GPGPU for Problem-Solving Environment in Accelerator Physics

Nataliia Kulabukhova(✉)

Saint-Petersburg State University, Saint Petersburg, Russia
n.kulabukhova@spbu.ru

Abstract. The paper contains the survey of benefits of using graphical processors for general purpose computations as a part of problem-solving environment in the beam physics studies. The comparison of testing numerical element-to-element modelling on CPU and the long-turn symbolic simulation with the general purpose GPUs in the working prototype is made. With the help of the graphical processors from both sides - the general purpose computations and the graphical units itself - the analysis of beam behaviour under the influence of the space charge is done.

1 Introduction

The use of graphical processors in the case of simulation of beam dynamics is widely spread through particle accelerator scientists [1–6] today as, without a doubt, a very popular approach, though it is not quite a new one in other areas of science. But in accelerator physics there are some problems with porting the earlier realised program components to the hybrid architecture. We will speak about it below.

Last year we present the idea of using graphical processors for simulation and visualization in the terms of constructing Virtual Accelerator Laboratory (VAL) [7]. In the work we described how to develop a model of real machine and test the behaviour of the beam inside the virtual system of control elements, such as dipoles, quadrupoles, sextupoles and drifts. As far as VAL is a problem-solving environment, it is divided in to special blocks of components.

In this paper we will involve only this set of components:

The block of control elements: with the help of it the user can construct the main view of the future accelerator (for example, Fig. 1).

The block of particle distribution: which forms the initial particle distribution coming from the source.

The block of data: the data base with all calculated results.

The block of GPU vizualization: the result of calculations presented in a 3D graphical form.

The work is supported by RFBR 16-07-01113A.

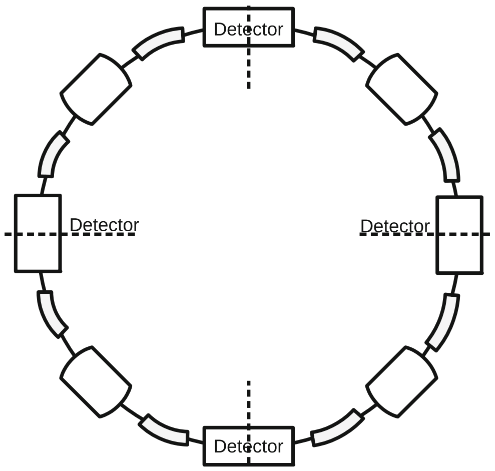

Fig. 1. Schematic view of accelerator ring for modelling the long-turn evolution

2 The Idea of the Matrix Form of the ODE

The main idea of the concept is to describe the behaviour of some dynamic system with the set of ordinary differential equations presented in the matrix form. Let us have a system:

$$\frac{dX}{dt} = F(X, t), \tag{1}$$

where $F(X, t)$ - arbitrary, analytic in the neighborhood $X = 0, X \in R^n$ and measurable $t \in [t_0, t_s] \in R^n$ function.

According to [8], Eq. (1) can be written:

$$\frac{dX}{dt} = \sum_{i=0}^{k} \mathbb{P}^{1i}(t) X^{[i]}, \tag{2}$$

where $\mathbb{P}^{1i}$ matrix with symbolic coefficients obtained from the original ODE. This matrix has the form:

$$\mathbb{P}^{11} = \begin{pmatrix} \mathbb{P}^{11} & \mathbb{P}^{12} & \dots & \mathbb{P}^{1k} & \dots \\ \mathbb{O} & \mathbb{P}^{22} & \dots & \mathbb{P}^{2k} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ \mathbb{O} & \mathbb{O} & \dots & \mathbb{P}^{jk} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{pmatrix},$$

Dimensions of control matrices depend on the length of the vector X and the order of nonlinearity necessary for the experiment.

In the course of symbolic computation elements of final matrix $\mathbb{P}$ can be calculated by series expansion:

$$\left\{\mathbb{P}^{1k}(t)\right\}_{ij} = \frac{1}{k_1! \dots k_n!} \left. \frac{\partial^k \mathbf{F}_i(X_j, t)}{\partial x_1^{k_1} \dots \partial x_n^{k_n}} \right|_{x_1 = \dots = x_n = 1}$$

Depending on the problem Eq. 1 can be represented in numerical form:

$$X = \sum_{i=0}^{k} \mathbb{R}^{1i}(t) X_0^{[i]}, \tag{3}$$

where $\mathbb{R}^{1i}$ – matrix with numerical coefficients.

Initial vector $X = \{x, y, x', y', s, \dots\}^T$ can also be written as a matrix consisting of vectors:

$$\mathbf{X}_0 = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ x'_1 & x'_2 & \cdots & x'_n \\ y'_1 & y'_2 & \cdots & y'_n \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$$

Thus, during calculations is the multiplication of the matrix by the matrix:

$$\mathbf{X}_s = \sum_{i=0}^{k} \mathbb{R}^{1i}(t) \mathbf{X}_j^{[i]}, \tag{4}$$

3 The Space Charge Example

The these part describes how it works on the example of the influence of space charge on the dynamics of the beam.

The equations of the cross-section motion in the beam without bunches can be written in the following form:

$$x'' = \frac{q}{p}\sqrt{1 + x'^2 + y'^2}(y'B_s - (1 + x'^2)B_y + x'y'B_x + \sqrt{1 + x'^2 + y'^2}\frac{E_x}{c\beta\gamma}),$$
$$y'' = -\frac{q}{p}\sqrt{1 + x'^2 + y'^2}(x'B_s - (1 + y'^2)B_x + x'y'B_y + \sqrt{1 + x'^2 + y'^2}\frac{E_y}{c\beta\gamma}) \tag{5}$$

In the linear case for example for quadrupole they will look as shown below

$$x'' + \frac{qB_{xy}}{m_0 c\beta\gamma}x - \frac{q}{\epsilon_0 m_0 c^2 \beta^2 \gamma^3} E_x^L = 0,$$
$$y'' - \frac{qB_{yx}}{m_0 c\beta\gamma}y - \frac{q}{\epsilon_0 m_0 c^2 \beta^2 \gamma^3} E_y^L = 0 \tag{6}$$

According to the matrix form the influence of the space charge will modify the Eq. (2) this way:

$$\frac{dX}{dt} = \sum_{i=0}^{k} (\mathbb{P}_e xt^{1i}(s) + \mathbb{P}_s elf^{1i}(s)) X^{[i]}, \tag{7}$$

Now we have the part, which respond to the external field and the part of the self space charge forces. The external field describes by the components of the vector of magnetic induction. And for the space charge responds the intensity vector.

Matrices ${\mathbb{P}_{ext}}^{11}(s) + {\mathbb{P}_{self}}^{11}(s)$ in linear case will be:

$$\mathbb{P}_{ext} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -k_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_y & 0 \end{pmatrix} \quad (8) \qquad \mathbb{P}_{self} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\eta_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\eta_y & 0 \end{pmatrix} \quad (9)$$

For non-linear case the Eq. (5) will be

$$\begin{aligned} x'' + (K_x - L_x)x &= a_x x^3 + b_x x {x'}^2 + cxy^2 + (K_x)' xyy' + d_x x {y'}^2 + K_x x' yy', \\ y'' + (K_y - L_y)y &= a_y y^3 + b_y y {y'}^2 + cyx^2 + (K_y)' yxx' + d_y y {x'}^2 + K_y y' xx' \end{aligned} \quad (10)$$

where

$$K_x = \frac{qB_x y}{m_0 c \beta \gamma},$$

$$K_y = \frac{qB_y x}{m_0 c \beta \gamma},$$

$$L_x = \alpha \frac{qI}{\pi \epsilon_0 m_0 c^3 \beta^3 \gamma^3 r_x (r_x + r_y)},$$

$$L_y = \alpha \frac{qI}{\pi \epsilon_0 m_0 c^3 \beta^3 \gamma^3 r_y (r_x + r_y)}.$$

Therefore the external and self matrices will take the following form

$$\mathbb{P}^{11} = \begin{pmatrix} {\mathbb{P}^{11}}_x & 0 \\ 0 & {\mathbb{P}^{11}}_y \end{pmatrix} \qquad {\mathbb{P}^{13}}_x = \begin{pmatrix} {\mathbb{P}^{11}}_x & \mathbb{O} & {\mathbb{P}^{13}}_x & \mathbb{O} \\ 0 & {\mathbb{P}^{22}}_x & \mathbb{O} & {\mathbb{P}^{24}}_x \end{pmatrix}$$

The idea of the matrix form of the ordinary differential equations is not new [9]. But with the help of evolution of the graphical processors these form of the equations get a new life. And next the usage of these method will be presented.

4 Linear Cases on CPU

The workflow of the environment is shown on the scheme on Fig. 2. The initial distribution of the particles and the structure of the machine is set to the database to begin working with. The user can choose the style of the machine: linear or cyclic. Then the mode of the simulation can be defined: element-to-element or long-turn evolution. By element-to-element we mean the test of the system, when the visualization of particle distribution is made after each element.

The results of computations in linear case for different distributions are shown on the pictures below. The system in this example consists of several drifts and quadrupoles under the FODO concept

$$\begin{tikzcd}\boxed{\text{drift}} \arrow[r] & \boxed{\text{FQ}} \arrow[r] & \boxed{\text{drift}} \arrow[r] & \boxed{\text{DQ}} \arrow[r] & \boxed{\cdots} \arrow[r] & \boxed{\text{drift}} \arrow[r] & \boxed{\text{FQ}} \arrow[r] & \boxed{\text{drift}} \arrow[r] & \boxed{\text{DQ}} \arrow[r] & \boxed{\text{drift}}\end{tikzcd}$$

where DQ - defocusing quadrupole, FQ - focus quadrupole.

The studies show that the algorithm works with commonly used distributions - Gauss (Fig. 3) and uniform (Fig. 4). On Fig. 5 new Gauss distribution of 10000 particles on CPU is shown. And on Fig. 6 new uniform distribution of 10000 particles on CPU is shown.

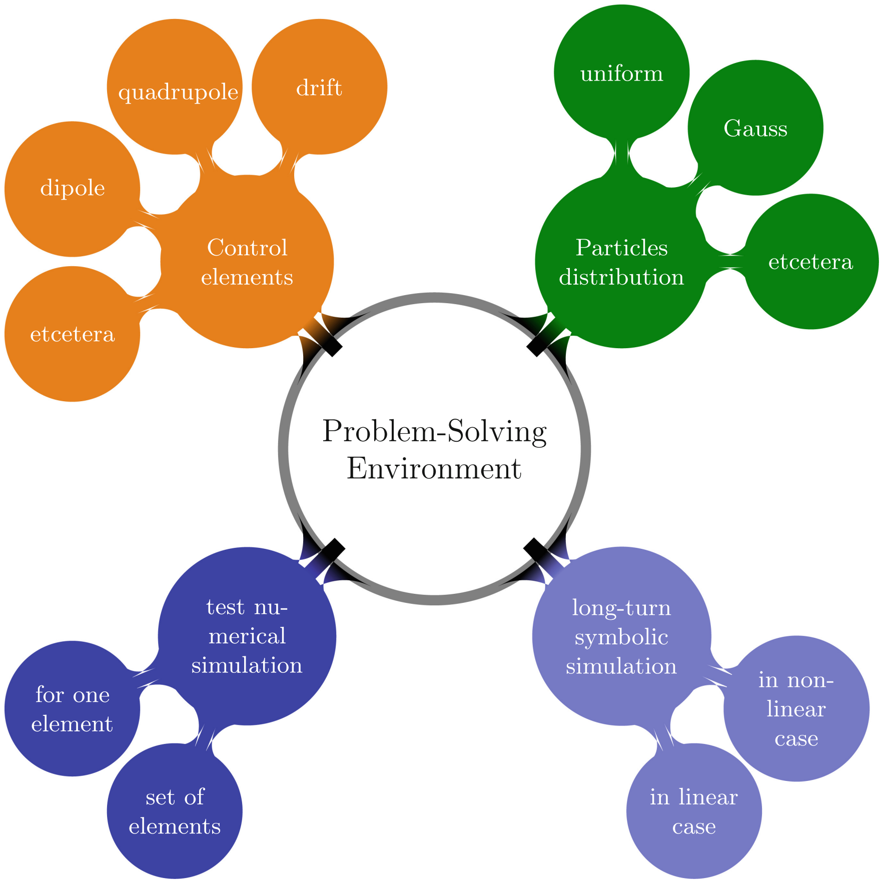

Fig. 2. General scheme of the workflow of the PSE

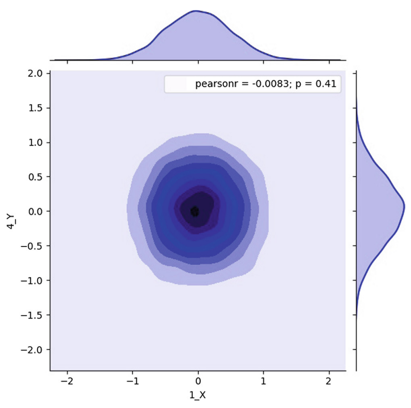

Fig. 3. Initial Gauss distribution, N = 10000 particles

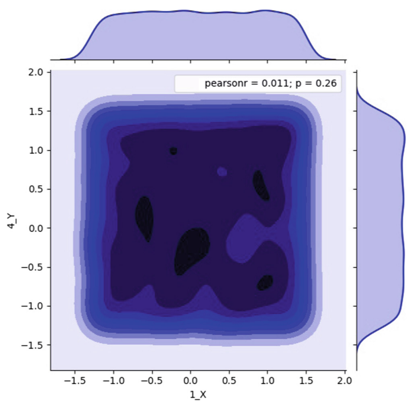

Fig. 4. Initial Uniform distribution, N = 10000 particles

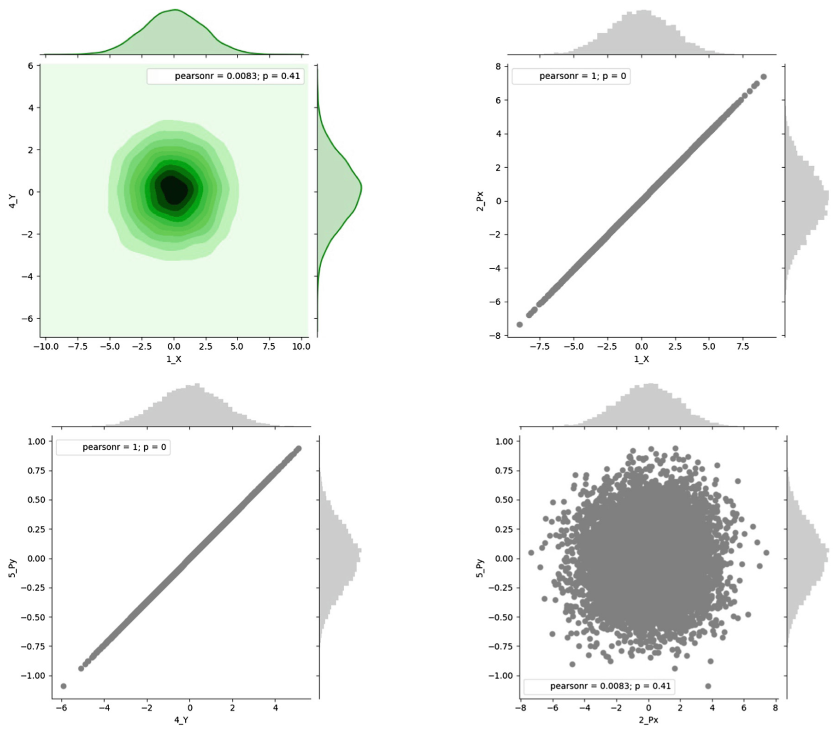

Fig. 5. New Gauss distribution of 10000 particles on CPU in (X, Y), (X, P_x), (Y, P_y), (P_x, P_y)

5 Long-Turn Evolution of the Beam on GPU

When we speak about the linear machine with some meters lengthwise, the personal computer will solve our problems. But in the case of the storage ring with the long-turn evolution of the beam, we need the machine like the accelerator itself to get the result in suitable time. As the result of transformation of the ODE we get matrices in the sparse form. The problem is how to compute them on the GPU in the most effective way. The advantage of using sparse matrices on GPU is that we have a lot of zero elements, which can be empty components to be loading from the CPU memory to the memory of the GPU. But the fact that we get the partial result from every element the memory exchanges will outweigh the advantages of the sparse matrices. For that reason we compute the part of accelerator (rather the quarter of it) in the symbolic way to get the total matrix of this part to load it on the device. As the whole machine consists of symmetric sections, with some corrections these matrix can be used for modelling the long-turn evolution of the beam with minimum (in some way) computational resources. The schematic view of such system is shown on the Fig. 1.

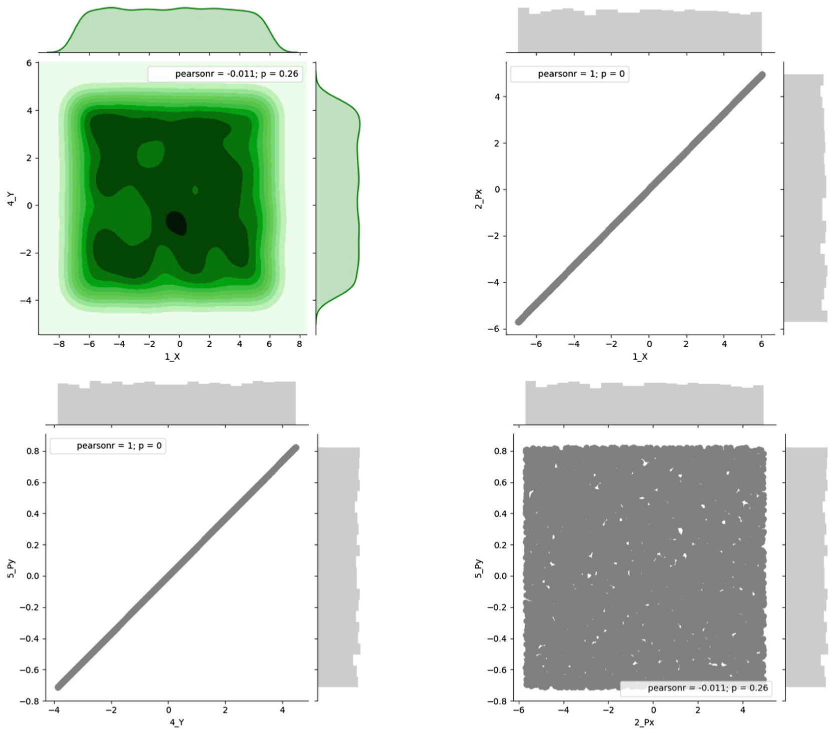

Fig. 6. New uniform distribution of 10000 particles on CPU in (X, Y), (X, P_x), (Y, P_y), (P_x, P_y)

Computations were made on GeForce GTX 1060 6 GB with compute capability 6.1. The whole environment is developed on Python 3.6 libraries. Obviously, Python is not a quick tool for computations, but it is used as background of all important components. Python plays a role of something like glue, it links everything in one working system. The main computational blocks are constructed with the following modules:

NumPy – scientific computations on CPU [10];
pyCUDA – general purpose computations on GPU [11];
SymPy – symbolic calculations of control elements for sending on GPU [12];
TkInter – graphical user interface [13];
Seaborn – visualization of obtained results [14];

Furthermore, there are some libraries, which are going to used for future work with machine learning and neural networks, such as Scikit-learn [15] and TensorFlow [16]. The results of computations of new Gauss distribution of 1000000 particles on GPU are shown on Fig. 7.

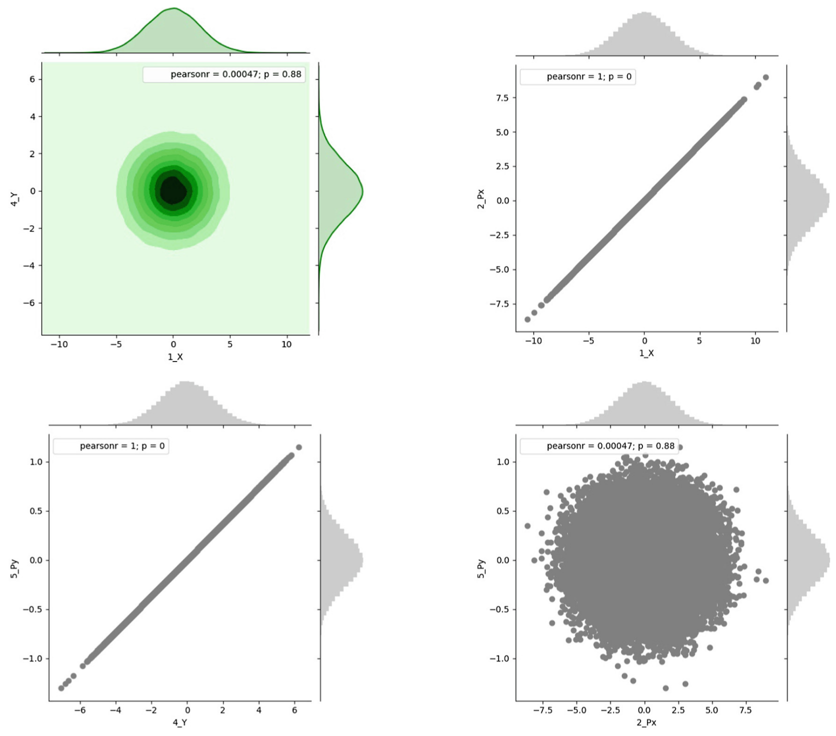

Fig. 7. New Gauss distribution of 1000000 particles on GPU in (X, Y), (X, P_x), (Y, P_y), (P_x, P_y)

6 Discussion

In works [17–19] the concurrency is made for a simple part of the large particle physics simulation toolkits, such as Geant4 and Elegant. The difficulty is that these toolkits were developed for CPU only systems. And they cannot be easily ported for general purpose GPU computations. On the contrary, the method of matrix form of ODE has parallel principle inside. Besides, we constructing the problem-solving environment as a toolkit based on the idea of matrix representation of ODE, but not in reverse. In this case, there is no such a problem of adapting, because we develop a parallel system originally.

The main difficulty in GPGPU development for the matrix algorithm is the amount of data sent on device. It is a bottleneck of practically every GPU program. As every non-zero P^{ij} matrices can be calculated separately and all zero elements are not needed for computations and not sent on device, save resources for the set of particles.

7 Conclusion

The idea of the problem-solving environment is to provide the scientist with a clear and easy environment to simulate and predict the behaviour of the beam. The result of the simulation should be shown in numerical and graphical ways.

In this work the long-turn simulation of the beam with the help of the graphical processors is made. The future work will be dedicated to visualization of the received results. Besides, the KV-distribution is under testing, and the results of simulation using this distribution can be compared with other works.

Another idea is to use the machine learning algorithms to analyse the numerical data and the neural networks, especially the deep learning approach to process visual data of the experiment.

Acknowledgments. The author would like to express gratitude to Vladimir Korkhov for valuable help. The work was sponsored by the Russian Foundation for Basic Research under the project: 16-07-01113 “Virtual supercomputer as a tool for solving complex problems” and the SPbSU equipment project: 9.40.1615.2017 “Deployment of experimental high-performance computing infrastructure to support scientific research of the Department of computer modelling and multiprocessor systems”.

References

1. Kulabukhova, N., Andrianov, S.N., Bogdanov, A., Degtyarev, A.: Simulation of space charge dynamics in high intensive beams on hybrid systems. In: Gervasi, O., et al. (eds.) ICCSA 2016. LNCS, vol. 9786, pp. 284–295. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42085-1_22
2. Kulabukhova, N.: Software for virtual accelerator environment. In: RuPAC 2012 Contributions to the Proceedings. JACOW (2012)
3. Petrov, D.A., Stankova, E.N.: Use of consolidation technology for meteorological data processing. In: Murgante, B., et al. (eds.) ICCSA 2014. LNCS, vol. 8579, pp. 440–451. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09144-0_30

4. Stankova, E.N., Balakshiy, A.V., Petrov, D.A., Shorov, A.V., Korkhov, V.V.: Using technologies of OLAP and machine learning for validation of the numerical models of convective clouds. In: Gervasi, O., et al. (eds.) ICCSA 2016. LNCS, vol. 9788, pp. 463–472. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42111-7_36
5. Bogdanov, A., Degtyarev, A., Korkhov, V., Gaiduchok, V., Gankevich, I.: Virtual supercomputer as basis of scientific computing. In: Horizons in Computer Science Research, vol. 11. Nova Science Publishers (2015)
6. Korkhov, V., Kukla, T., Krefting, D., Terstyanszky, G.Z., Caan, M., Olabarriaga, S.D.: Exploring workflow interoperability tools for neuroimaging data analysis. In: Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science (2011)
7. Kulabukhova, N., Bogdanov, A., Degtyarev, A.: Problem-solving environment for beam dynamics analysis in particle accelerators. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 473–482. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_35
8. Andrianov, S.N.: Dynamical Modeling of Control Systems for Particle Beams. Saint Petersburg State University, Saint Petersburg (2004)
9. Miklos, S.: Electron and Ion Optics. Mir, Moscow (1990). (in Russian)
10. NumPy: the fundamental package for scientific computing with Python. http://www.numpy.org/
11. pyCUDA: Nvidia's CUDA parallel computation API for Python. https://documen.tician.de/pycuda/
12. SymPy: a Python library for symbolic mathematics. http://www.sympy.org/en/index.html
13. TkInter: Pythons de-facto standard GUI package. https://wiki.python.org/moin/TkInter
14. Seaborn: Python visualization library based on matplotlib. http://seaborn.pydata.org/index.html
15. Scikit-learn: tools for data mining and data analysis. http://scikit-learn.org/stable/
16. TensorFlow: an open source machine learning framework
17. Seiskari, O., Kommeri, J., Niemi, T.: GPU in Physics Computation: Case Geant4 Navigation (2011). https://arxiv.org/pdf/1209.5235.pdf
18. Amyx, K., Balasalle, J., King, J., Pogorelov, V., Borland, M., Soliday, R.: Beam dynamics simulations with a GPU-accelerated version of elegant. JACOW (2013)
19. King, J.R., Pogorelov, I.V., Amyx, K.M., Borland, M., Soliday, R.: GPU acceleration and performance of the particle-beam-dynamics code Elegant (2011). https://arxiv.org/pdf/1710.07350.pdf