ISBN: 978-3-95450-223-3 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2019-TUPLO04

THE LATEST CODE DEVELOPMENT PROGRESS OF JSPEC*

H. Zhang#, S. Benson, Y. Zhang, Y. Roblin
Jefferson Lab, Newport News, VA, USA

Abstract

The JLab Simulation Package on Electron Cooling (JSPEC) is an open source software developed at Jefferson Lab for electron cooling and intrabeam scattering (IBS) simulations. IBS is an important factor that leads to the growth of the beam emittance and hence the reduction of the luminosity in a high density ion collider ring. Electron cooling is an effective measure to overcome the IBS effect. Although JSPEC is initiated to fulfil the simulation needs in JLab Electron Ion Collider project, it can be used as a general design tool for other accelerators. JSPEC provides various models of the ion beam and the electron beam. It calculates the expansion rate and simulates the evolution of the ion beam under the IBS and/or electron cooling effect. In this report, we will give a brief introduction of JSPEC and then present the latest code development progress of JSPEC, including new models, algorithms, and the user interface.

INTRODUCTION

JLab simulation package for electron cooling (JSPEC) is an efficient C++ program for intrabeam scattering (IBS) effect and electron cooling simulations. It is developed at JLab to fulfil the requirements of JLab Electron-Ion Collider (JLEIC) [1] cooling scheme and cooler design. It provides various models and tools for IBS expansion rate and/or electron cooling rate calculations and cooling process simulations. JSPEC has been thoroughly benchmarked with BETCOOL [2]. For a typical JLEIC IBS and cooling simulation, the two programs agree and JSPEC has been observed to achieve a noticeable improvement in efficiency. Now JSPEC is being actively used in JLEIC design. JSPEC is open source, with the source code and the documents available on the github repository [3]. A cloud version has been developed by Radiasoft in their SIREPO platform [4]. We have reported the development of JSPEC in the IPAC conference in Busan, Korea, 2016 [5]. In this report, we will concentrate on the latest development of JSPEC, including a turn-by-turn model for IBS and/or cooling process simulation, a model for user-defined arbitrary electron beam, and the input file for JSPEC.

TURN-BY-TURN MODEL

JSPEC originally had the RMS dynamic model and the particle model for IBS and/or electron cooling process simulation. The RMS dynamic model assumes the ion beam always maintains the Gaussian distribution so that the ion beam can be represented by the macroscopic parameters, *i.e.* emittance, momentum spread, and bunch length (for bunched beam). The particle model uses sample particles to represent the ion beam, hence the beam does not necessarily maintain the Gaussian distribution. In each time step, each particle receives a random phase advance for betatron and synchrotron oscillations. The turn-by-turn model is a development of the particle model. Instead of the random phase advance, the betatron and synchrotron motion is simulated by a linear one-turn map, which currently is generated from the tunes, but could be replaced by a high-order transfer map generated by an accelerator design/simulation program, *e.g.* MAD-X [6] and COSY Infinity [7], for more accurate modeling. The algorithm of the turn-by-turn model could be descibed as follows: (1) Create particles w.r.t. the original emittances of the ion beam; (2) Calculate the friction force on each ion, which leads to a momentum change (a kick); (3) Calculate the IBS rate and apply the IBS kick to each ion; (2) Apply the one-turn map on all particles; (5) Emittances are calculated statistically from the 6D phase space coordinates of all the particles; and (6) Repeat from step (2).

Comparing with the other models, the turn-by-turn model is much slower and may not be suitable as a design tool for a long cooling process. But it is considered more fundermental and hence more accurate. It can be used to benchmark the other models.

We have compared the turn-by-turn model with the RMS model, trying to figure out what is the proper particle number and step size for the RMS simulations. Take a typical JLEIC cooling case, perform the simulation of a 10-second cooling using the turn-by-turn model with 10,000 or 100,000 particles. Then repeat the same simulations again using the RMS dynamic model with the step size of 1 second and 10 seconds. Comparing the result, we can see the relative error of the emittances for one time step, listed in Table 1. The accumulated relative error for one hour can be estimated, which is listed in Table 2. We can see that more particles with smaller step

Table 1: Relative Error of Emittance in One Step

Step size (s)	N=10,000	N=100,000
1	1.54×10^{-5}	2.42×10^{-6}
10	3.06×10^{-4}	1.04×10^{-4}

Table 2: Relative Error of Emittance in One Hour

Step size (s)	N=10,000	N=100,000
1	5.70%	0.88%
10	11.65%	3.81%

* Work supported by the Department of Energy, Laboratory Directed Research and Development Funding, under Contract No. DE-AC05-06OR23177.
#hezhang@jlab.org

ISBN: 978-3-95450-223-3 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2019-TUPLO04

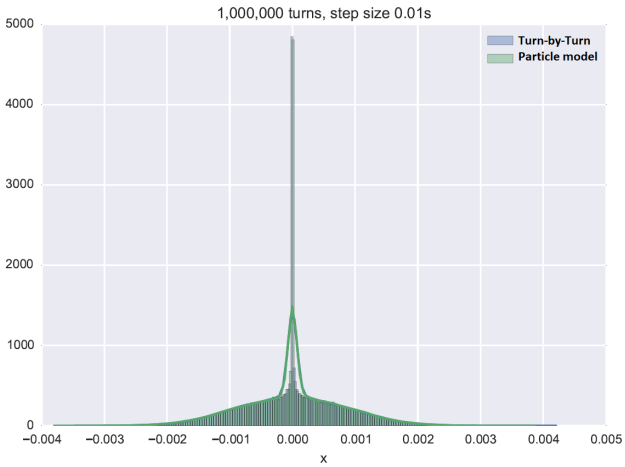

Figure 1: Compare the results by the turn-by-turn model and the particle model.

size lead to a smaller error. The accumulated relative error in one hour is only 0.88% when the particle number is 100,000 and the step size is 1 second, which gives us a guide on how to select the parameters for the RMS simulations.

We have also compared the turn-by-turn model with the particle model, in order to find the proper parameters for the particle model that can simulates non-Gaussian ion beam distribution. The distribution of the ion beam may deviate from the Gaussian distribution under very strong cooling. We run a turn-by-turn simulation of a proton beam under strong cooling for 1,000,000 turns and compare the result with particle model simulation with different step sizes. The distribution of the proton beam is originally Gaussian and will changes into bi-Gaussian due to the strong cooling on the core. We found that when the step size is 0.01 second, the results of the two models are almost identical, as shown in Fig. 1. The difference is minor and acceptable, when the step size is increased to 0.1 second.

USER-DEFINED ARBITRARY ELECTRON BEAM

JSPEC provides various models for the cooling electron beam with regular geometries, such as round, elliptic, or Gaussian DC/bunched beams. JSPEC also allows to use the user-defined electron beam that has no assumptions on the shape and charge distribution. The electron bunch is defined by sample particles with 6D coordinates (x, y, z, v_x, v_y, v_z) saved in an ascii file or a binary file. (x, y, z) is the position of a sample particle in the lab frame. (v_x, v_y, v_z) is the velocity of a sample particle in the beam frame. For the purpose of friction force calculation, one needs to calculate the electron density and temperature around a given ion, for which a tree-based algorithm has been implemented. We first enclose all the electrons inside a large cubic box, and then divide the box into eight small boxes of equal size. The process is repeated until the number of electrons in each box is less than a predetermined number, S. Thus smaller/larger boxes will be generated where there is higher/lower electron density. The relation between the boxes can be represented as a partial tree if we remove all the empty boxes. A 2D example is illustrated in Fig. 2, while it is in principle the same for a 3D case. When an ion is given, we simply need to locate which box it is in and the local electron density and temperature will be calculated using the electrons inside the box. In electron cooling simulations, the electron beam is often assumed unchanged. In such a case, the tree of the electrons only needs to be generated once, which can be reused in the following simulation steps. The efficiency of this algorithm is $O((N_e+N_sN_i)\lg N_e)$, where N_e is the number of the sample electrons, N_i the number of the sample ions and N_s the number of simulation steps. If the change of the electron beam needs to be considered in each step, one needs to generate the tree on each time step and the efficiency of the algorithm will be $O(N_s\,(N_e+N_i)\lg N_e)$.

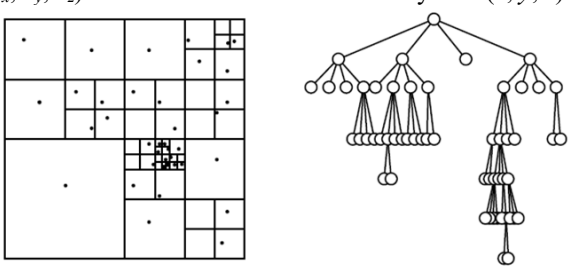

Figure 2: Tree structure of boxes.

There are parameters affecting the accuracy of the model: the sample electron number N_e and the maximum number of sample electrons in a box S. Obviously, a larger N_e is preferred for better representation of the electron beam. But the choice of S is a dilemma. If S is too large, the number of boxes will be too small and the charge density calculation may not be accurate. But if S is too small, the number of electrons in a box is even smaller and the local temperature calculation suffers from statistical error. In Table 3, we check the effect of the value of S to the cooling rate calculation. The first row shows the cooling rate of an ideal Gaussian electron beam. The following row shows the cooling rate calculated by the numerical model with 1,000,000 sample electrons and various S listed in the first column. We can see that $S = 200$ gives the best result. It is also good to see that the result is not very sensitive to the value of S, which means even if S deviates from the best value a little, the final result will not be very bad. This shows the robustness of the model.

Table 3: Cooling Rate (in $10^{-4}s^{-1}$) for Various S

s	R_x	R_y	R_z
N/A	1.57	2.04	2.75
50	1.68	2.20	3.07
100	1.60	2.10	2.89
200	1.57	2.04	2.77
300	1.56	2.02	2.75
400	1.55	2.01	2.73

ISBN: 978-3-95450-223-3 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2019-TUPLO04

JSPEC INPUT FILE

JSPEC takes one input file in the plain text format. An example is shown in Fig. 3. The input file is composed with sections that fall into three categories: scratch section (inside the blue block in Fig. 3), definition section (red block) and operation section (green block). The scratch section is the only section that allows the user to define some variables and do some simple calculations with the variables. Those variables are accessible in the following section. As shown in Fig. 3, two variables "emit_nx" and "emit_ny" are defined in the scratch section, and then they are used to define the normalized emittance of the ion beam in "section_ion". As the name suggests, the scratch section works as your scratch paper. One can have multipole scratch sections in one input file and one can put them anywhere as one likes. The definition category includes all the sections that are used to define the machine, the beams, and the calculation/simulation environments. In Fig. 3, "section_ion", "section_ring", and "section_ibs" are definition sections and they define the ion beam, the ring, and the IBS calculation environment respectively. In each line inside a definition section, the left size to the "=" is a keyword and the right size is the respective value, which should be a number, a variable defined in a previous scratch section, or an expression that can be calculated by the math parser. Although the elements are defined here, they are not created until the corresponding commands are called in the operation section. The operation section is the place to call the operational commands, which can create the accelerator elements, create the beams, carry out the expansion rate calculation or the cooling/IBS dynamic simulation, and/or print out the results to the screen. As seen in Fig. 3, three commands are called in "section_run". The first one creates the ion beam; the second one creates the collider ring; and the third one calculates the IBS expansion rates.

```
section_scratch #scratch section
        emit_nx = 0.5e-6
        emit_ny = emit_nx/5

section_ion #define the ion beam
        charge_number = 1
        mass = 938.272
        kinetic_energy = 100000
        norm_emit_x = emit_nx
        norm_emit_y = emit_ny
        momentum_spread = 0.0008
        particle_number = 0.98e10
        rms_bunch_length = 0.01

section_ring #define the ring
        lattice = MEICColliderRedesign1IP.tfs

section_ibs #define the arguments for IBS calculation
        model = martini
        nu = 100
        nv = 100
#       nz = 40
        log_c = 20.6
        coupling = 0

section_run #start calculation
        create_ion_beam
        create_ring
        calculate_ibs
```

Figure 3: Sample of JSPEC input file.

Run JSPEC in the command line followed by the input file name, the input file will be processed line by line. Anything that follows a "#" is considered as a comment and ignored by the program. Spaces and tabs at the both ends of a line are ignored. Empty lines are ignored, too. There are no requirements on indents. The indents in the example of Fig. 3 are added simply to make it easier to read.

A list of all the sections and the keywords for each section including the meaning and the proper value of the keywords are available in the manual, which is hosted in the github repository [3].

SUMMARY

JSPEC is a program that calculates the instant expansion rate and simulates the evolution of the ion beam under the IBS effect and the electron cooling effect. It is developed at JLab and is being actively used in JLEIC design. A bunch of ion beam models and electron beam models are provided. In this report, we presented the latest updates of JSPEC: a turn-by-turn model for IBS/cooling dynamic simulation, a tree-based model for user-defined electron beam, and the text-based input file. JSEPC is open-source. The source code and the documentation of JSPEC are published on its github repository [3].

ACKNOWLEDGEMENT

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177.

REFERENCES

[1] S. Abeyratne *et al.*, "*MEIC Design Summary*", arXiv:1504.07961, (2015).

[2] I. Meshkov *et al.*, "*BETACOOL Physics Guide*", (2007); http://betacool.jinr.ru

[3] https://github.com/zhanghe9704/electroncooling

[4] http://radiasoft.net

[5] H. Zhang, J. Chen, R. Li, Y. Zhang, H. Huang, and L. Luo, "Development of the Electron Cooling Simulation Program for JLEIC", in Proc. IPAC'16, Busan, Korea, May 2016, pp. 2451-2453. doi:10.18429/JACoW-IPAC2016-WEPMW014

[6] F. Schmidt and H. Grote, "MAD-X -- An Upgrade from MAD8", in Proc. PAC'03, Portland, OR, USA, May 2003, paper FPAG014, pp. 3497-3499.

[7] K. Makino, M. Berz, "*COSY Infinity Version 9*", NIM A, p. 346, (2006).