SPACE CHARGE DOMINATED ENVELOPE DYNAMICS USING GPUs

N. Kulabukhova*, Saint-Petersburg State University, Russia

Abstract

High power accelerator facilities lead to necessity to consider space charge forces. It is therefore important to study the space charge dynamics in the corresponding channels. To represent the space charge forces of the beam we have developed special software based on some analytical models for space charge distributions. Because calculations for space charge dynamics become extremely time consuming, we use a special algorithm for predictor-corrector method for evaluation scheme for beam map evaluation including the space charge forces. This method allows us to evaluate the map along the references trajectory and to create the beam envelope dynamics. The corresponding computer codes are realized using CUDA implementation of maps for particle dynamics. Some numerical results for different types of the beam channels are discussed. The survey of advantages and disadvantages of using different methods of parallelization and some parallel approaches will be done.

INTRODUCTION

Space charge effects can be very important for the dynamics of intense particle beams, as they repeatedly pass through nonlinear focusing elements, aiming to maximize the beam's luminosity properties in the storage rings of a high energy accelerator [1]. The evaluation of the space charge effects on the beam dynamics requires GPU-time intensive numerical simulations. There are some works [3, 4], where parallel algorithms are used. The raw computational power of a GPU dwarfs that of the most powerful CPU, and the gap is steadily widening. Furthermore, GPUs have moved away from the traditional fixed-function 3D graphics pipeline toward a flexible general-purpose computational engine. Today, GPUs can implement many parallel algorithms directly using graphics hardware. Well-suited algorithms that leverage all the underlying computational horsepower often achieve tremendous speedups [2]. In [5, 6] and [7] the algorithm of PIC-method based on object-oriented architecture is described. As the alternative of this method matrix formalism was used. Matrix formalism [8, 9] is a high-performance mapping approach for ODE solving. It allows to present solution of the system in following form

$$X = \sum_{i=0}^{k} R^{1i}(t) X_0^{[i]}, \qquad (1)$$

where R^{1i} are numerical matrices. So this approach can be easy implement in parallel code. Due to the fact that only matrix multiplication and addition are used, GPU programming is especially suitable for this purpose.

There we have two ways for beam dynamics simulation:

- based on particle simulation;
- based on envelope description.

ENVELOPE SIMULATION

Beam dynamics description based on envelope provides an efficient approach to modeling. Let's consider the envelope simulation in linear case(see Fig. (1)). In nonlinear case the equations are quite difficult, but the concept is based on linearization by introducing an extended space.

In linear case equation (1) is wrote in following form

$$X = R \circ X_0.$$

The elliptical envelope can be described by a quadratic form

$$X^* A X < 1,$$

where X^* means transpose of vector X.

By these equations a new envelope can be obtained:

$$X^*(R^* A R)X < 1,$$

where $R^* A R$ is a matrix of new quadratic form.

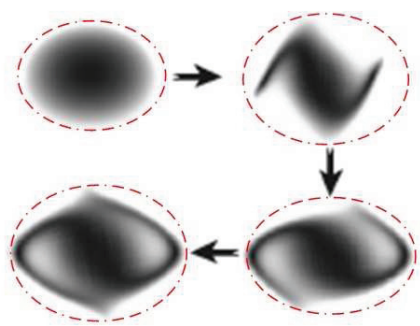

Figure 1: Envelope simulation.

EQUATIONS FOR THE ENVELOPES WITH THE SPACE CHARGE

According to the matrix formalism, we compare the evolution operator and the infinite-dimensional matrix

$$M = (M^{11} \dots M^{1k} \dots),$$

with the help of the block-matrices and using

* kulabukhova.nv@gmail.com

ISBN 978-3-95450-122-9

$$\sigma^{11}(t) = \sum_{l=1}^{\infty}\sum_{k=1}^{\infty} M^{1l}(t \mid t)\sigma_0^{lk}(M^{1k}(t \mid t)),$$

calculate the root-mean-square matrix of envelopes and find the current distribution function to calculate own fields of space charge.

Solution Algorithm

Let $T = [t_0, t_1]$, and $\Delta t = t_1 - t_0$, $f(X, t_0) = f_0(X)$ - the initial value of the set of phase points when $t = t_0$, N - the order of approximation.

- Step one. Calculate matrices σ_0^{ik}, $i, k = \overline{1, N}$ by

$$\sigma_0^{ik} = \int_{\eta_0} f_0(X) X^{[i]} (X^{[k]})^* dX.$$

as a form matrix A_0 choose $(\sigma_0^{11})^{-1}$ or σ_0^{-1} if the initial set η_0 is ellipsoid with the border

$$X_0^* \sigma_0^{-1} X_0 = \epsilon.$$

Next build approximant $\varphi_0(\kappa_0^2) \approx f_0(X_0)$, where $\kappa_0^2 = X_0^* A_0 X_0$.

- Step two. Calculate block-matrices $P^{1k}(B^{ext}, E^{ext}, t)$ and $N_1^{1k} = P^{1k}(B^{ext}, E^{ext}, t)$ [8, 9].
- Step three. Calculate $E^{self} = E(\varphi_0(\kappa_0^2))$ for different distribution of the beam (e.g. uniform, normal, quadratic, etc.).
- Step four. Calculate block-matrices P with space charge effect:

$$\{P^{1k}(t)\}_{ij} = \frac{1}{k_1! \dots k_m!} \frac{d^k F_i(X_j, t)}{dx_1^{k_1} \dots dx_m^{k_m}} |_{x_1 = \dots = x_m = 1}.$$

- Step five. Calculate block-matrices M^{ik} where $i \le k \le N$,

$$M_1^{ik} = M^{ik}(t \mid t_0; \{N_1^{1l}\}),$$

$l = \overline{1, k}$,

$$M_2^{ik} = M^{ik}(t \mid t_0; \{N_2^{1l}\}),$$

$$M_0^{ik} = M_1^{ik} + M_2^{ik},$$

- Step six. Calculate block-matrices σ_0^{ik}

$$\sigma_0^{ik} = \sum_{l=i}^{\infty}\sum_{j=k}^{\infty} M_0^{il}\sigma_0^{lj}(M_0^{jk})^*.$$

- Step seven. Calculate block-matrices - virtual changes of settings while beam evolution:

$$\sigma_1^{ik} = \alpha\sigma_0^{ik} + (1 - \alpha)\sigma_0^{ik},$$

$0 < \alpha < 1$. By virtual change we mean changes of settings that are necessary to build a map. Envelope matrices, functions of distributions, etc., are not changed.

- Step eight. Check the conditions:

$$\| \sigma_1^{ik} - \sigma_0^{ik} \|_c < \varepsilon^{ik} \quad (2)$$

As (2) can be used different equivalent rules. If the condition is right, the process is stoped. Otherwise,

$$\sigma_0^{ik} = \sigma_1^{ik}.$$

and going the next step.

- Step nine. Looking for $\varphi(\kappa^2)$ for function $f(X, t)$:

$$\varphi(\kappa^2) \approx f_0(\mu_0^{-1} \circ X_0) = f_0(\sum_{i=1}^{\infty} T_0^{1i} X_0^{[i]}).$$

Assuming $\varphi_0(\kappa^2) = \varphi(\kappa^2)$ return to step three.

Considered algorithm is simplified by using as approximant funtion that is constant on the ellipsoid and zero outside it. That's why the step three becomes easier to calculate. Step nine is not needed at all and it makes the calculations faster. Moreover, choosing approximant in the class of polynomials allows us to use pre-computed block-matrices from special database. So this approach can significantly reduce the computations on the step of numerical simulations.

SIMULATION

We evaluate the effectiveness of using data parallelism to program GPUs by providing results for a set of compute-intensive benchmarks. All calculations were performed on a hybrid cluster of SPbSU (see Fig. (2)) computing center. Its nodes contain a NVIDIA Tesla S2050 system that was developed specifically as a GPGPU unit [10]. For our goal we choose OpenMP technology (see Fig. (3)). The research have shown that there is no great benefits via parallelization of computational code for one particle by using GPU. In Table 1) performance of different parallelization modes are presented, where

- **1 mult** means one parallel section for multiplication;
- **2 mult** means two parallel section for multiplication;
- **add** means one parallel section for addition.

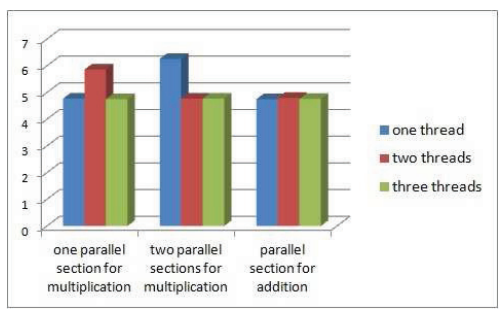

Figure 2: Diagram of comparative performance.

In this case overhead on data sending is significant. On the other hand matrix formalism allows to process a set of the initial points, where parallelization is more preferably.

ISBN 978-3-95450-122-9

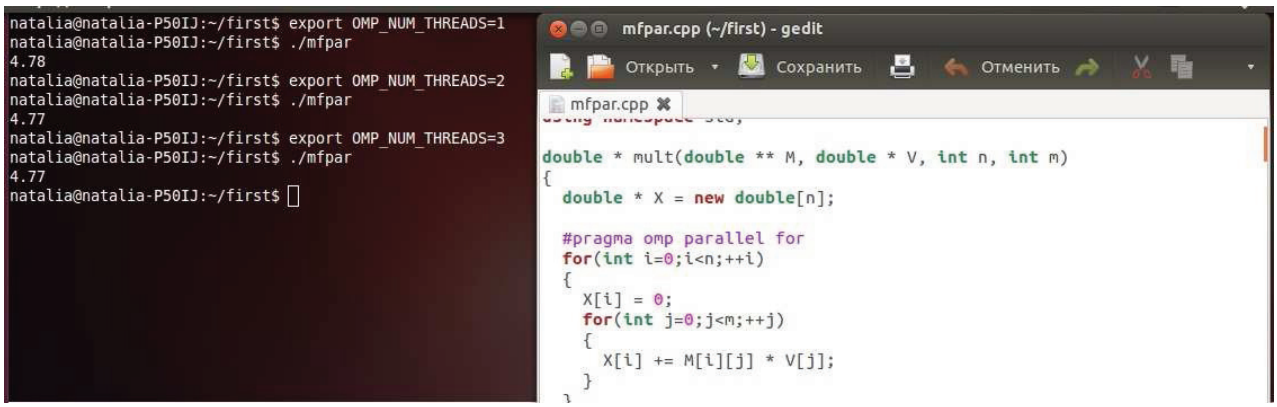

Figure 3: The workflow.

Let's introduce a set of initial particle

$$M = (X_0^1 X_0^2 \dots X_0^p).$$

In according to the equation (1) the resulting points can be calculated

$$M = \sum_{i=0}^{k} R^{1i}(t)((X_0^1)^{[i]}(X_0^2)^{[i]} \dots (X_0^p)^{[i]}). \quad (3)$$

Note that the sizes of matrices in the equation (3) is much greater than in (1) when a set of initial particles is quite large.

Table 1: Performance of Different Parallelization, Sec.

	1 mult	2 mult	add
1	4.77	6.26	4.75
2	5.86	4.77	4.8
3	4.75	4.77	4.76

CONCLUSION

Matrix formalism is a high-performance approach for beam dynamic modeling. The method can be implemented in parallel codes on GPU. It allows simulate both long-term evolution of a set of particles, and evaluating based on envelope description.

ACKNOWLEDGMENT

Computations were partly carried out on cluster HPC-0011654-001 of Saint-Petersburg State University, Faculty of Applied Mathematics and Control Processes. Special thanks for my scientific supervisor S. Andrianov.

REFERENCES

[1] Tassos Bountis, Charalampos Skokos, "Space Charge Can Significantly Affect the Dynamics of Accelerator Maps," arXiv:physics/0605084v1, 2006.

[2] D. Luebke, G. Humphreys, "How GPUs Work," How Things Work, Computer, February, 2007, pp. 126-130.

[3] M. Giovannozzi, "Space-Charge Simulation Using Parallel Algorithms," http://accelconf.web.cern.ch/AccelConf/e98/PAPERS/THP04B.PDF, pp. 1189-1191.

[4] S. Andrianov, N. Kulabukhova, V. Ryabusha, "Space-charge Simulations using Parallel and Distributed Computer Systems," MOPWO018, these proceedings.

[5] J.P.Verboncoeur,A.B.Langdon, N.T. Gladd, "An Object-Oriented Electromagnetic PIC Code," Computer Physics Communications 87, 1995,pp.199-211.

[6] Ji Qiang,R.D. Ryne, S. Habib, V. Decyk, "An Object-Oriented Parrallel Particle-in-Cell Code for Beam Dynamics Simulation in Linear Acceleratitors," Jornal of Computational Physics, 163, 2000, pp.434-451.

[7] K.J. Bowers, "Accelerating a Particle-in-Cell Simulation Using a Hybrid Computing Sort," Jornal of Computational Physics, 2001, pp. 393-411.

[8] S. Andrianov, "The Convergence and Accuracy of the Matrix Formalism Approximation," Proceedings of 11th International Computational Accelerator Physics Conference,2012, pp.195-197.

[9] A. Ivanov, S. Andrianov, "Matrix formalism for long-term evolution of charged particle and spin dynamics in electrostatic fields," Proceedings of 11th International Computational Accelerator Physics Conference,2012, pp.257-259.

[10] N Kulabukhova, "GPGPU Implementation of Matrix Formalism for Beam Dynamics Simulation," Proceedings of 11th International Computational Accelerator Physics Conference,2012, pp.143-145.