ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-MOPME011

MATRIX INTEGRATION OF ODES FOR SPIN-ORBIT DYNAMICS SIMULATION

A. Ivanov*, Yu. Senichev†
Institute of Nuclear Physics, Forschungszentrum Juelich
on behalf of the JEDI Collaboration

Abstract

MODE (Matrix integration of Ordinary Differential Equations) is a software package that provides nonlinear matrix maps building for spin-orbit beam dynamics simulation. In this article we briefly describe the developed integrated development environment features and present computational comparison with other simulation tools. MODE mathematical model is based on Newton-Lorentz and T-BMT equations that are expanded to Taylor series up to the necessary order of nonlinearity. The numerical algorithm is based on matrix presentation of Lie propagator. Spin-orbit simulation results of MODE are compared with results of COSY Infinity and OptiM. MODE provides a flexible graphic user interface, code auto complete technology and visual designer for accelerators. There is also a possibility to generate codes in different programming languages and parallelization techniques.

INTRODUCTION

The key idea of the research is to develop an IDE (Integrated Development Environment) for beam dynamics simulation and accelerator designing. This software implements a mapping approach for ODEs solving based on nonlinear matrix integration technique. A map is represented as a set of numerical matrices. Each of ones corresponds to the certain nonlinear order. In other words the solution of a system of ODEs describes as the matrix Taylor series expansion.

Finding this expansion means estimation of Taylor series coefficients. For this purposes a symplectic step-by-step integration scheme [1] is applied. After a map is built it can be used for long turn dynamics simulation.

In the articles [2] the mathematical models that are encapsulated into MODE are presented. Spin-orbit dynamics is described by Newton–Lorentz and T–BMT equations in curvilinear coordinate system. The coordinates corresponds to the design orbit and can be written as a state vector $\mathbf{X} = (x,\, y,\, t, p_x/p_0,\, p_y/p_0,\, \delta W, S_x,\, S_y,\, S_z,\, L)$, where $x,\, y$ are transverse and vertical offsets of a particle, t is physical time of motion, $p_x,\, p_y$ are transverse and vertical components of momentum, p_0 is the momentum, δW is energy deviation, $\mathbf{S} = (S_x,\, S_y,\, S_s)$ is vector of spin, L is a length of trajectory.

According to the matrix form of Taylor series expansion the general solution can be find

$$\mathbf{X} = \mathbb{R}_0 + \mathbb{R}_1\mathbf{X} + \mathbb{R}_2\mathbf{X}^{[2]} + \ldots,$$

where $X^{[k]}$ means k-th Kronecker pow of state vector $\mathbf{X}$.

This solution describes a particle dynamics with initial condition $\mathbf{X}(0) = \mathbf{X}_0$. The matrices $\mathbb{R}_k$ depends only on electromagnetic field distribution. MODE as an IDE allows possibility to build matrix map for arbitrary fields $\mathbf{E} = (E_x,\, E_y,\, E_s), \mathbf{B} = (B_x,\, B_y,\, B_s)$ that could be expanded in a Taylor series as a functions of s that respects to the length along the design orbit.

IDE DESCRIPTION

In the articles [3,4] a brief introduction to general MODE architecture is presented. In this paper the key possibility and samples will be described.

Standart Electromagnetic Elements

MODE involved a library for the standard electric and magnetic fields distribution. They are drifts, quadrupoles, and sextupoles. To create an element one need to call a function with predefined variable list (see Fig. 1).

```
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.

Arbitrary Field Distribution

Numerical nonlinear matrix integration is a general purpose numerical method. This allows to build matrix maps for any system of ODEs. In MODE one can describe own electromagnetic field by simple declaration of $\mathbf{E}$ and $\mathbf{B}$ fields. It is also necessary to indicate type of design orbit (a straight line or an arc length) and variables of the state vector X. For instance, it is simple to describe parametric map by introducing an additional coordinate.

Fig. 2 shows an example of a quadrupole with parametric voltage on tips declaration. For magnetic field description one only need to define functions B_x, B_y, B_s and identify variable list with additional parameter δV. In field definition one can use any mathematical functions and operations as well as in other part of codes. The resulting map for such ind of field will have order $dim(\mathbf{X}) + 1$. Note that for any

* 05x.andrey@gmail.com
† y.senichev@fz-juelich.de

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-MOPME011

introducing parameter (e.g. δV) the system of ODEs is extended by the equation $\delta V/ds = 0$.

```
curvature = 0;    // 1/R, where R is a radius
variable_list = "x|y|t|px|py|dK|Sx|Sy|Ss|L|dV";
V = 9000;    // voltage on a tip
HGAP = 0.05; // aperture
g = V/HGAP;

V=V*(Polinom(1)+Polinom("dV")); // V=V*(1+dV)

Polinom Bx(x, y, s)      // B = (Bx; By; Bs)
{
    return -g*y;         // return By
}
Polinom By(x, y, s)
{
    return -g*x;         // return Bx
}
Polinom Bs(x, y, s)
{
    return null;         // return Bs = 0
}
```

Figure 2: Arbitrary field distribution.

Code Editor

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

In code researcher can use all standard mathematical 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:

- 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 (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 automatically. 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.

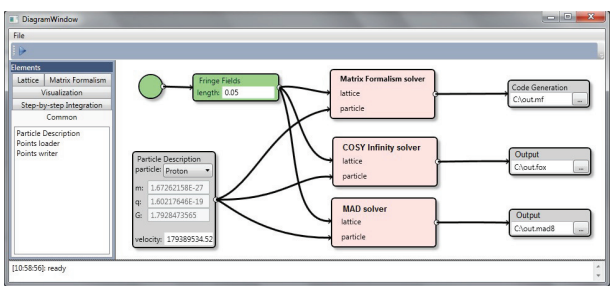

Figure 3: Workflow interface.

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 lens one must only call function with the necessary 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, that includes following components 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.

Twiss Parameters

In MODE one can not only build a map of predefined order for the sequence of elements (standard or arbitrary) but also investigate lattice parameters. It is possible to calculate beta functions, dispersions, betatron oscillations (Q_x, Q_y), momentum compaction factor, and chromaticity. For this calculation linear approach is applied, when a nonlinear matrix map corresponds to the transfer matrix.

All functions and parameters can be visualized (see Fig. 4 and 5). It is also possible to draw a resonance diagram. So one can perform an optimization of lattice taking into account numerical estimations of the functions and parameters mentioned above.

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-MOPME011

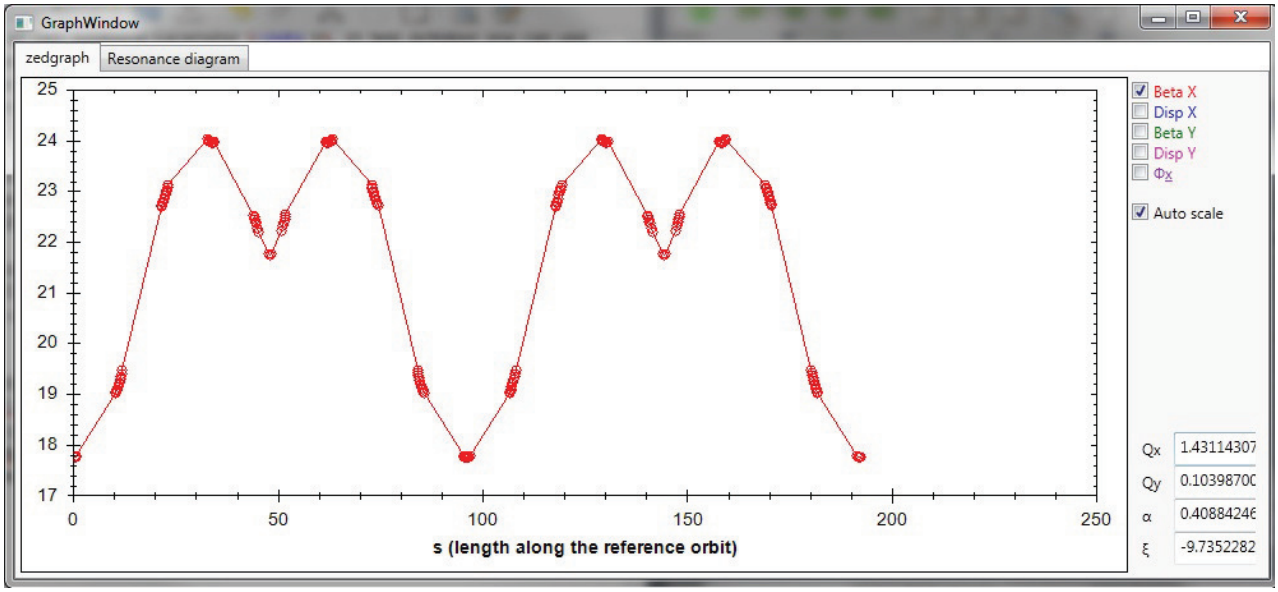

Figure 4: Betatron function.

CONCLUSION

MODE implement nonlinear numerical matrix integration of spin-orbit particle dynamics. A researcher can either us standard elements or introduce arbitrary elements with various electromagnetic field distribution. MODE also allows flexible technique to variable list modification and parametric maps building.

The key idea of MODE is providing tool to nonlinear map building and lattice investigation. Beam dynamic simulation can be performed by simple matrix multiplication and addition. Due to the simplicity of this operation it is possible to translate matrices into the computational code in different programming languages (C++, Fortran, MATLAB, etc.).

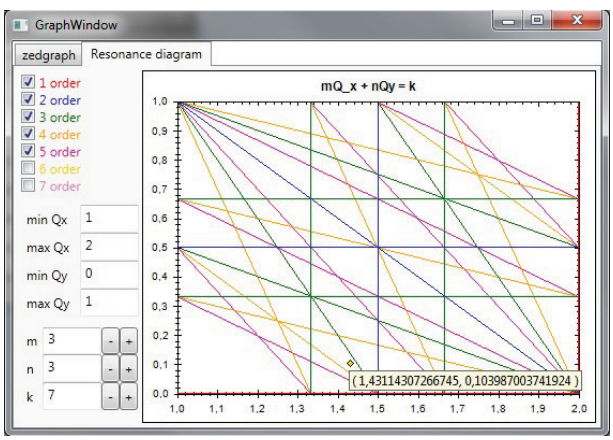

Figure 5: Resonance diagram up to 5th order.

ACKNOWLEDGMENT

The authors would like to thank Yurij Senichev and Serge Andrianov for continues discussion on problems formulation and mathematical models implementation.

REFERENCES

[1] 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.

[2] A. Ivanov, N.Kulabukhova, "An IDE for spin-orbit dynamics simulation," Proceedings of IPAC2013, China, 2013. P.921–923.

[3] 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.

[4] 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,