AN IDE FOR SPIN-ORBIT DYNAMICS SIMULATION

A. Ivanov*, N. Kulabukhova, St. Petersburg State University, Russia

Abstract

In this paper a prototype of an IDE for simulation of spin-orbit motion is described. It is based on the component software development and provides a flexible graphic user interface. One of the main parts of it is numerical methods for ordinary differential equations integration. For numerical simulation it is possible use either the matrix map algorithm or traditional step-by-step methods. This workflow allows choosing one of numerical algorithms and to provide necessary computational experiments. It is also contains both a visual designer of an accelerator lattice and additional tools for control parameters of the model. There is also exists possibility for code generation in different programming languages and computation on high-performance systems.

INTRODUCTION

The key idea of the research is to develop an IDE (Integrated Development Environment) for beam dynamics simulation and accelerator designing. This program have to compute the dispersion and betatron functions, etc., and also provide tracking of spin-orbit motion of particles. In the linear case transfer matrices are used. For the nonlinear dynamics matrix integration [1] (up to the necessary order) and step-by-step integration [2] are applied. In both cases symplectification is available.

The appointment of any IDE is to facilitate the process of programming and to unify access for different tools. So the main requirements for an IDE for beam dynamics investigation are the following

- syntax highlighting and intelli-sense;
- grafical tools for designing;
- workflow for model description;
- code generation possibility.

The last one means both converters to well known beam dynamics simulation codes (MAD, OptiM, COSY Infinity) and ability to generate codes for general-purpose mathematical packages (e.g. Matlab, Mathematica, Maple) and programming languages (C++, Fortran, etc.).

The prototype of the IDE was developed on the .NET framework and permits only Windows operation system. To achieve the cross-platform feature in release version the Qt framework will be used. It will lead us to an environment that can be run in wide range of different platforms with a single interface.

* 05x.andrey@gmail.com

IDE DESCRIPTION

The key for understanding how this IDE works is imagining the process of an accelerator designing and tracking. The main concept of it is project, that means a collection of files. It includes lattice description, settings (characteristics of the particle and energy) and code behind. After a project will built in the solution folder output files will be generated and researcher will achieve possibility to simulation in the model. Such architecture is commonly used in IDE and helps to organize a project in convenient and scalable form.

For errors tracking the special output stream is used. The researcher can obtain detailed information to understand why something does not work. The errors are displayed interactively in special window.

Solution explorer

The window represents project files in a tree view. The start-up file are indicated by bold font. For instance, a researcher can create several lattices and sorage them within one project. In pre-build activity it is necessary to indicate, which one must be used. The Solution Explorer window allows all standart activities, such as copy, rename and delete files.

Workspace

The workspace of the IDE is a tab control. Files with different extensions are presented as a tab with corresponded editable area. For code behind it is a text editor or visual designer. Each tab can be saved and closed. There is also indicator "star" near the modified files.

```
1:  Lattice Create()
2:  {
3:      len = 9.68599;
4:      cd1 = CylindricalDeflector(len, 24.6651962934106,
5:
6:      return cd1;
7:  }
8:
9:  void OnElementCompleted(int elementID)
10: {
11:     // TO DO: describe the activities that will be run after each element of the lattice
12:     // elementID is an element indetifier
13: }
```

CylindricalDeflector(length, radius, **voltage**, aperture)
voltage: voltage in V at side of deflector

Figure 1: Text editor for code behind.

Code behind

For code writting a text box with syntax highlighting and intelli-sense is used (see Fig. 1).

In code researcher can use all standart matheatical operators and functions, and instructions *if, else, for*. One can define own functions and use it. But for logic implementation it is necessary to define two functions:

ISBN 978-3-95450-122-9

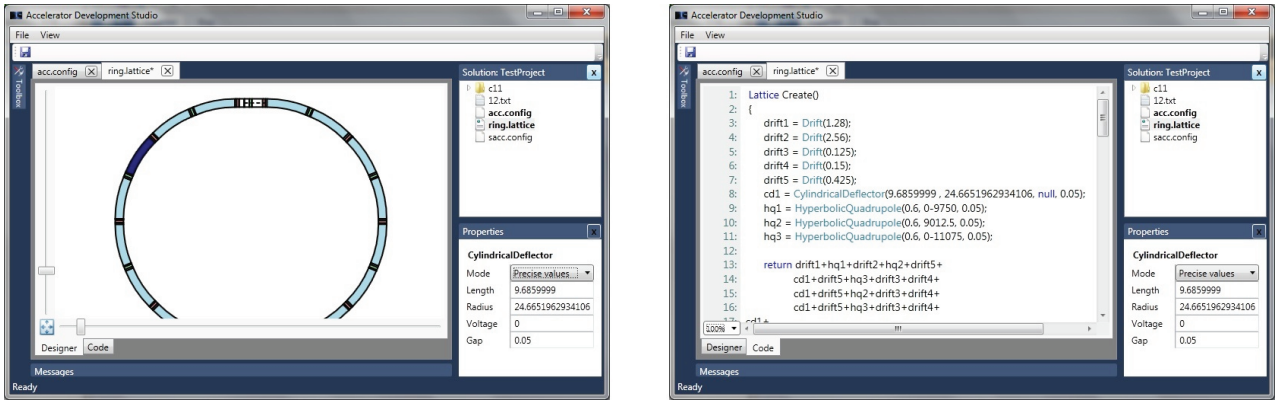

Figure 2: Visual lattice designer and code behind.

- Create function for lattice description;
- OnElementCompleted describes the activity that will performed after each element during the simulation process;
- OnTurnCompleted describes the activity that will performed after each turn during the simulation process.

Visual designer

Visual designer (see Fig. 2) is a tool that exist in almost any IDE (e.g. Qt Creator, Visual Studio). It allows to bind code and visual lattice description. The researcher can change optics via Properties window and this changes will reflect into the code authomatically. In visual mode drag and drop events can be also used for optics development. Visual designer and code editor provide a powerful tool for lattice description. Flexible parameter settings in code behind can realize sophisticated logic. While visualization help us to check and verify it.

Workflow

The programming language that is used in code behind is quit simple. It uses C-like syntax with dynamic typization. It means that you can declare variables in any place and of any type. All operations will be checked during project building. To create a lense one must only call function with the necessery parameter list. In program code the researcher can specify the activity that must bu done during the simulation after each element is tracked. For example, one can introduce random errors for field distribution or run an additional effect.

Nevertheless one can create model and describe lattice without any code writing. For this purposes a workflow can be used. This mechanism is similar to such visual programming systems as LabVIEW or MATLAB/Simulink.

In the workflow the researcher describes the experiment model in visual schemes. Then different computational methods and models can be used for a particular step. For instance, for a researcher there is no difference in using step-by-step integration methods or other approaches, he only indicates the desired one. The workflow will choose the actual solver and appropriate computing resources.

Figure 3 shows a schematic view of the developed workflow. A number of components are mentioned on this figure:

- Lattice designer allows constructing a lattice via predefined elements by settings their physical parameters. It consists of a text editor and GUI with drag and drop capability. In Figure 3 the green circle represents the lattice designer.
- Fringe Fields module generates fringe fields near the selected elements. The user can choose different models of field distribution, e.g. linear or Enge functions.
- Particle Description element defines the kind of particle and its properties used for modeling.
- Solvers process the accelerator declaration and particle data. This is the most resource consuming step which is executed on distributed computing resources.
- Code generation module provides tool for generation computational code in different languages for using in such packages as MATLAB or Mathematica.

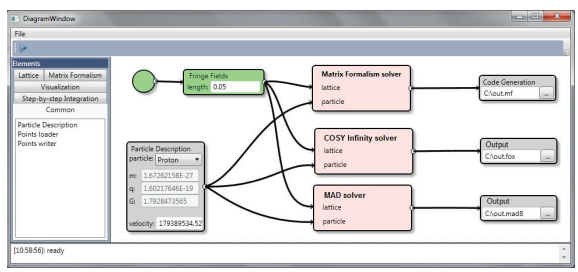

Figure 3: Workflow interface.

The key functionality of the IDE in beam dynamics is to provide a high-level environment for accelerator design, simulation and analysis. Implementation issues are hidden from the users. If the selected method needs symbolic calculations or Taylor series expansion, they will process without direct user activity. On the other hand all information and model descriptions are available. So the researcher can be sure what exactly he calculates and how to interpretate the results.

ISBN 978-3-95450-122-9

Code generation

The main approach that is used for simulation is a numerical implementation of matrix formalism [3]. This method allows to represent an element of accelerator lattice by a set of the numerical matrices

$$X = R^1 X_0 + R^2 X_0^{[2]} + R^3 X_0^{[3]} \ldots,$$

where X is a state vector of particlle description. Note that the linear part R^1 is transfer matrix exactly, and matrix R^k is related to the non-linearities of order k. Moreover, it is possible to compute such accelerator parameter as the dispersion and betatron functions based on this matrix map elements.

The researcher can directly run simulation in IDE, visualize the result or save data. Alternatively in order to achieve the increase of performance one can generate computational code in different program languages (e.g. see Fig. 4):

- C++ (with OpenMP);
- MATLAB (with Parallel Computing Toolbox);
- Fortran;
- .NET Framework languages (with Parallel Extension).

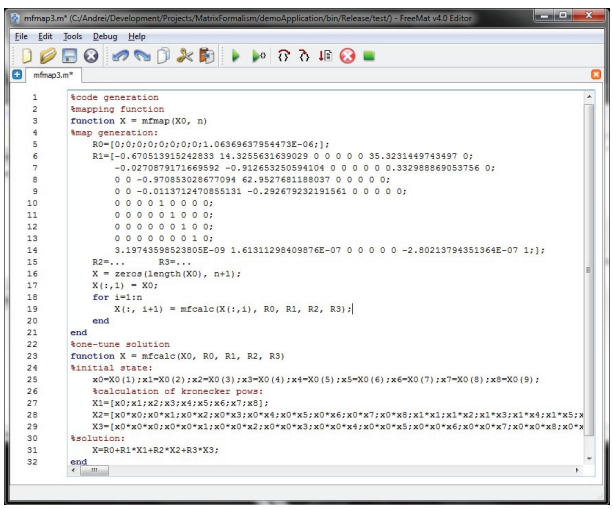

Figure 4: Matlab output file.

High-performance module

The IDE provides an abstract declaration of an accelerator machine and numerical methods that used for simulation. In matrix integration mode it is possible to run the task in a parallel system. The most natural approach in this case is GPU calculation [4]. But the computatonal task can be also run in multiprocessor systems.

This can be difficult for researcher to write a parallel program where memory locality is crucial to performance or where the programmable parts of GPUs have had limited support for primitive types typically found on well known languages. So a low-level library is integrated to the IDE. It provides unified access to such technologies as OpenCL, CUDA, OpenMP, MPI, without code writing.

CONCLUSION

It must be emphasized that the proposed IDE is a modeling environment and is not directly related to the real accelerator control systems. However, the organization of communications between such systems is possible [5].

The task of running diverse software packages (MAD, COSY Infinity, etc.) that have different requirements for the installed operating systems, libraries and other dependencies can be simplified by using the technology of cloud computing. In this case, the virtual machine images ready to set up and configured simulation package can be deployed on provided computing resources. In addition, the use of Cloud enables the experiment in case of lack of local computing resources, as well as in the mode of "urgent computing", when it is necessary to get the results to a preset time.

ACKNOWLEDGMENT

The authors would like to thank Yurij Senichev and Serge Andrianov for continues discussion on mathematical modeling and critical verification of software implementation. Also thanks to Vladimir Korkhov for software architecture discussion.

REFERENCES

[1] A. Ivanov, S. Andrianov, "Matrix Formalism for long-term evolution of charged particle and spin dynamics in electrostatic fields," Proceedings of ICAP2012, Rostock, Germany, 2012. P.187-189.

[2] W. Oevel, M. Sofroniou, "Symplectic Runge-Kutta schemes II: classification of symmetric methods," http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.5060

[3] A. Ivanov, "Comparison of Matrix Formalism ans step-by-step integration for the long-term dynamics simulation in electrostatic fields," Proceedings of RuPAC2012, St. Petersburg, Russia, 2012. P.370-372.

[4] N.V. Kulabukhova, "GPGPU Implementation of Matrix Formalism for Beam Dynamics Simulation," Proceedings of ICAP2012, Rostock, Germany, 2012. P.59-61.

[5] N.V. Kulabukhova, A.N. Ivanov, V.V. Korkhov, D.A. Vasyunin, S.N. Andrianov, "Virtual accelerator: grid-oriented software for beam accelerator control system," Book of abstracts of the 5th International Conference on Distributed Computing and Grid Technologies in Science and Education, 2012.

ISBN 978-3-95450-122-9