CuBA — A CUDA IMPLEMENTATION OF BAMPS*

Nuno Cardoso, Pedro Bicudo, Ulrike Eilhauer

CFTP, Departamento de Física, Instituto Superior Técnico
Universidade Técnica de Lisboa, Av. Rovisco Pais, 1049-001 Lisbon, Portugal

Ioanni Bouras

Institut für Theoretische Physik, Johann Wolfgang Goethe-Universität
Max-von-Laue-Strasse 1, 60438, Frankfurt am Main, Germany

(Received September 3, 2012)

Using CUDA as programming language, we create a code named CuBA which is based on the CPU code "Boltzmann Approach for Many Parton Scattering (BAMPS)" developed in Frankfurt in order to study a system of many colliding particles resulting from heavy ion collisions. Furthermore, we benchmark our code with the Riemann Problem and compare the results with BAMPS. They demonstrate an improvement of the computational runtime, by one order of magnitude.

DOI:10.5506/APhysPolBSupp.5.1027
PACS numbers: 11.15.Ha, 12.38.Gc, 12.38.Mh

1. Introduction

Basing ourselves on the BAMPS code developed in Frankfurt by C. Greiner, Z. Xu *et al.*, we decided to study the interaction between the gluons of a gluon gas produced at the onset of Heavy Ion Collisions [1]. We use CUDA as programming language to create the code CuBA "The Boltzmann Approach for Many Parton Scattering written with CUDA" [2]. We expect to get an improvement of the computational runtime. In addition, both codes are benchmarked with the Riemann problem to compare the results of the two programs.

In this paper, we investigate the physical concepts behind this program, the CUDA language and finally the prior results obtained.

* Presented by U. Eilhauer at the Workshop "Excited QCD 2012", Peniche, Portugal, May 6–12, 2012.

2. Theory

We are specially interested in solving the Riemann problem in viscous matter using the relativistic Boltzmann equation which is as follows

$$\left(\frac{\delta}{\delta t}+\frac{p_1}{m}\cdot\nabla_r+F\cdot\nabla_{p1}\right)f_1 = \int d^3p_2d^3p_1^{'}d^3p_2^{'}\delta^4(P_f-P_i)\,|T_{fi}|^2\left(f_2^{'}f_1^{'}-f_2f_1\right). \tag{1}$$

To get a good compromise between computational runtime and physical accuracy, we used the application of microscopic theories together with strong assumptions like neglecting quantum mechanical effects.

The main idea for solving the Boltzmann equation with the Particles-In-A-Cell-method (PIC) consists in dividing a certain volume into many cells with volume $V_{\text{cell}}=\Delta x\Delta y\Delta z$, where we have N particles, which will suffer movement- and collision-laws in a certain time interval Δt. Each particle will have its own position r and momentum p. So if the particle does not collide, its propagation is given by

$$x\mapsto x+v_x\Delta t=x+c^2\frac{p_x}{E}\Delta t\,. \tag{2}$$

The same is valid for the y and z directions.

On the other hand, it is important to consider that the collisions are binary and can only occur between particles in the same cell. Therefore, the probability of collisions to occur is given by using the Monte Carlo method in Δt

$$P_{22}=v_{\text{rel}}\frac{\sigma}{N_{\text{test}}}\frac{\Delta t}{V_{\text{cell}}} \tag{3}$$

being σ the total cross section, which is considered to be isotropic and v_{rel} the relative velocity given by $v_{\text{rel}}=\frac{s}{2E_1E_2}$, where s is the Mandelstam variable, $s=(p_1+p_2)^2$ [1].

To reduce statistical fluctuations and to keep the accuracy of our pretended solution, we use the testparticle method. It consists in introducing $N_{\text{test}}=r_{\text{test}}N$ with r_{test} as a chosen factor, which increases the number of particles. To keep the mean free path λ independent of N_{test} we reduce the probability P_{22} by the same r_{test}. To get the direction of the outgoing momentum we boost from the plasma frame to the center-of-mass frame applying the Lorentz transformation. In the center-of-mass frame we choose the momentum randomly. After that, we boost back to the original frame. If a particle collides with one of the six walls established by the box volume, it will be elastically reflected.

3. CUDA language

CUDA is a language for parallel programming in GPUs, which recently started being used in numerical computations in physics, due to the potential performance increased by order of magnitude.

The CUDA logic is built by writing kernel functions, which calculate the physical matters, in the device and calling them using the host. The device is constituted by various grids which include about 65535^3 blocks for Fermi architectures and 65535^2 blocks for older architectures. Each block has 256 threads. The position and momentum of each particle in Δt is stored in a thread. So we point out that the big advantage of using CUDA consists in the fast shared memory region that can be shared among threads [3, 4] .

4. Flowchart

Our code structure is presented in figure 1.

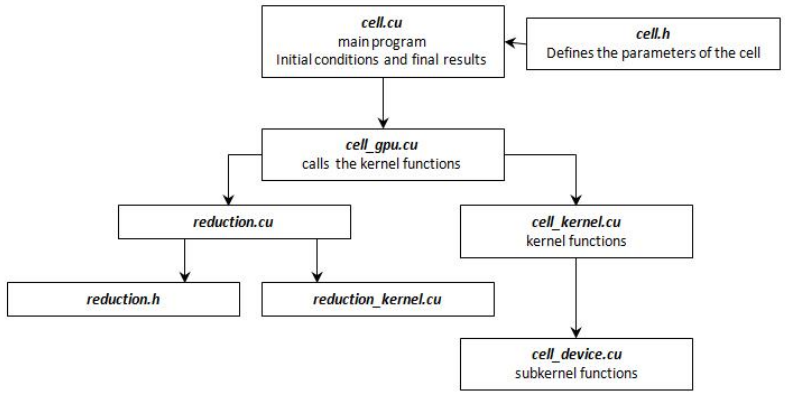

Fig. 1. Flowchart of our CuBA code.

5. Results

To test our code, we have to take into account the initial conditions we choose. The two important parameters are the time variation Δt and x variation Δx, once we consider a transverse homogeneous plan. Δt is always chosen to be smaller than Δx to avoid large local variations in one time step. If we increase Δx, we have to increase the testparticle number N_{test}. The more testparticles we have, more the curve of the Riemann problem approximates to the theoretical solution. A small testparticle number affects the fluctuations. To simulate an ideal fluid, we may choose a very small viscosity.

First, we check some numerical solutions for CuBA considering various parameters. For starters we consider our box volume to be $32^3\,\mathrm{fm}^3$, the cross section, $\sigma = 10\,\mathrm{GeV}^{-2}$, $dt = 0.1\,\mathrm{fm}/c$, with equal particle distribution at the beginning and different temperatures on each side of the box, $T_{\mathrm{left}} = 0.4\,\mathrm{GeV}$ and $T_{\mathrm{right}} = 0.2\,\mathrm{GeV}$. The conservation of the total energy is verified, just as it was expected. We observe the evolution in Δt in figure 2.

In addition, we observe in figure 2 the typical figure of the Riemann problem. This problem consists of a propagating shock wave because the initial conditions impose different temperatures [5].

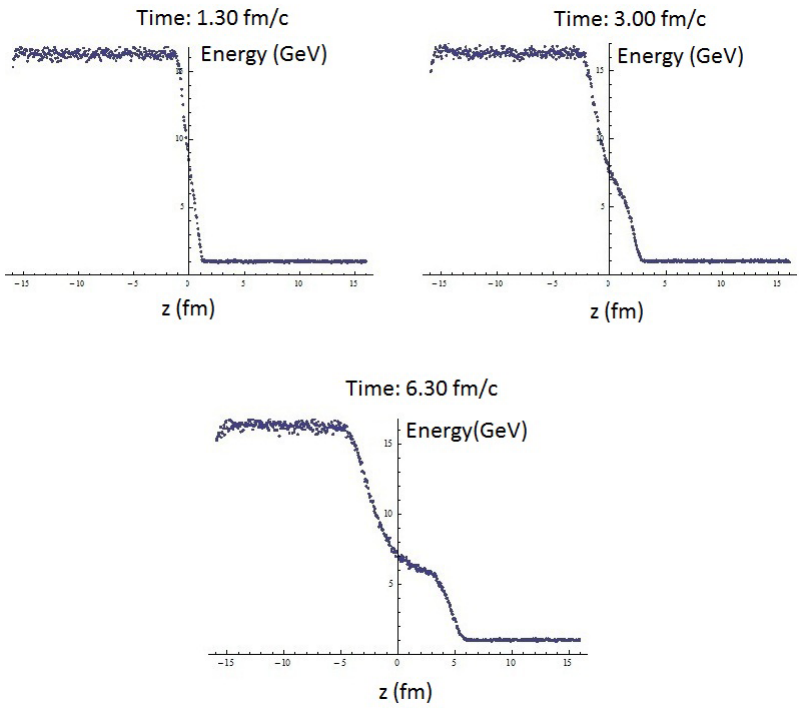

Fig. 2. Evolution of the energy density, shown for different time-slices Δt. The propagation of the two waves from the initial boundary of the Riemann problem is clearly visible.

Secondly, we range the cross section, considering the other variables constant and as previously referred. We observe the differences in figure 3.

As we can verify, the slope undoes itself by increasing the cross section, which physically means to have a larger viscosity.

At last, to compare our results to the BAMPS code we choose the same initial conditions in both codes, which are the ones mentioned at the beginning of this section.

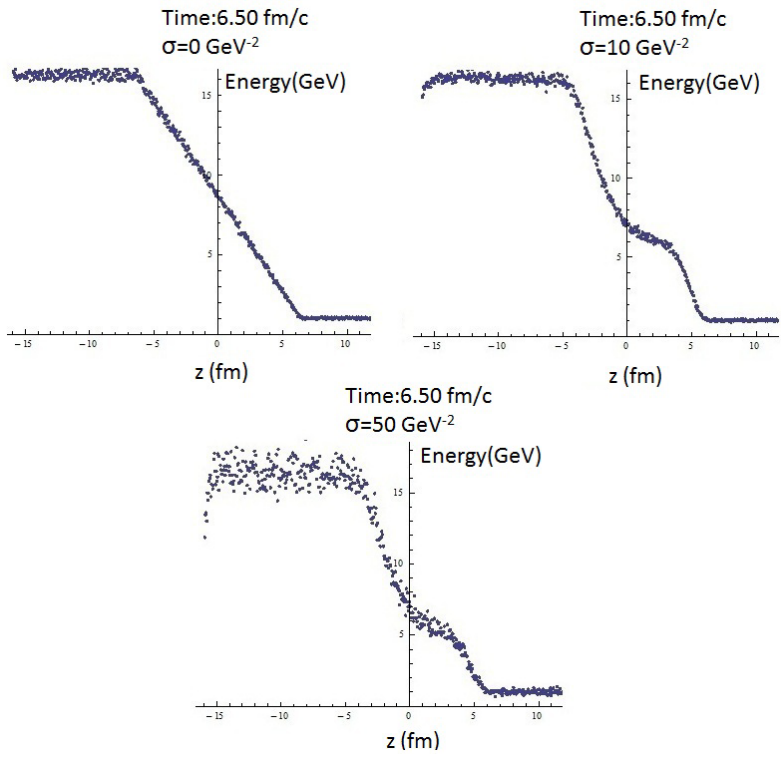

Fig. 3. Evolution of the local cross section, shown for different time-slices.

In figure 4 we can surely identify the overlapping of the results obtained with CuBA (black/blue points) and BAMPS (light gray/red points).

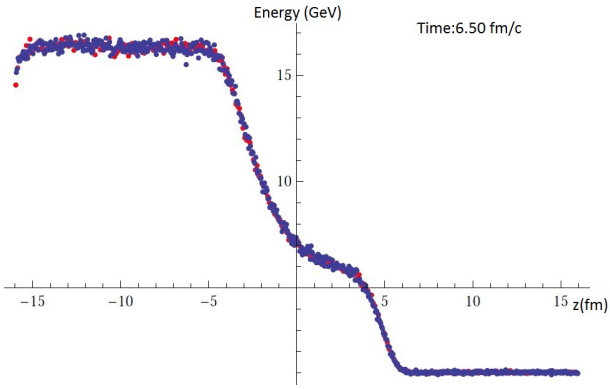

Fig. 4. Comparing the energy density of BAMPS (light gray/red points) with CuBA (black/blue points), both codes produce the same results, except for statistical fluctuations.

While the BAMPS code spends 12 minutes and 36 seconds to calculate the data, CUBA just needs 58.09 seconds.

6. Conclusions

The resulting data can be used to confirm the CPU code and improve the study of shocking particles. For now, we can say that CuBA is about 13 times faster than BAMPS.

In the near future, we pretend to implement the parameter dt as variable and optimize our code in computational runtime. Furthermore, we will check our code with other initial conditions and compare it to BAMPS.

As final result, we expect to obtain a code which is able to calculate any problem of this type and being as fast as CUDA allows us.

This work was financed by the FCT contracts POCI/FP/81933/2007, CERN/FP/83582/2008, PTDC/FIS/100968/2008, CERN/FP/109327/2009, NVIDIA Academic Partnership and the CRUP/DAAD exchange A-10/10. Nuno Cardoso is also supported by FCT under the contract SFRH/BD/44416/2008.

REFERENCES

[1] I. Bouras *et al.*, *Phys. Rev. Lett.* **103**, 032301 (2009) [arXiv:0902.1927v2 [hep-ph]].

[2] NVIDIA, CUDA Toolkit 4.1 CUFFT Library (2012).

[3] *Programming Massively Parallel Processors, A Hands On Approach*, David B. Kirk (NVIDIA Corporation), Wen-mei W. Hwu, Elsevier Inc. (2010), ISBN: 978-0-12-381472-2.

[4] *CUDA by Example: An Introduction to General-Purpose GPU Programming*, J. Sanders, E. Kandrot, Published by Addison-Wesley Professional (2010), Print ISBN-10: 0-13-138768-5, Web ISBN-10: 0-13-218016-2, Print ISBN-13: 978-0-13-138768-3, Web ISBN-13: 978-0-13-218016-0.

[5] I. Bouras *et al.*, *Phys. Rev.* **C82**, 024910 (2010) [arXiv:1006.0387 [hep-ph]].