# JSPEC v. 2.0.0 User Manual

## Using the text-based user interface

### Run JSPEC with the input file

Put the input file in the same directory with the JSPEC program and run JSPEC as :

> jspec.exe *inputfile*

### Format of the input file

The input file is a plain text file and it will be parsed by the program line by line. Each command or expression should occupy a separate line.  Comments start with "#". Everything behind the "#" in the line will be ignored by the program. Blank lines, white spaces and tabs are also ignored. The input file is NOT case-sensitive. From version 2.0, it is allowed to break a long line into several lines. Any line ending with "&&" is considered unfinished and will be combined with the following line before processing. All white spaces at the both ends of the two lines and the "&&" will be trimed before combining them. [1] Since the program check for comments before check for "&&", the "&&" after "#" will be ignored as a part of the comment and will not connect the following line. However, if after chopping off the comments and the whitespace, the line ends with "&&", then it will be connected to the following line.

The input file is organized by various sections. All the sections fall into four different categories: (1) scratch section, (2) comment section (3) definition sections and (4) operation section.  All the sections with the respective categories and usages are listed in the following table.

| Section name | Category | Usage |
|---|---|---|
| section_scratch | scratch | Define variables and do calculations with the variables. The variables defined in this section can be used in definition sections. |
| section_comment | comment | Anything in this section will be treated as comments. This is provided in case one wants to write a long comment. |
| section_ion | definition | Set parameters for the ion beam |
| section_ring | definition | Set parameters for the ion ring |
| section_e_beam | definition | Set parameters for the cooling electron beam |
| section_cooler | definition | Set parameters for the cooler |
| section_ibs | definition | Set parameters for IBS rate calculation |
| section_ecool | definition | Set parameters for electron cooling rate calculation |
| section_luminosity | definition | Set parameters for luminosity calculation |
| section_run | operation | Create the objectives (ion beam, ion ring, electron beam, cooler) and perform the calculation and/or the simulation. |
| ===end=== | special keyword | Anything below this line will be ignored. |
| ========= | special keyword | Any line of nine or more consecutive "=" is considered then end of the script. Anything below this line will be ignored. |

The input file starts with a section by calling the section name. Once a section name is called, the respective section is created, and this section ends when another section name is called or when the input file ends. Sections can be repeated called and the latter one overwrite the previous ones. But if a parameters is not set again in the latter one, its value remains.

A special keyword is "===end===". If a line only contains this keyword, which could be followed by comments, anything below this line will be ignored. This keyword is not required to end the program. Without this keyword, the program will process the input script file till the last line. This keyword is provided to bring some convenience to users who want to record the results or add long comments at the end of the input file.

The following example includes three different sections in three different categories.

```
section_scratch #scratch section
    m = 938.272
    ke = 8000
    gamma = ke/m + 1
    print gamma
    list_const
section_e_beam #definition section, define the parameters for electron beam
    gamma = gamma
    tmp_tr = 0.1
```

```
    tmp_l = 0.1
    shape = dc_uniform
    radius = 0.004
    current = 2
section_run #operation function
    create_e_beam
```

The first section is a scratch section. In this section, three variables, m, ke, and gamma, are defined. The values of m and ke are assigned, and gamma is calculated from ke and m. The calculation is supported by the math parser, muParser. Fundamental calculations and functions are supported, including summation, subtraction, multiplication,division, square root, exponential function, etc. For more details about the muParser, please refer to http://beltoforion.de/article.php?a=muparser . The command "print gamma" will print the value of gamma to the screen. The following command will show a list of all the constant variables supported by the scratch section on the screen. All the constant variables and their values are listed in the following table.

| Constant | Value | Meaning |
|---|---|---|
| k_c | 299792458.0 | speed of light, in m/s |
| k_e | 1.602176565E-19; | Elementary charge, in C |
| k_pi | 3.1415926535897932384626 | $\pi$ |
| k_u | 931.49406121 | Atomic mass unit, in MeV/c^2 |
| k_me | 0.510998928 | Electron mass, in MeV/c^2 |
| k_ke | 8.9875517873681764E+9 | Coulomb's constant, in N*m^2/C^2 |

The second section is a definition section, which sets the parameters for the cooling electron beam. In all the expressions in this section, the left side of the "=" sign is a keyword in section_e_beam, which corresponds to a parameter of the electron beam, and the right side is the valued assign to the keyword (the parameter). The first expression in this section is "gamma = gamma".[2] The left gamma is a keyword, which represents the Lorentz factor of the electron beam. The right gamma is the variable defined in the above scratch section. This expression assigns the value of the scratch variable gamma to the keyword gamma. Please note that a scratch variable can be used in other sections to set the value for a keyword, but a keyword cannot be used in the same way. A keyword should always be on the left side of the "=" sign. This is the most important difference between a scratch variable and a keyword. The following expressions assign values for other parameters of the electron beam, which are the transverse temperature, the longitudinal temperature, the shape, the radius and the current of the electron beam respectively. Depending on the shape of the electron, various parameters need to be set. In this example, one needs to set the radius and the current for a uniform DC electron beam. Other supported shapes and the related parameters (keywords) can be found in the lists in the next chapter.

The third section is the operation section. In the operation section, one can create the objects of the elements, calculate the expansion rate and perform the simulation. In this example, we create an object of the electron beam that has been defined in the above definition section. Please note that the definition section only records the values of the parameters, an element will not be created until the respective command is called in the operation section. For more commands supported in the operation section, please check out the list in the next chapter.

## IBS Expansion Rate Calculation

To calculate the IBS expansion rate, one needs to define the ion beam and the ring. Then set the parameters for IBS rate calculation. Finally, in the operation section create the ion beam and the ring, and call the command to calculate the IBS expansion rate.

```
section_ion     # Define the ion beam
    ......
section_ring    # Define the ring
    ......
section_ibs     # Set parameters for IBS rate calculation
    ......
section_run
    create_ion_beam     # Create the ion beam
    create_ring         # Create the ring
    calculate_ibs       # Calculate the IBS rate
```

To calculate the total expansion rate, which is the summation of the IBS expansion rate and the electron cooling rate, one can call the command "total_expansion_rate" in section_run.

## Cooling Rate Calculation

To calculate the cooling rate, one needs to define the ion beam, the ring, the electron beam and the cooler. Then set the parameters for cooling rate calculation. Finally,  in the operation section create all the related elements aforementioned and call the command to calculate the cooling rate.

```
section_ion     # Define the ion beam
    ......
section_ring    # Define the ring
    ......
section_e_beam  # Define the electron beam
    ......
sectoin_cooler  # Define the cooler
    ......
section_ecool   # Set the parameters for the electron cooling rate calculation
    ......
section_run
    create_ion_beam     # Create the ion beam
    create_ring         # Create the ring
    create_e_beam       # Create the electron beam
    create_cooler       # Create the cooler
    calculate_ecool     # Calculate the electron cooling rate
```

## Calculate and output the friction force

When calculating the cooling rate, the friction force on all sample ions are calculated. To output the friction force to a file, one can set the paramter "force_output" to be true  in the section_ecool. The results will be saved in a file named as "Friction_force_on_ions_YYYY-MM-DD-HH-MM-SS.sdds". The default value of this parameter is false and it is recommanded to leave the value as false in simulations when the rate is calculated repeatedly.

It is possible to calculate and save the friction force in a user-defined paramter (longitudinal momentum spread and transvere angle) range without calculating the cooling rate using the command "calculate_friction_force".

```
section_ion     # Define the ion beam
    ......
section_ring    # Define the ring
    ......
section_e_beam  # Define the electron beam
    ......
sectoin_cooler  # Define the cooler
    ......
section_ecool   # Set the parameters for the electron cooling rate calculation
    force_formula = PARKHOMCHUK # Choose the force formula
    limit_momentum_spread = 0.001 # Set the limit of the longitudinal momentum
spread
    limit_angle = 0.0002 # Set the limit of the transverse angle in the lab
frame
    n_l = 10 # Set the grid number (2*n_l+1) for the longitudinal momentum
spread
    n_tr = 10 # Set the grid number (n_tr+1) for the transverse angle
    n_l = 1
#   electron_density = 8.0416806140e+14 # The electron density in electron beam
frame
section_run
    create_ion_beam     # Create the ion beam
    create_ring         # Create the ring
    create_e_beam       # Create the electron beam
    create_cooler       # Create the cooler
    calculate_friction_force        # Calculate and save the friction force
```

In the above exmaple, JSPEC will calculate the friction force on ions with the longitudinal momentum spread from [-0.001, 0.001] with a step size of 0.0001 and the transverse angle from [0, 0.0002] with a step size of 0.00002. If the electron_density is set, the value will be used in friction force calculation on all particles. If not as in the above exmaple, the density will be calculated at the center of the electron beam defined in the section_e_beam. (If a position shift or a velocity shift between the electron beam and the ion beam is defined, it will be included in the friction force calculation. ) This approach means we practically assumes all the ions are in the same position with different momentums and can be used to study how the friction force is affected by the velocity of the ions.

## Simulation

One can simulate the evolution of the ion beam under the IBS effect and/or electron cooling effect during a predetermined time. The emittances, momentum spread, bunch length (for bunched ion beam), and the total expansion rate in all the three dimensions will be outputted into a text file. If desired, the coordinates of all the ion samples can also be saved into files. These parameters are set in section_simulation, and the simulation starts when the command "run_simulation" is called in section_run.

```
section_simulation  # Set the parameters for the simulation
    ......
section_run
    run_simulation  # Start simulation
```

## Luminosity calculation

To calculate the luminosity, one needs to define the particle number and the rms size of the two colliding particles, the center-to-center distance between the two beams, and the colliding frequency. Instead of giving the rms size of the beams, one can define the geometrical emittance and the beta function at the collision point of them, which may be convenient in many cases. An example is given as follows.

```
section_luminosity
    distance_x = 1e-3
    distance_y = 1e-6
    particle_number_1 = 1e7
    particle_number_2 = 1e10
    frequency = 1000
    bet_x_1 = 0.01
    bet_y_1 = 0.01
    bet_x_2 = 0.01
    bet_y_2 = 0.01
    geo_emit_x_1 = 1e-6
    geo_emit_x_2 = 4e-7
    geo_emit_y_1 = 1e-6
    geo_emit_y_2 = 4e-7
    use_ion_emittance = false

section_run
    calculate_luminosity
```

If one wants to use the ion beam defined in the cooling simulation in the luminosity calculation, the parameter use_ion_emittance should be set to true. ( The default value of it is true.) Then the program will use the geometrical emittance of the ion beam to set up the first colliding beam. Please note one has to create the ion beam before the luminosity calculation.

```
section_ion #define the ion beam
    ...

section_luminosity
    distance_x = 0
    distance_y = 0
    particle_number_1 = 1e7
    particle_number_2 = 1e10
    frequency = 1000
    bet_x_1 = 0.01
    bet_y_1 = 0.01
    bet_x_2 = 0.01
    bet_y_2 = 0.01
    geo_emit_x_2 = 4e-7
    geo_emit_y_2 = 4e-7
    use_ion_emittance = true

section_run
```

```
    create_ion_beam
    calculate_luminosity
```

If one wants to calculate the instant luminosity during the simulation, one should set the calc_luminosity parameter in section_simulation to be true.

```
section_luminosity
    ...

section_simulation
    calc_luminosity =  true
    ...

section_run
    run_simulation
```

# List of sections, keywords, and commands

**section_scratch**

| Keywords | Meaning |
|---|---|
| list_var | list all the variables that has been defined. |
| list_const | list all the constants. |
| list_exp | list all the expression. |
| print | Use this command in format "print x" and it will print the value of the variable x in the screen. |
| printstr | Use this command in format "printstr string" and it will print the "string" in the screen. |
| save | Use this command in format "save var" and it will save the value of "var" in a file named as JSPEC_SAVE_YYYY_MM_DD_HH_MM_SS.txt as "var = the value of var". For each run, only one file will be created bye the first save command even if multiple save commands are used. All the following save commands write to the file. If the file already exists, results will be appended to it. |
| savestr | Use this command in format "savestr str" and it will write the "str" to the file. If the output file does not exist, it will create it in the same way as the "save" command does. |
| append | Use this command in format "append var" and it will save the value of "var" to the end of the input script file. If it is the first time to call it or "appendstr", it will write "=========" and the date and time to the end of the input script file. |
| appendstr | Use this command in format "appendstr str" and it will write the "str" to the end of the input script file. If it is the first time to call it or "append", it will write "=========" and the date and time to the end of the input script file. |

The following keywords records the results from the previous computation. They can be used to set up the value for the following computation or to display the results onto the screen.

| Keywords | Meaning |
| --- | --- |
| vl_emit_nx | horizontal normalized emittance. |
| vl_emit_ny | vertical normalized emittance. |
| vl_momentum_spread | The momentum spread. |
| vl_bunch_length | The rms bunch length for a bunched ion beam. The value is zero for a coasting ion beam. |
| vl_rate_ibs_x | horizontal ibs expansion rate. |
| vl_rate_ibs_y | vertical ibs expansion rate. |
| vl_rate_ibs_s | longitudinal ibs expansion rate. |
| vl_rate_ecool_x | horizontal electron cooling rate. |
| vl_rate_ecool_y | vertical electron cooling rate. |
| vl_rate_ecool_s | longitudinal electron cooling rate. |
| vl_rate_total_x | total expansion rate in the horizontal direction. |
| vl_rate_total_y | total expansion rate in the vertical direction. |
| vl_rate_total_s | total expansion rate in the longitudinal direction |
| vl_t | time |

**section_ion**

| Keywords | Meaning |
| --- | --- |
| charge_number | Number of the charges of the ion |
| mass | Mass in [MeV/$c^2$] of the ion |
| kinetic_energy | Kinetic energy in [MeV] of the ion |
| norm_emit_x | Normalized horizontal emittance in [m*rad] of the ion beam |
| norm_emit_y | Normalized vertical emittance in [m*rad] of the ion beam |
| momentum_spread | Momentum spread of the ion beam |
| particle_number | Total particle number for coasting ion beam or the particle number of one bunch for bunched ion beam. |
| rms_bunch_length | RMS bunch length for bunched ion beam in [m] |

**section_ring**

| Keywords | Meaning |
| --- | --- |
| lattice | The name of the file that saves the lattice. This file should be in the MAD X output format (.tfs). |
| qx | Transverse betatron tune |
| qy | Vertical betatron tune |
| qs | Synchrotron tune |
| gamma_tr | Transition gamma |
| rf_v | Voltage of the RF cavity in [V] |
| rf_h | Harmonic number |
| rf_phi | RF phase in [2$\pi$] |

**section_cooler**

| Keywords | Meaning |
| --- | --- |
| length | Length of the cooler in [m] |
| section_number | Number of the coolers |
| magnetic_field | Magnetic field in [T] |
| bet_x | Beta function in horizontal direction in [m] |
| bet_y | Beta function in vertical direction in [m] |
| disp_x | Dispersion in horizontal direction in [m] |
| disp_y | Dispersion in vertical direction in [m] |
| alpha_x | Alpha in horizontal direction |
| alpha_y | Alpha in in vertical direction |
| disp_dx | Derivative of the dispersion in horizontal direction |
| disp_dy | Derivative of the dispersion in vertical direction |
| pipe_radius | radius of the cooler vacumn chamber in [m]. Set it if electron edge effect needs to be considered. |

**section_e_beam**

| Keywords | Meaning |
|---|---|
| gamma | Lorentz factor gamma for the cooling electron beam |
| tmp_tr | Transverse temperature in [eV] |
| tmp_l | Longitudinal temperature in [eV] for the cooling electron beam |
| shape | Electron beam shape. Choose from dc_uniform, bunched_gaussian, bunched_uniform, bunched_uniform_elliptic, dc_uniform_hollow, bunched_uniform_hollow, bunched_user_defined. |
| radius | Radius of dc_uniform or bunched_uniform electron beam in [m]. |
| current | Current of dc_uniform or bunched_uniform electron beam. For bunched_uniform beam, set the current as if it is a dc_uniform beam in [A]. |
| length | Length of the bunched_uniform electron beam in [m]. |
| sigma_x | RMS size in horizontal direction of bunched_gaussian electron beam in [m]. |
| sigma_y | RMS size in vertical direction of bunched_gaussian electron beam in [m]. |
| sigma_z | RMS bunch length of bunched_gaussian electron beam in [m]. |
| sigma_dx | RMS angle in horizontal direction. Instead of directly defining the temperatures, one can define the temperature of a Gaussian bunch with the three parameters sigma_dx, sigma_dy, and sigma_dpp. When they are defined, temperatures are igonred. |
| sigma_dy | RMS angle in vertical direction.Instead of directly defining the temperatures, one can define the temperature of a Gaussian bunch with the three parameters sigma_dx, sigma_dy, and sigma_dpp. When they are defined, temperatures are igonred. |
| sigma_dpp | momentum spread in longitudinal direction. Instead of directly defining the temperatures, one can define the temperature of a Gaussian bunch with the three parameters sigma_dx, sigma_dy, and sigma_dpp. When they are defined, temperatures are igonred. |
| rh | Length of the semi-axis in horizontal direction in [m]. |
| rv | Length of the semi-axis in vertical direction in [m]. |
| r_inner | Inner radius of a hollow beam in [m] |
| r_outter | Outter radius of a hollow beam in [m] |
| particle_file | Name of the file that saves the particles if the beam shape is defined as "bunched_user_defined" |
| total_particle_number | Total number of particles to load from the user-provided file |

| Keywords | Meaning |
| --- | --- |
| box_particle_number | Maximum number of particles in each childless box when constructing the tree structure. Default is 200. |
| line_skip | Number of lines to skip when loading particles from the user-provided text file. |
| vel_pos_corr | Whether to consider the correlation between the velocity and the position. Default is false. |
| binary_file | Whether the user-provided file is in binary format. Default is false, which means a text file. |
| buffer_size | Buffer size when loading particles from the user-provided binary file. |
| multi_bunches | If true, use multple electron bunches to cool one ion beam. The electron bunches are assumed to be the same except that their positions are different. One should set at leat one of list_cx, list_cy, and list_cz for the centers of the electron bunches. |
| list_cx | When multi_bunches to be true, use list_cx to set the horizontal coordinates of the electron bunches. The format should be integer,double,double, ... The first number should be an integer, which tells how many horizontal coordinates (the double flouting numbers) follows it. All the numbers are seperated by coma. |
| list_cy | When multi_bunches to be true, use list_cy to set the vertical coordinates of the electron bunches. The format is the same as that of list_cx. |
| list_cz | When multi_bunches to be true, use list_cz to set the longitudinal coordinates of the electron bunches. The format is the same as that of list_cx. |
| p_shift | True: centers of the electron beam and the ion beam do not overlap. |
| v_shift | True: velocities of the electron beam and the ion beam do not equal. |
| cv_l | An additional longitudinal velocity of the electron bunch. |
| rise_time | rising time of the electron bunch, in [s], use it when considering longitudinal kick due to the electron bunch edge effect |
| fall_time | falling time of the electron bunch, in [s], use it when considering longitudinal kick due to the electron bunch edge effect |

**section_ibs**

| Keywords | Meaning |
| --- | --- |
| nu | Set the grid number in horizontal direction for the 3D integrationin Martinit model. |
| nv | Set the grid number in vertical direction for the 3D integrationin Martinit model. |
| nz | Set the grid number in longitudinal direction for the 3D integration in Martinit model. Set the integration step number in BMC/BMZ model. |
| log_c | Coulomb logarithm. If log_c is set, then the integration in the longitudinal direction is replaced by the Coulomb logarithm. Thus the parameter nz is ignored. |
| coupling | Transverse coupling rate, ranging from 0 to 1. |
| factor | Scale factor for the upper bound of the integration in BMC/BMZ model. |
| model | Model for IBS expansion rate calculation: Martini, BM (Bjorken-Mtingwa model ignoring vertical dispersion and some small terms), BMC/BMZ (Bjorken-Mtingwa model including all terms). BMC and BMZ use different ways to calculate the integrations, but they are the same model. |

**section_ecool**

| Keywords | Meaning |
| --- | --- |
| sample_number | Number of the sample ions. |
| force_formula | Choose the formula for friction force calculation. Now support four formulas for non-magnetized cooling force ("NONMAG_DERBENEV", "NONMAG_MESHKOV", "NONMAG_NUM1D", and "NONMAG_NUM3D") and the Parkhomchuk formula for magnetized cooling force. |
| tmp_eff | Set the effective temperature for parkhomchuk formula. The value should NOT be negative. Setting this parameter makes the "v_eff" be zero. |
| v_eff | Set the effective velocity for parkhomchuk formula. Setting this parameter make the "tmp_eff" be zero. |
| smooth_rho_max | Use the formula that has a smooth dependence on ion velocity to calculate the maximum impact parameter for non-magnetized friction force. |
| use_mean_rho_mean | Use the mean minimal impact parameter to calculate the Coulomb logrithm in the 3D numerical formula for non-magnetized friction force. |
| use_gsl | Use gsl integrator to perform the 3D numerical integration for non-magnetized friction force. If set false, one can perform the 3D integration in a regular grid, which could be faster with a little sacrifice on accuracy. |
| n_tr | Usage scenario 1: Set the number of grid for the 3D integration in a regular grid when calculating non-magnetized force, when use_gsl is false. Usage scenario 2: Use together with the command "calculate_friction_force" and set the grid of transverse angle from 0 to limit_angle with a step size of limit_angle/n_tr. |
| n_l | Usage scenario 1: Set the number of grid for the 3D integration in a regular grid when calculating non-magnetized force, when use_gsl is false. Usage scenario 2: Use together with the command "calculate_friction_force" and set the grid of longitudinal momentum spread from -limit_momentum_spread to limit_momentum_spread with a step size of limit_momentum_spread/(2*n_l). |
| n_phi | Set the number of grid for the 3D integration in a regular grid when calculating non-magnetized force, when use_gsl is false. |
| force_output | Set whether to output the friction force on each ion when calculating the electron cooling rate. Default value is false. When set true, the friction force on each ions will be saved in the file named as "Friction_force_on_ions_YYYY-MM-DD-HH-MM-SS.sdds". |

| Keywords | Meaning |
| --- | --- |
| limit_angle | Use together with the command "calculate_friction_force" and set the limit of the transverse angle as [0, limit_angle]. |
| limit_momentum_spread | Use together with the command "calculate_friction_force" and set the limit of the longitudinal momentum spread as [-limit_momentum_spread, limit_momentum_spread]. |
| electron_density | Use together with the command "calculate_friction_force" and set electron density in the electron beam frame in [1/m^3]. Without setting this parameter, the electron density will be calculated at the center of the user-defined electron beam in section_e_beam. (If a position shift between the electron beam and the ion beam is defined, it will be included in the density calculation. ) |

**section_luminosity**

| Keywords | Meaning |
| --- | --- |
| distance_x | Horizontal distance between the centers of the two colliding beam, in [m]. |
| distance_y | Vertical distance between the centers of the two colliding beam, in [m]. |
| particle_number_1 | Particle number of the first colliding beam. |
| particle_number_2 | Particle number of the 2nd colliding beam. |
| frequency | Colliding frequency, in [1/s]. |
| bet_x_1 | Horizontal beta function of the first colliding beam at the colliding point, in [m]. |
| bet_y_1 | Vertical beta function of the first colliding beam at the colliding point, in [m]. |
| bet_x_2 | Horizontal beta function of the second colliding beam at the colliding point, in [m]. |
| bet_y_2 | Vertical beta function of the second colliding beam at the colliding point, in [m]. |
| beam_size_x_1 | Horizontal rms size of the first colliding beam, in [m]. |
| beam_size_y_1 | Vertical rms size of the first colliding beam, in [m]. |
| beam_size_x_2 | Horizontal rms size of the second colliding beam, in [m]. |
| beam_size_y_2 | Vertical rms size of the second colliding beam, in [m]. |
| geo_emit_x_1 | Geometrical horizontal emittance of the first colliding beam, in [m*rad]. If the beam size is given, this parameter is ignored. |
| geo_emit_y_1 | Geometrical vertical emittance of the first colliding beam, in [m*rad].If the beam size is given, this parameter is ignored. |
| geo_emit_x_2 | Geometrical horizontal emittance of the second colliding beam, in [m*rad].If the beam size is given, this parameter is ignored. |
| geo_emit_y_2 | Geometrical vertical emittance of the second colliding beam, in [m*rad].If the beam size is given, this parameter is ignored. |
| use_ion_emittance | Whether to use the ion beam emittance to set up the first colliding beam: yes (true) or no (false). The default value is true. When the value is true, parameters of the beam size and the emittance of the first colliding beam is ignored and the ion beam should be defined and created before the luminosity calculation. |

**section_simulation**

| Keywords | Meaning |
| --- | --- |
| time | Total time to simulate, in [s]. |
| step_number | Total number of steps. The time interval of each step is time/step_number. |
| sample_number | Number of the sample ions. The parameter must be set when using the Particle model to simulate the IBS expansion process without cooling. When setting this parameter with cooling effect, the "sample_number" parameter in the "section_ecool" will be overwritten by this value. |
| ibs | Choose to simulate the IBS effect or not by setting the value as "true" or "false". |
| e_cool | Choose to simulate the electron cooling effect or not by setting the value as "true" or "false". |
| model | "RMS" or "Particle" model to choose for the simulation. |
| output_file | Output file name. Default value is "output_" followed by the input script file name. |
| output_interval | The interval of steps to write into the output file. Default is one. |
| save_particle_interval | The interval of steps to save the 6D coordinates of the ions. No saving if the value is less than zero. Default is -1. This is only useful when using the Particle model in simulations. |
| ref_bet_x | TWISS parameters for the reference point. Only needed when the "model beam" method is selected and the electron cooling effect is not included in the simulation. |
| ref_bet_y | Same as above. |
| ref_alf_x | Same as above. |
| ref_alf_y | Same as above. |
| ref_disp_x | Same as above. |
| ref_disp_y | Same as above. |
| ref_disp_dx | Same as above. |
| ref_disp_dy | Same as above. |
| fixed_bunch_length | Maintain a constant ion bunch length. Default is false. |
| reset_time | Whether to reset the starting time to zero (value: true) or use the final time from the previous simulation (value: false). |
| overwrite | Whether overwrite the output file is it exists. The default value is true. If the value is false, a new output file will be generated. The name of the new file is created by adding a number before the specific file name. |

| Keywords | Meaning |
| --- | --- |
| calc_luminosity | Whether to calculate the luminosity during the simulation: yes (true) or no (false). The default value is false. |
| edge_effect | Considering the longitudinal kick due to the electron bunch edge effect is the value is true. |

**section_run**

| Keywords | Meaning |
| --- | --- |
| create_ion_beam | Create the ion beam. |
| create_ring | Create the ring. Must create the ion beam before calling this command. |
| create_e_beam | Create the electron beam |
| create_cooler | Create the cooler. |
| calculate_ibs | Calculate the IBS rate and output to the screen. Must create the ion beam and the ring before calling this command. |
| calculate_ecool | Calculate the electron cooling rate and output to the screen. Must create the ion beam, the ring, the electron beam, and the cooler before calling this command. |
| calculate_luminosity | Calculate the luminosity, in [1/s * 1/cm^2] |
| calculate_friction_force | Calculte the friction force in the user-defined parameter domain and save the result in the file named as "Friction_force_on_ions_YYYY-MM-DD-HH-MM-SS.sdds". |
| total_expansion_rate | Calculate the total expansion rate (summation of the ibs rate and electron cooling rate) and output to the screen. Must create the ion beam, the ring, the electron beam, and the cooler before calling this command. |
| run_simulation | Simulate the evolution of the ion beam under IBS and/or electron cooling effect(s). |
| srand | Seed the random number. It is used as "srand expression". The expression will be processed by the math parser and the simplest choice is an integer. Seed the random number with the same value will generate the same sequence of random numbers. This may be useful in debugging, testing, or verifying some results. |
| set_n_thread | Set the thread number of OPENMP. It is used as "set_n_thread desired_thread_number". With serial version JSEPC, this command will envoke a warning message, but will not prevent JSPEC from running. With parallel versoin JSPEC, if the thread number is not set using this command, OPENMP will use all the available threads. |

# Example

In the following example, a DC electron cooler and a bunched proton beam is defined. The IBS rate and the electron cooling rate are calculated. Then the evolution of the proton beam under both the IBS effect and the electron cooling effect is simulated for 600 seconds.

```
section_ion             # Define the ion (proton) beam
    charge_number = 1   # Charge number
    mass = 938.272      # Mass of the ion
    kinetic_energy = 8000   # Kinetic energy
    norm_emit_x = 2.2e-6    # Normalized emittance in horizontal direction
    norm_emit_y = 2.2e-6    # Normalized emittance in vertial direction
    momentum_spread = 0.0006    # Momentum spread
    particle_number = 6.58e11   # Total ion number (per bunch)
    rms_bunch_length = 7        # Rms bunch length of the bunched ion beam
section_ring                                # Define the ring
    lattice = MEICColliderRedesign1IP.tfs   # file that saves the lattice of the
ring
section_ibs #define the arguments for IBS calculation
    nu = 100        # Grid number in horizontal direction for IBS integration
    nv = 100        # Grid number in vertial direciton for IBS integration
    nz = 40         # Grid number in longitudinal direction for IBS integration
    log_c = 20.6    # Define Coulomb logrithm. nz is ignored after log_c is
defined.
    coupling = 0    # No coupling
section_cooler              # Define the cooler
    length = 3.4            # Cooler length
    section_number = 1      # Number of coolers
    magnetic_field = 0.039  # Magnetic field
    bet_x = 10              # Twiss parameter at the cooler
    bet_y = 10
    #disp_x = 0             # If the values are zero, the command can be
omitted.
    #disp_y = 0
    #alpha_x = 0
    #alpha_y = 0
    #disp_dx = 0
    #disp_dy = 0
section_scratch             # A scratch section
    m = 938.272             # Define variable m and assign a value.
    ke = 8000               # Define variable ke and assign a value.
    gamma = ke/m + 1        # Define variable gamma and calculate its value.
section_e_beam              # Define the electron beam
    gamma = gamma           # Lorentz factor, the right "gamma" is the variable
define above.
    tmp_tr = 0.1            # Transverse temperature
    tmp_l = 0.01            # Longitudinal temperature
    shape = dc_uniform      # Shape of the electron beam, DC beam with uniform
charge density
    radius = 0.004          # Radius of the DC electron beam
    current = 2             # Current is 2 A
section_ecool                       # Set parameters for electron cooling rate
calculation
    sample_number = 10000           # Number of ion samples
    force_formula = PARKHOMCHUK     # Formula for friction force calculation
```

```
section_run                     # Operation section
    create_ion_beam
    create_ring
    calculate_ibs               # Calculate the IBS rate
    create_e_beam
    create_cooler
    calculate_ecool             # Calculate the electron cooling rate
    total_expansion_rate        # Calculate the total rate = IBS rate + electron
cooling rate
section_simulation                          # Set parameters for simulation
    ibs = on                                # Simulate ISB effect
    e_cool = on                             # Simulate electron cooling effect
    time = 600                              # Time to simulate
    step_number = 600                       # Number of steps
    sample_number = 100000                  # Number of ion samples
    #save_particle_interval = 100           # Save the coordinates of the ions
every 100 steps
    output_file = simulation_test.txt       # File to save the simulation
results
    model = particle                        # Select the model used in the
simulation
section_run                     # Operation section
    run_simulation              # Start simulation
```

# Extended Topics

## Keep a constant bunch length of the ion beam in simulation

The momentum spread of the ion beam changes due to the intrabeam scattering effect and the electron cooling effect during the simulation, hence the bunch length changes if the RF voltage is constant. However, if the RF voltage changes accordingly with the momentum spread, it is possible to maintain a constant bunch length. JSPEC allows the user to choose whether to keep the bunch length constant in the simulation. When the bunch length is maintained constant, the RF voltage is calculated and saved in the output file.

To use this feature, one needs to set the parameter "fixed_bunch_length" in section_simulation to be **true**. One also needs to set the parameters, rf_h (harmonic number), rf_phi (RF phase), and gammar_tr (transition gamma) in section_ring.

```
section_ring #define the ring
    ...
    rf_h = 3584
    rf_phi = 0
    gamma_tr = 12.46

...

section_simulation
    ...
    fixed_bunch_length = true
```

## Use of the scratch section

In the scratch section, one can define variables and perform some simple calculations using the variables. These variables are accessible in the following sections. In the following example, one puts many parameters in the scratch section and use them to define the ion beam, electron beam and the cooler in the following sections. This is convenient to adjust the parameters in simulations since all the parameters are defined on top of the input file.

```
section_scratch
    #Ion beam parameters:
    ex = 0.75e-6    # normalized horizontal emittance
    ey = 0.15e-6    # normalized vertical emittance
    dp = 0.0006 # momentum spread
    np = 0.98e10    # proton number
    ds = 0.02   # proton bunch length
    ke = 100000 # proton kinetic energy
    me = 938.272    # proton mass
    gamma = ke/me+1
    beta = (1-gamma^(-2))^(1/2)
    dx = 0.9    # horizontal dispersion at the cooler
    dy = 0.4    # vertical dispersion at the cooler
    cpl = 0.5   # transverse coupling
    twiss_beta = 100 # beta function at the cooler
    sigma_x = (twiss_beta*ex/beta/gamma)^(1/2) # rms horizontal bunch size
    sigma_y = (twiss_beta*ey/beta/gamma)^(1/2) # rms vertical bunch size

    #Electron beam parameters:
    q_e = 3.2E-9    # electron number
    l_e = 0.03  # electron bunch length
    k_c = 299792458.0   # speed of light
    I_e = q_e*beta*k_c/l_e  # peak current of the electron beam

section_ion #define the ion beam
    charge_number = 1
    mass = 938.272
    kinetic_energy = ke
    norm_emit_x = ex
    norm_emit_y = ey
    momentum_spread = dp
    particle_number = np
    rms_bunch_length = ds

section_ring #define the ring
    lattice = MEICColliderRedesign1IP.tfs

section_ibs #define the arguments for IBS calculation
    model = bm
    log_c = 20
    coupling = cpl

section_cooler
    length = 60
    section_number = 1
    magnetic_field = 1
    bet_x = twiss_beta
    bet_y = twiss_beta
    disp_x = dx
```

```
    disp_y = dy

section_e_beam
    gamma = gamma
    shape = bunched_uniform_elliptic
    rh = sigma_x
    rv = sigma_y
    current = I_e
    length = l_e
    tmp_tr = 0.246
    tmp_l = 0.184
...
```

---

1. If a line ends with "&&&", it will be combined with the following line after two "&"s are trimed. If a line ends with " &&", the "&&" will be trimed, but the space " " will not. Only the white spaces at the left end of the line or after "&&" at the right end of the line will be trimed. ↩

2. The author intended to write this expression in this way in order to emphasize the difference between a scratch variable and a keyword. However, this expression may be confusing. So it is not recommended to use scratch variables with the same name of a keyword. ↩