text{ Fm} \end{cases} \tag{15}$$

where $V_0 = 35$ MeV is the depth and $a = 2$ Fm is the radius of the potential well, r_{ij} represents the distance between the two bodies, and q is the electric charge. We empirically chose $k = 200$.

The following interactions were considered [8,24–27]:

$p + n \to p + \Delta^0$	50%
$p + n \to \Delta^+ + n$	50%
$p + p \to \Delta^+ + p$	100%
$n + n \to \Delta^0 + n$	100%
$\Delta^0 \to p + \pi^-$	50%
$\Delta^0 \to n + \pi^0$	50%
$\Delta^+ \to p + \pi^0$	50%
$\Delta^+ \to n + \pi^+$	50%

Working at 10^{-3} Fm/c temporal resolution, we simulated 500 C + C events, with random collision parameters. The comparison with experimental data illustrates an encouraging result for momentum, energy, rapidity, and angular π^- distributions.

6. Conclusion

Starting from existing analysis of Fermi nuclear systems, one of our main goals was to apply chaos theory to relativistic nuclear collisions at 4.5 A GeV/c (the SKM200 collaboration). In this context, we developed a C# reaction module for "Chaos Many-Body Engine". An important attention was paid to the application flexibility and extensibility. Thus, we improved the code in order to accept any number of, dynamically defined, particle properties (spin, isospin, color charge and so on). The list of all particles and possible reactions could also be supplied as a simulation parameter using a specific XML input file.

As the structure of one many-body system with reactions could change in time, in the implementation of the multi-dimensional Lyapunov Exponent we considered a particular definition for the distance between two systems, Eq. (12). We propose also the "Clusterization Map", Eq. (14), as a new analysis method, which can bring intuitive information on the clusters created over the time in the system.

The previous described example of use (C + C collision) does not require significant resources to run (CPU 1.0 GHz, 128 M free RAM, .Net Framework 2.0 running on MS Windows XP or later). However, the resources could become a critical problem with the increasing of reactions and particles numbers (e.g. Cu + Cu or Au + Au collisions).

Using a simplified model for C + C nuclear collision at 4.5 A GeV/c, we obtained some encouraging results for momentum, energy, rapidity, and angular π^- distributions (Fig. 2). Further analyses along those lines are currently in progress. Thus, we intend to apply Chaos Many-Body Engine to other nuclear collisions (He + Li, Li + C, He + Cu, etc.), using also more complex interaction schemas.

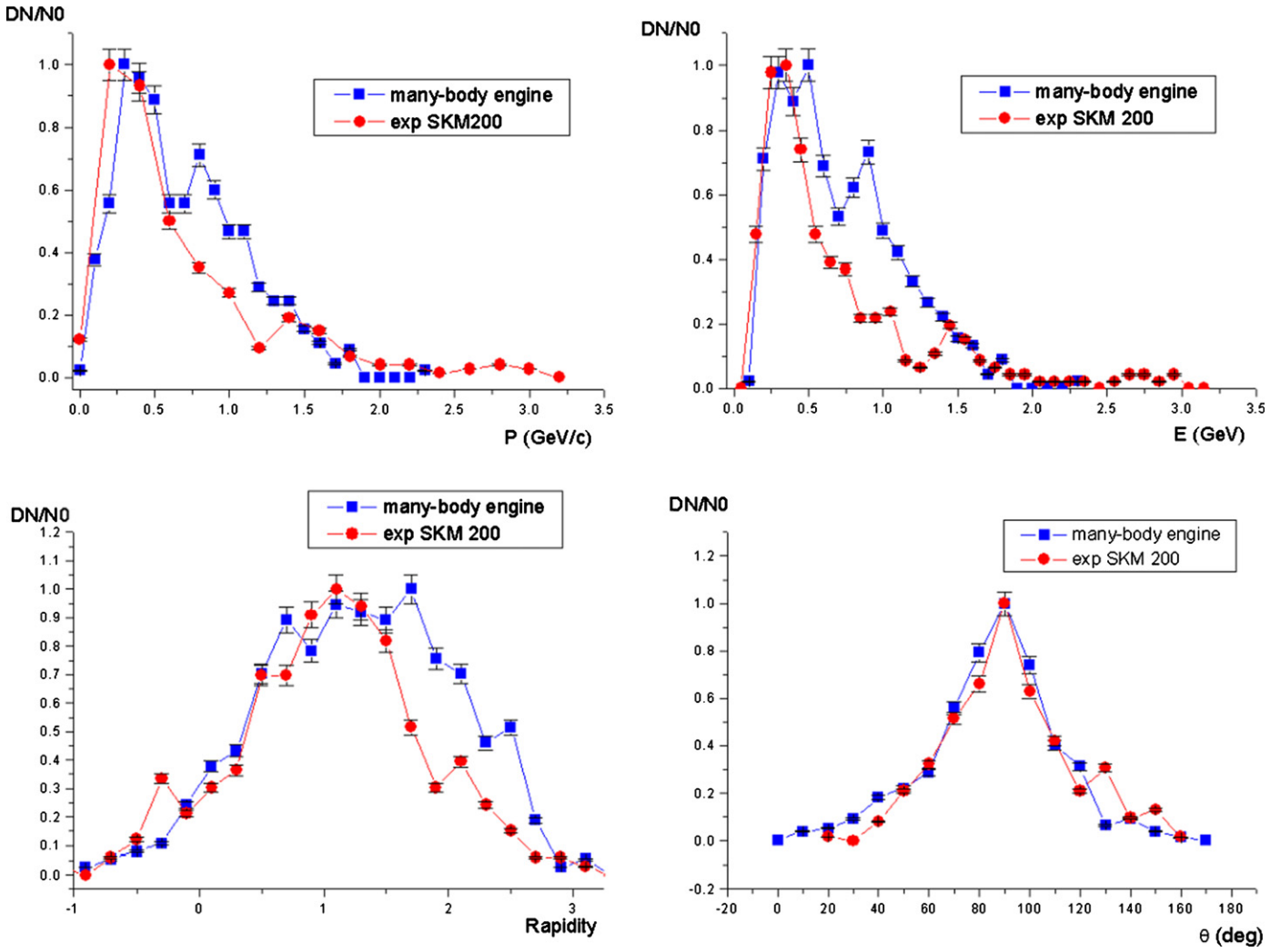

Fig. 2. Comparison between Chaos Many-Body Engine and experimental π^- distributions for C + C at 4.5 A GeV/c. Up left: Momentum distribution. Up right: Energy distribution. Down left: Rapidity distribution. Down Right: Angular distribution.

Acknowledgement

This study is supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number 34/05.10.2011 PN-II-ID-PCE.

References

[1] I.V. Grossu, C. Besliu, Al. Jipa, C.C. Bordeianu, D. Felea, E. Stan, T. Esanu, Comput. Phys. Comm. 181 (2010) 1464–1470.
[2] G.F. Burgio, F.M. Baldo, A. Rapisarda, P. Schuck, Phys. Rev. C 58 (1998) 2821–2830.
[3] D. Felea, PhD thesis, University of Bucharest, Physics Department, 2002.
[4] C.C. Bordeianu, D. Felea, C. Besliu, Al. Jipa, I.V. Grossu, Comput. Phys. Comm. 179 (2008) 199–201.
[5] C.C. Bordeianu, D. Felea, C. Besliu, Al. Jipa, I.V. Grossu, Commun. Nonlinear Sci. Numer. Simul. 16 (2011) 324–340.
[6] S. Nagamya, Prog. Part. Nucl. Phys. XV (1985) 363.
[7] R. Stock, Prog. Part. Nucl. Phys. XV (1985) 455.
[8] Al. Jipa, C. Besliu, Elemente de fizica nucleara relativista – Note de curs, Editura Universitatii din Bucuresti, Bucuresti, Romania, 2002.
[9] C. Besliu, Al. Jipa, Rev. Roum. Phys. 33 (1988) 409.
[10] Al. Jipa, PhD thesis, University of Bucharest, Physics Department, 1989.
[11] Christian Nagel, Bill Evjen, Jay Glynn, Morgan Skinner, Karli Watson, Professional C# 2008, Wiley, Indianapolis, IN, 2008.
[12] Rolf Hagedorn, Relativistic Kinematics—A Guide to the Kinematic Problems of High-Energy Physics, W.A. Benjamin, 1973.
[13] Eero Byckling, K. Kajantie-John, Particle Kinematics, Wiley & Sons, 1973.
[14] R. Ion-Mihai, O.G. Duliu, M. Penescu, et al., Fizica Nucleara: Culegere de Probleme, ALL, 1994.
[15] Glenn Johnson, Programming Microsoft ADO.NET 2.0 Applications: Advanced Topics, Microsoft Press, Washington, US, 2006.
[16] R.H. Landau, M.J. Paez, C.C. Bordeianu, Computational Physics: Problem Solving with Computers, Wiley-VCH-Verlag, Weinheim, 2007.
[17] M. Sandri, Numerical calculation of Lyapunov exponents, Math. J. 6 (1996) 78–84.
[18] C.C. Bordeianu, D. Felea, C. Besliu, Al. Jipa, I.V. Grossu, Comput. Phys. Comm. 178 (2008) 788–793.
[19] G.P. Williams, Chaos Theory Tamed, Joseph Henry Press, Washington DC, 1997.
[20] A.U. Abdurakhimov, et al., preprint JINR 13-10692, 1977.
[21] P. Rice-Evans, Spark, Streamer, Proportional and Drift Chambers, The Richelieu Press, London, 1974.
[22] C. Besliu, N. Ghiordanescu, M. Pentia, Studii si Cercetari de Fizica 29 (1977) 817.
[23] I.V. Grossu, PhD thesis, University of Bucharest, Physics Department, 2009.
[24] Al. Jipa, C. Beşliu, Participants in relativistic nuclear collisions, Nuovo Cimento A 106 (1993) 317–325, doi:10.1007/BF02771447.
[25] C. Beşliu, Al. Jipa, Method to evidence the contributions of the spectator and participant regions to the particle production in relativistic nuclear collisions, Romanian Rep. Phys. 55 (4) (2003) 420–432.
[26] T. Esanu, PhD thesis, University of Bucharest, Physics Department, 2010.
[27] E. Stan, PhD thesis, University of Bucharest, Physics Department, 2011.