ISBN: 978-3-95450-245-5 ISSN: 2226-0374 doi:10.18429/JACoW-COOL2023-THPOSRP14

JSPEC: A PROGRAM FOR IBS AND ELECTRON COOLING SIMULATION*

H. Zhang†, Y. Zhang, S.V. Benson, M.W. Bruker
Thomas Jefferson National Accelerator Facility, Newport News, VA, United States

Abstract

JSPEC (JLab Simulation Package on Electron Cooling) is an open-source C++ program developed at Jefferson Lab to simulate the evolution of the ion beam under the intrabeam scattering effect and/or the electron cooling effect. JSPEC includes various models of the ion beam, the electron beam, and the friction force, aiming to reflect the latest advances in the field and to provide a useful tool to the community. JSPEC has been benchmarked against other cooling simulation codes and experimental data. It has been used to support the cooler design for JLEIC, an earlier JLab design for the Electron-Ion Collider. A Python wrapper of the C++ code, pyJSPEC, for Python 3.x environment has also been developed and released. It allows users to run JSPEC simulations in a Python environment and makes it possible for JSPEC to collaborate with other accelerator and beam modeling programs, as well as plentiful Python tools in data visualization, optimization, machine learning, *etc*. A Fortran interface is being developed, aiming at seamless call of JSPEC functions in Fortran. In this report, we introduce the features of JSPEC, with a focus on the latest development, and demonstrate how to use JSPEC, pyJSPEC, and the FORTRAN interface with sample codes.

INTRODUCTION

Intrabeam scattering (IBS) [1] is an effect that may reduce the quality of a high-intensity beam and impair the luminosity of a collider. Electron cooling [2] is an experimentally proven leading method to reduce the ion beam emittance and it is often used to mitigate the IBS effect. JSPEC (JLab Simulation Package on Electron Cooling) is an open-source program [3] developed at Jefferson Lab, which simulates the evolution of the ion beam under the influence of both IBS and electron cooling effects. Originally developed to support the Electron Ion Collider (EIC) design at Jefferson Lab [4], our goal is now to provide a convenient toolkit for the IBS and electron cooling effect simulation to the wider accelerator community. JSPEC includes the most frequently used formulas for the friction forces and variant models of electron/ion beams. The main code is developed in C++ with emphasis on both the validity of the physical models and the efficiency of computation. Most modules in JSPEC supports parallel computing in shared-memory structure using OPENMP. A Python wrapper, pyJSPEC, gives users access to most JSPEC functions in Python 3.x environment allowing JSPEC to work collaboratively with other simulation tools that have a Python interface [5, 6]. Many legacy codes, developed in Fortran, remain widely used in accelerator modelling. In response to this, we are developing a Fortran interface for JSPEC. Our goal is to allow JSPEC functions to be seamlessly called within Fortran, facilitating the modelling of the electron cooling effect in those codes.

FEATURES

The basic feature of JSPEC is to calculate the emittance growth rate of the ion beam under the IBS and/or the electron cooling effect. The rate at time t is defined as $r_i(t) = \frac{1}{\epsilon_i(t)}\frac{\mathrm{d}\epsilon_i(t)}{\mathrm{d}t}$, where $i = x, y, s$, representing the horizontal, vertical, and longitudinal direction, and ϵ_i is the emittance in the respective direction. For the IBS rate, JSPEC provides the Martini model [7], the original Bjorken-Mtingwa model [8] calculated by Nagaitsev's method [9], and the complete Bjorken-Mtingwa model with vertical dispersion and non-relativistic terms included [10]. The electron cooling rate is calculated statistically on a group of sample ions, each receiving a *kick* by the friction force. The rate is calculated as the relative change of the emittance per unit time before and after the kick. JSPEC provides several formulas [11-14] for both the non-magnetized and the magnetized friction force. Using different formulas in the transverse and the longitudinal direction is allowed.

JSPEC also simulates the evolution of the ion beam under the IBS effect and/or the electron cooling effect. The RMS dynamic model represents the ion beam by its macroscopic parameters, *i.e.* the emittances, the momentum spread, and the bunch length (for a bunched beam), calculates the instant expansion rate r at a time t and updates the parameters using $\epsilon_i(t + \Delta t) = \epsilon_i(t)\exp(r_i \Delta t)$ for the time step Δt. The particle model applies kicks due to IBS and electron cooling to sample ions and moves them by a random phase advance for the betatron and synchrotron oscillation in Δt. The turn-by-turn model is similar to the particle model but the betatron and synchrotron motion is modeled by a linear transfer matrix and the simulation is carried out in a turn-by-turn manner.

Recently we have added two new features [15]. One is to treat the cooler as a lengthy element. Previously, JSPEC used a thin lens model for the cooler. The electron beam is assumed constant during the cooling process. In the new model, we cut the cooler into n slices. For ions, each slice works as a drift. The position of an ion changes but the velocity does not when moving through a slice. In the middle of the slice, the ion get a kick due to cooling. For cooling simulations, it is not necessary to consider the motion of individual electrons but rather the properties of the electron beam as a whole. For non-magnetized beam, we calculate the Twiss function β and α at different locations and then

* Work supported by the U.S. Department of Energy, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177.
† hezhang@jlab.org

ISBN: 978-3-95450-245-5 ISSN: 2226-0374 doi:10.18429/JACoW-COOL2023-THPOSRP14

compute the new bunch size, density function, and velocity distribution function. In magnetized cooling, electrons always travel a helical trajectory along the magnetic line. The "matched" β function is constant in the cooler, resulting in a constant bunch size and the elimination of α function. Therefore, the electron beam can still be considered constant in the cooler for magnetized cooling. The other new feature allows the introduction of electron beam horizontal dispersion for non-magnetized cooling, which affects the distribution of cooling between the longitudinal direction and the transverse direction. Due to the dispersion, the density function and the velocity distribution function of the electron beam deviate from the original distribution [16]. JSPEC performs 3D numerical integration using the new distribution function to calculate the friction force. A numerical model without the assumption of initial Gaussian distribution is also developed, based on the arbitrary electron beam model [15] in JSPEC. However, additional testing is needed to better understand the numerical properties of this model.

BENCHMARK

JSPEC has been benchmarked with BETACOOL [11] for various scenarios. The two programs agree well. For the typical simulations we have done for the EIC project, a significant improvement of efficiency has been achieved even without using multiprocessing in JSPEC. Parallel computation will further improve the efficiency.

We also compared JSPEC simulations with experimental data, obtained from the collaboration of Jefferson Lab in the U.S. and Institute of Modern Physics in China from 2016 to 2019 [17]. Figure 1 shows the cooling of the $^{86}Kr^{25+}$ beam with an energy of 5 MeV/nucleon using electron pulses with a length varying from 600 ns to 1000 ns. A longer pulse length means a longer overlap between the two beams and a stronger cooling, which is observed through the larger slope of the plots. The solid lines in the plot are the results from the simulation using the turn-by-turn model. The dots are experimental data. In all the cases, the simulation agrees with the experiment reasonably well.

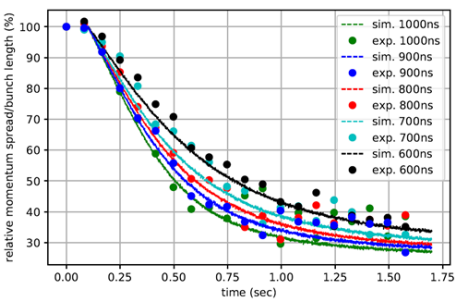

Figure 1: Cooling the $^{86}Kr^{25+}$ beam using pulsed electron beam, simulation (solid lines) and experiments (dots).

USER INTERFACE

JSPEC has been tested on both Windows and Linux systems. It can be run from the command line by typing the executable file name followed by the input file name.

The JSPEC input file is in plain text format. It is composed of a few sections as shown in Fig. 2. Most sections are used to define the elements or to set up models for the effects under study. In these sections, we set values to the keywords, which describe an element or represent parameters of a model/formula. In the last section (section_run), the elements are created and the simulation is carried out. A valid input must include the above two types of sections. There are two optional sections: section_scratch and section_comment. We can define variables and perform simple calculations in section_scratch. The variables can be used in the following sections. In section_comment, we can write a long note that is not suitable for inline comments. For more useful examples, we suggest that the readers check out the *github* repository [3].

```
section_ion #define the ion beam
        charge_number = 1
        mass = 938.272
        kinetic_energy = 3e4
        ...
section_ring #define the ring
        ...
section_cooler #define the cooler
        ...
section_scratch
        m = 938.272
        ke = 3e4
        gamma_ion = ke/m + 1
section_e_beam #use gamma_ion above
        gamma = gamma_ion
        ...
section_ibs #set up IBS calculation
        model = bm
        log_c = 24
        coupling = 0
section_ecool
        ...
section_simulation
        ...
section_run
        create_ion_beam
        create_ring
        create_e_beam
        create_cooler
        run_simulation
```

Figure 2: Structure of JSPEC input.

PYTHON INTERFACE

Python is one of the most popular programming languages in use today. Those who work with Python benefit from a vast array of libraries in data processing, optimization, machine learning, *etc*. pyJSPEC serves as the Python interface for JSPEC, and it has been developed using Pybind11 [18]. pyJSPEC is a library for Python 3.x. Once imported, users can access most functions offered by JSPEC. The Python wrapper enables JSPEC to integrate seamlessly with the rich set of Python libraries, as well as with other accelerator modeling programs that offer Python support. For instance, one could utilize an evolutionary optimizer in conjunction with JSPEC to optimize cooling performance [6]. Figure 3 shows an example on calculating the expansion rate due to both the electron cooling and the IBS effect and carrying out a dynamic simulation using the particle model. The code is easy to understand. Since

ISBN: 978-3-95450-245-5 ISSN: 2226-0374 doi:10.18429/JACoW-COOL2023-THPOSRP14

JSPEC is object-oriented, one first needs to create an element. Once the element is established, one can call its member functions to set up the element's properties or to conduct specific calculations or simulations.

```python
import jspec

# create the ion beam
...
p_beam = jspec.Beam(n_charge, n_mass,
    ke, ex, ey, dp, ds, np)

# create the ring
lat = jspec.Lattice("lattice.tfs")
ring = jspec.Ring(lat, p_beam)

# create the cooler
...
cooler = jspec.Cooler(length, section_number,
    magnetic_field, twiss_beta, twiss_beta,
    dx, dy)

# create electron bunch
...
e_beam = jspec.GaussianBunch(ne, sigma_x,
    sigma_y, sigma_z)
e_beam.set_gamma(gamma)
e_beam.set_tpr(0.5, 0.1)

# calculate electron cooling rate
force_solver = jspec.ForcePark()
n_sample = 40000
ecool_solver = jspec.ECool()
rate = ecool_solver.rate(force_solver, p_beam,
    n_sample, cooler, e_beam, ring)

# calculate IBS rate
ibs_solver = jspec.IBSSolver_BM(log_c)
rate = ibs_solver.rate(lat, p_beam)

# create proton samples
p_samples = jspec.Ions_MonteCarlo(n_sample)
p_samples.set_twiss(cooler)
p_samples.create_samples(p_beam)

# run simulation
...
simulator = jspec.ParticleModel(time, n_step)
simulator.set_ibs(True)
simulator.set_ecool(True)
simulator.run(p_beam, p_samples, cooler,
    e_beam, ring, ibs_solver, ecool_solver,
    force_solver)
```

Figure 3: pyJSPEC sample code.

FORTRAN INTERFACE

In modern proposed colliders like the EIC, high-density ion beams achieve unparalleled luminosity. This introduces the necessity to simulate the IBS effect and electron cooling in conjunction with other collective effects, such as the space charge effect, CSR effect, and more. While the simulation of these collective effects is beyond the purview of JSPEC, there are robust tools available in other accelerator modeling programs. Several legacy programs, developed in FORTRAN, remain in widespread use. These programs, such as BMAD [19], are well-benchmarked, meticulously documented, and furnish many reliable models. Integrating JSPEC with these legacy programs will significantly enhance our capabilities in electron cooling simulation. This realization prompted us to develop a FORTRAN interface for JSPEC. We hope to seamlessly invoke JSPEC functions within FORTRAN. Although this initiative is still underway, we have already ported the majority of data types from JSPEC into FORTRAN, validating that our ultimate objective is feasible. Figure 4 offers a sample code that demonstrates the computation of the cooling rate on a proton beam using JSPEC in FORTRAN.

```fortran
program main
    use jspec
    use iso_c_binding
    implicit none

    type(Beam) :: my_beam
    type(Lattice) :: my_lattice
    type(Ring) :: my_ring
    type(Cooler) :: my_cooler
    type(FrictionForceSolver) :: my_force_solver
    type(EBeam) :: my_ebeam
    type(ECoolRate) :: rate_ec
    real(8) :: rx, ry, rs

    ! Define variables
    integer(c_int) :: charge = 1
    real(c_double) :: mass = 938.272
    ...

    ! Create ion beam
    my_beam = create_beam(charge, mass, ke, &
            ex, ey, dp, ds, np)

    ! Create the lattice from some file
    my_lattice = create_lattice("lattice.txt")

    ! Create a ring
    my_ring = create_ring(my_lattice, my_beam)

    ! Create a cooler
    my_cooler = create_cooler(length, n_section, &
            mag_field, twiss_beta, twiss_beta)

    ! Create friction force solver
    my_force_solver = create_force_solver(PARKHOMCHUK)

    ! Create electron beam with Gaussian distribution
    my_ebeam = create_gaussian_bunch(ne, sigma_x, &
            sigma_y, ds)
    call ebeam_set_gamma(my_ebeam, gamma)
    call ebeam_set_temperature(my_ebeam, tmp_tr, tmp_l)

    ! Calculate the cooling rate rx, ry, rs
    rate_ec = create_ecool_rate_calculator()
    call ecool_rate(rate_ec, my_force_solver, &
            my_beam, n_sample, my_cooler, &
            my_ebeam, my_ring, rx, ry, rs)
end program main
```

Figure 4: FORTRAN interface sample code.

SUMMARY

JSPEC is an open-source program for IBS and electron cooling simulations developed at Jefferson Lab. The source code, manuals, and examples are provided [3]. A Python wrapper, pyJSPEC, has been developed and most functions from JSPEC has been ported to Python 3.x environment. pyJSPEC enables us to tap into the vast array of Python libraries for cooling scheme design and study. Additionally, a FORTRAN interface is under development. This interface aims to make JSPEC functions accessible in FORTRAN, allowing for the simulation of electron cooling in conjunction with other collective effects.

ACKNOWLEDGMENT

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.

ISBN: 978-3-95450-245-5 ISSN: 2226-0374 doi:10.18429/JACoW-COOL2023-THPOSRP14

REFERENCES

[1] A. Piwinski, “Intra-beam-scattering”, in *Proc. 9th Int. Conf. on High Energy Accelerators*, Stanford, CA, USA, May 1974, pp. 405-409. doi: 10.5170/CERN-1992-001.226

[2] Ya. Derbenev, "Theory of electron cooling", arXiv:1703.09735. doi:10.18429/10.48550/arXiv.1703.09735

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

[4] S. Abeyratne *et al.*, “MEIC Design Summary”, arXiv:1504.07961. doi:10.48550/arXiv.1504.07961

[5] pyJSPEC, https://github.com/zhanghe9704/jspec2-python/

[6] H. Zhang *et al.*, “pyJSPEC – A Python module for IBS and electron cooling”, in *Proc. 5th North American Particle Accelerator Conf. (NAPAC2022)*, Albuquerque, NM, USA, Aug. 2022, pp. 672-674. doi: 10.18429/JACoW-NAPAC2022-WEPA24

[7] M. Martini, “Intrabeam scattering in the ACOL-AA machines,” Rep. CERN-PS-8-4-9-AA, CERN, 1984.

[8] J. Bjorken and S. Mtingwa, “Intrabeam scattering,” *Part. Accel*. 13, pp. 115-143, 1982.

[9] S. Nagaitsev, “Intrabeam scattering formulas for fast numerical evaluation,” *Phys. Rev. ST Accel. Beams*, vol. 8, p. 064403, 2005. doi: 10.1103/PhysRevSTAB.8.064403

[10] F. Zimmermann, “Intrabeam scattering with non-ultrarelativistic corrections and vertical dispersion for MAD-X,” Rep. CERN-AB-2006-002, CERN, 2005.

[11] I. Meshkov *et al*., “BETACOOL physics guide”, Joint Institute for Nuclear Research, Dubna, Russian Federation, 2007.

[12] I. Meshkov, “Electron cooling: status and perspectives,” *Phys. Part. Nucl.*, vol. 25, pp. 631-661, 1994.

[13] V. Parkhomchuk, “New insights in the theory of electron cooling,” *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 441, pp. 9-17, 2000. doi: 10.1016/S0168-9002(99)01100-6

[14] Y. Derbenev and A. Skrinsky, “The effect of an accompanying magnetic field on electron cooling,” *Part. Accel.*, vol 8, pp. 235-243, 1978.

[15] H. Zhang *et al*., "Latest updates on JSPEC – an IBS and electron cooling simulation program", in Proc. 14th International Particle Accelerator Conf. (IPAC2023), Venice, Italy, May. 2023, pp. 2253-2256. doi:10.18429/JACoW-IPAC2023-TUPM030

[16] M. Blaskiewicz., "Dispersion and electron cooling", Brookhaven National Lab, Upton, NY, USA, Rep. BNL-210932-2019-TECH, 2019.

[17] M. Bruker *et al.*, “Demonstration of electron cooling using a pulsed beam from an electrostatic electron cooler,” *Phys. Rev. Accel. Beams*, vol. 24, p. 012801, 2021. doi: 10.1103/PhysRevAccelBeams.24.012801

[18] Pybind11, https://pybind11.readthedocs.io

[19] BMAD, https:// www.classe.cornell.edu/bmad/