User guider for CETASim

# Code compiling

CETASIm is an open-source C++ code for beam collective effect simulation in electron rings. Two extra numerical libraries, GSL and FFTW are needed for code compiling. The version of the GSL library has to be larger than 2.7. If the user working with the Linux system, supplying the right path (LIBFLGS) in the Makefile, the source code can be compiled easily by the *make* command

./make

# I/O definition

CETASim takes a file with normal text format as input. All of the key parameters have to be set in the input files. The variables are defined in the format as *“string\_variable =* *values”.* During the operation, CETASim scans the *string\_variale* in the input line-by-line, where the upper and lower cases are not distinguished. Thereafter the *string\_variable* is compared with the *key-word* defined in code to get the values correspondingly for further simulation. The previous definition *“string\_variable =* *values”* will be overwritten by thelatter ones and only the latest ones will be applied in the simulation. The inner structure of the input looks like the *namelist* format in Fortran. It is only for a better parameter classification.

The output of the data given by CETASIm follows the SDDS format. All the data files printed out are with the ASCII format. Users can easily get the meaning of the data by the header of the files. These data can be manipulated by the SDDS tools as well.

# Physical models

There is another document published together with the source code, CETASim\_Reference.pdf [1]. In this document, the user can find out the algorithms and physical models applied in theCETASIm. We highly encourage the users to go through the source code themselves if they are interested in the details.

# Structure of the input

The parameters in the CETASim input file is organized as *namelist* in Fortran. We briefly list the designed “namelist” sections and their purpose in below:

1. &Ring: Parameters to set up the one-turn map in the transverse plane
2. &RFSet: Parameters to set up the RF in the longitudinal plane
3. &Ramping: Parameter to set up raming terms in tracking
4. &Longrangewake: Parameters to set up the long-range wakes in tracking
5. &DriveMode: Parameters to set up the external drivers
6. &ShortRangeWake: Parameters to set up the short-range wakes
7. &BoardBandImpdance: Parameters to set up the board-band impedance
8. &Filling\_pattern: Parameters to set up the beam-filling pattern
9. &Bunch: Parameters to set up the initial beam condition
10. &Ion\_effect: Parameters to set up the ion effect simulation
11. &Bunch-By-Bunch-Feedback: Parameters to set up the bunch-by-bunch feedback simulation
12. &RunSettings: Parameters to specify the run conditions

For the simplest case to run the CETASim, the users have to supply the parameters in &Ring, &RFSet &Filling\_pattern, and &RunSettings at least. The other effects, as impedance, ion effects, et. ac. can be turned on and off by an extra *flag* variable in &RunSettings.

As explained, in each *“nameist”,* the parameters are defined with a format as *“string\_variable =* *values”.* All of the *string\_variables* are designed long so that they are understandable literally. In below we explain the parameters as well.

# &ring

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| ringCircRing | Double | m | circumference of the ring |
| ringWorkQ | double[3] |  | tune values from x,y and z |
| ringSynchRadDampingTime | double[3] |  | radiation damping time for x, y and z planes in units of turns |
| ringPipeAperatureX | Double | m | Pipe radius limit in x-plane |
| ringPipeAperatureY | Double | m | Pipe radius limit in y plane |
| ringPipeAperatureR | Double | m | Not used |
| ringElectronBeamEnergy | Double | eV | Beam energy |
| ringSdelta0 | Double | Rad | Natural energy spread |
| ringNaturaEmit | Double | m | Natural beam emittance |
| ringCoupling | Double |  | Transverse coupling emittance coupling factor |
| ringSkewQuadK | Double | 1/m | Skew quad strength |
| ringU0 | Double | eV | Energy loss per turn |
| ringCurrent | Double | A | Total beam current |
| ringChrom | Double[2] |  | Chromaticity in x and y plane |
| ringAlphac | Double[3] |  | The 1st 2nd and 3rd order of momentum compact factor |
| ringADTX | Double[2] |  | The 1st and the 2nd order of amplitude-depended tune in x plane |
| ringADTY | Double[2] |  | The 1st and the 2nd order of amplitude-depended tune in y plane |
| ringADTXY | Double[2] |  | The 1st and the 2nd order of amplitude-depended tune in xy plane (coupling) |
| ringTwiss | String |  | File to read the one-turn lattice parameter |

Notice:

1. Beam loss criteria in transverse is an elliptical pipe decided by ringPipeAperatureX and ringPipeAperatureY.
2. The longitudinal working tune (ringWorkQ[2]) is not used in the simulation. Supply one number as a placeholder. The longitudinal dynamics is decided by parameters in &RFSet “.
3. ringNaturaEmit is the natural emittance in the horizontal plane.
4. ringCoupling is the emittance coupling factor in the eigenplane. Together with ringNaturaEmit, ringCoupling decides how the natural emittance will be partitionpated in horizontal and vertical planes by the radiation damping and excitation.
5. ringSkewQuadK defines an extra skew quadrupole besides the one-turn matrix. It is used to simulate the linear coupling effect.
6. The design of amplitude-depended tune coefficients refers to the methods used in Elegant code.
7. The twiss data for one turn-matrix construction is imported by *ringTwiss* file. The ion gas pressure and temperature are also defined in this file.

# & RFSet

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| rfRingHarm | Int |  | RF harmonic numer |
| rfResNum | Int |  | Numer of cavities |
| rfResMode | Int[rfResNum] |  | 0: ideal rf, no-zero: generator dynamics and beam induced voltage are considered |
| rfResHarm | Int[rfResNum] |  | Cavity Harmonic number |
| rfResType | Int[rfResNum] |  | Active or passive |
| rfResVol | double[rfResNum] | V | cavity voltage required |
| rfResPhase | double[rfResNum] | rad | cavity phase required |
| rfResShuntImpRs | double[rfResNum] | Ohm | Cavity shunt impedance |
| rfResQuallityQ0 | double[rfResNum] |  | Unloaded quality factor |
| rfResCouplingBeta | double[rfResNum] |  | Cavity coupling factor |
| rfResDetuneFre | double[rfResNum] | Hz | Cavity de-tuning frequency |
| rfResransientCavParWriteTo | String |  | Filename to write cavity data |
| rfResCold | Int[rfResNum] |  | 0: beam induced voltage is build up (0) or not (1) before simulation |
| reResExciteInstability | Int[rfResNum] |  | Coupled bunch instability is excited (1) or not (0). |
| rfResDirFB | Int[rfResNum] |  |  |
| rfResDirFBGain |  |  |  |
| rfResDirFBDelay |  |  |  |
| rfBunchBinNum | Int |  | Number of bins to cut bunch for beam induced voltage simulation |

Notice:

1. rfResNum. It defined how many RF cavities will be applied in the simulation. It also specifies the size of the arrays in the following. It has to be defined firstly in this *namelist*
2. rfResMode. If it is 0, then the beam loading from the fundamental mode is skipped. Only target cavity voltage and phase will drive particle longitudinal dynamics. If it is 1, then the transient beam loading will be simulated. The generator dynamics will be simulated as well. The cavity RF feedback only works when rfResMode is 1 in the active system.
3. rfResType. In the setting to simulate the beam loading, it specifies the cavity is active (1) or passive (0). For the multi-RF system simulation, the main cavity has to be either active type or ideal (rfResMode=0).
4. rfResPhase and rfResVol. The target rf phase and voltage are defined with the Cose convention.
5. rfResShuntImpRs. The cavity shunt impedance is normal definition Rs.
6. rfBunchBinNum. The number of bins to cut the beam into in the transient beam loading simulation. The beam passes through the cavity bunch-by-bunch. Within each bunch, it passes through the cavity bin-by-bin.
7. rfResDirFB. It supplies an ideal cavity feedback algorithm. This module is still underdeveloped and not 100% ready.

# & Ramping

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| rampingNu | Int[2] |  | Ramp horizontal tune or vertical tune or not |
| rampingDTurn | Int |  | Ramping takes place every rampingDTurn turns |
| rampingDNu | Double[2] |  | Change of tune per rampingDTurn turn |
| rampingSKQ | Int |  | Ramp skew quadrupole or not |
| rampingDSKQK | Double | 1/m | Change of the strength of the skew quadrupole per rampingDTurn turn |
| rampingTurns | Int[2] |  | Turns of start and end of ramping simulation |

Notice:

1. The raming element is designed for linear coupling resonance crossing
2. Ramping only works for tune and skew quadrupole strength.
3. Can be extended to study the RF modulation as well.

# & LongRangeWake

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| LWPipeGeoInput | String |  | input file to get the parameters of the resistive wall sectors |
| LRWBBRinput | String |  | Input file to get the resonator parameters |
| LRWTurnsWakeTrunction | Int |  | Turn number of the long-range wake is truncated |

Notice:

1. Long-range wakes are limited to resistive wall and Resonator models.
2. Turns to truncation is comprised of simulation speed and accuracy.

# &BoardBandImpedance

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| BBIimpedInput | String |  | Specifies the file where to get the numerical board band impedance data |
| BBIimpedSimLandTFlag | Int[5] |  | To specify if the tracking includes the longitudinal, horizontal dipole, horizontal quadrupole, vertical dipole and vertical quadrupole or not. |
| BBIimpedSimTimeOrFreFlag | Int |  | Single-bunch beam-impedance interaction is tracking in the frequency domain (0) or time domain (1) |
| BBIimpedSimQuasiGreenBunchLength | Double | M | If tracking is set in the time domain, this parameter specifies the bunch length to get the quasi-green function in time domain. |
| BBIWakeInput | String |  | Specify the filename to print the quasi wake-field for the given bunch length. |

Notice:

1. We recommend the user supply the broadband impedance data launch tracking in the frequency domain.

# & ShortRangeWake

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| SWPipeGeoInput | String |  | Specifies the input file where to get the parameters of the resistive wall parameters. |
| SRWBBRinput | String |  | Specifies the input file where to get the parameters of the broadband resonators. |
| SRWBunchBinNum | Int |  | Number of bins for beam-wake interaction, the bin size is 1 ps |
| SRWWakePotenWriteTo | String |  | File in which the wake potential is written. |

Notice:

1. short wakes are limited to resistive wall and Resonator models.
2. We do not recommend using wakes for single bunch effect, but instead use impedance namelist for single bunch effect.

# & DRIVEMODE

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | Explanation |
| drivemodeON | int |  | The external driver is turned on (1) or off (0) |
| driveStart | Int |  | Turn the number of turns to start the external |
| driveEnd | Int |  | Turn the number to stop the external excitation |
| driveAmp | Double | V/rad | The amplitude of the external driving strength |
| driveFre | Double | Hz | Frequency of the external driver |
| drivePlane | Int |  | The plane that motion is excited. 0: z; 1: x; 2: y |
| driveHW | Int |  | When the coupled bunch motion is simulated by the IQ decomposition method, whether a sine Hanning window is applied to the one-tune bunch-by-bunch data or not |

Notice:

1. The exciter is simply a sinusoid RF field (AC dipole) with the given frequency and amplitude.
2. The exitor only works within turns specified by *driveStart* and *driveEnd.*

# &filling\_pattern

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explanation |
| fpTrainNumber | int |  | Number of bunch trains in the ring |
| fpTotBunchNumer | Int |  | Number of total bunches |
| fpBunchNumberPerTrain | Int[fpTrainNumber] |  | Number of bunches for each bunch train |
| fpTrainGaps | Int[pTrainNumber] |  | Number of empty gaps following each bunch train |
| fpBunchGaps | Int[pTrainNumber] |  | Number of gaps between adjacent bunche in each bunch train |
| fpSetBunchchargeNum | Int |  | Number of bunches have different charges |
| fpSetBunchChargeIndex | Int[fpSetBunchchargeNum] |  | Index for bunches with different charges |
| fpsetBunchCharge | Int[fpSetBunchchargeNum] |  | Relative bunch charge ratio |

Notice:

1. *fpTrainNumber* has to be defined first.
2. There are *rfRingHarm* RF buckets in total
3. Take Petra4 brightness mode for example. “*fpTrainNumber*=80, *fptotBunchNumber*=1600, *fpBunchNumberPerTrain*=20…, *fpTrainGaps*=8,…, *fpBunchGaps*=1,…,”. Then it can be expressed as 3840 = *fpTrainNumber* \* (*fpBunchNumberPerTrain* \*( 1 + *fpBunchGaps*) + *fpTrainGaps* ). 3840 = 80 \* (20\*(1b+1e)+8e)
4. *fpSetBunchchargeNum* specifies how many bunches have different charge states.
5. Example: *fpSetBunchchargeNum*=2; *fpSetBunchChargeIndex* = 0 1; *fpsetBunchCharge=2,2.* It means that the 1st and 2nd bunch have a bunch charge two times larger than the rest of the bunches. The total charge of all bunches ensures the beam current (*ringCurrent*) defined in &ring namelist.

# &bunch

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explaination |
| bunchEmittance | Double | m | Natural emittance of bunch |
| Bunchkappa | Double |  | Initial coupling factor |
| bunchRmsBunchLength | Double | m | Initial rms bunch length |
| bunchRmsEnergySpread | Double | Rad | Initial rms energy spread |
| bunchInitialStaticOffSet | Double[6] | m/rad | Initial stastic offsets of bunch (dx,dy,dz,dpx,dpy,dpz) |
| bunchInitialDynamicOffSet | Double[6] |  | Initial Gaussian error for (dx,dy,dz,dpx,dpy,dpz) |
| bunchDistributionType | Int |  | Initial transverse bunch distrbtion type. 1: KV, 2: WB, 3:GS |
| bunchMacroEleNumPerBunch | Int |  | Macro-electrons per bunch |

Notice:

1. This section is used to generate the initial bunch distribution*.*
2. In multi-bunches simulation, all bunches have the same initial emittance, bunch length, energy spread et.ac.
3. The *bunchEmittance* and *Bunchkappa* only specify the initial emittance.
4. For the longitudinal plane, the initial bunch profile is always 3 sigma Gaussian type.

# &ionEffect

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | explaination |
| ionCalTwissIput | String |  |  |
| ionCalGasSpec | Int |  | Numbe of gas types |
| ionCalIonMassNumber | Int[ionCalGasSpec] |  | Ion mass number |
| ionCalIongasPercent | Doble[ionCalGasSpec] |  | Perstange of ion gas |
| ionCalCorssSectionEI | Doble[ionCalGasSpec] | Mb | Crosssection of each ion specy |
| ionCalIonlossboundary | Double |  | Distance beyond which ions is considered as lost. The distance is in the nit of rms beam size |
| ionCalIonMaxNumber | Double |  | Maximum number of macro-ions allowed in simulation. If macro-ions number is larger than ionCalIonMaxNumber,code stoped. |
| ionCalNumberofIonBeamInterPoint | Int |  | how many interaction point in the ring for ions effect simulation |
| ionCalMacroIonNumberGeneratedPerIP | Int |  | How many maco-ions is gerenated per collision per gas. |
| ionCalIonInfoPrintInterval | Int |  | Turn number to print the ion’s cord out |
| ionCalIonDisWriteTo | String |  | Filename to be printed out |

Notice:

1. According to the lattice data supplied in *ionCalTwissIput*, the ring is cutted into different sections. The transfer matrix of each section is established accordingly. Meanwhile, the *ionCalNumberofIonBeamInterPoint* has to be set consistently.
2. *ionCalGasSpec* has to be set before other parameters
3. *ionCalIonlossboundary* specifies a distance in the unit of transient rms beam size. Ions are cleaned if their distances to the bunch center are beyond the value.

# &bunch-by-bunch-feedback

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | Explanation |
| fbnumberofSections | Int |  |  |
| fbStart | Int[fbnumberofSections] |  | Turn number to turn on the bunvh-by-bunch feedback |
| fbEnd | Int[fbnumberofSections] |  | Turn number to turn off the bunch-by-bunch feedback |
| fbKickStrengthKx | Double |  | Factor to be multiplied on kick angle in x plane |
| fbKickStrengthKy | Double |  | Factor to be multiplied on kick angle in x plane |
| fbKickStrengthF | Double |  | Factor to be multiplied on kick angle in z plane |
| fbGain | Double |  | Fbgain factor |
| fbdelay | Int |  | FB Delay in unit turns |
| fbTaps | Int |  | Taps of FIR filter |
| fbKickerDispP | Double | M | Dispersion’ at the FBkicker |
| fbKickerDispP | Double |  | Dispersion at the FBkicker |
| fbFIRBunchByBunchFeedbackPowerLimit | Double | watt | Kicker power limit |
| fbFIRBunchByBunchFeedbackKickerImped | Double | ohm | Kicker impedance |
| fbFircoeffx | Double[fbTaps] |  | FIR filter coefficients in x plane |
| fbFircoeffy | Double[fbTaps] |  | FIR filter coefficients in y plane |
| fbFircoeffz | Double[fbTaps] |  | FIR filter coefficients in z plane (not ready) |
| fbFircoeffxy | Double[fbTaps] |  | FIR filter coefficients in xy plane (not ready yet) |

Notice:

1. *fbFircoeffz* is not benchmarked. *fbFircoeffxy* is not ready yet.

# &runsettings

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter name | data type | units | Explanation |
| runcalSetting | Int |  | Each bunch is composed by one macro-particle or multi macro-partilce |
| runNTurns | Int |  | Number of tracking tunrs |
| runBunchInfoPrintInterval | Int |  | Time distance to printout the Turn-by-turn data in unit of turns |
| runGrowthRateFittingStart |  |  | Data to be taken from this turn to get the coupled bunch growth rate |
| runGrowthRateFittingEnd |  |  | Data to be taken to this turn to get the coupled bunch growth rate |
| runTBTBunchPrintNum | Int |  | The number of bunches to be printed out |
| runTBTBunchDisDataBunchIndex | Int[runTBTBunchPrintNum] |  | Specify the bunch indexes to be printed out |
| runTBTBunchAverData | String |  | Filename of the turn-by-turn, bunch-by-bunch parameters |
| runTBTBunchHaissinski | String |  | Filename where the BunchProiles to be printed out |
| runTBTBunchDisData | String |  | Filename where the turn-by-turn bunch-by-bunch distribution is printed out |
| runTBTBunchProfile | String |  | Filename where the Turn-by-turn bunch-by-bunch profiles are printed out |
| runSynRadDampingFlag | Int |  | Tracking includes the radiation and quantum excitation or not |
| runBeamIonFlag | Int |  | Tranking includes the ion effect or not |
| runFIRBunchByBunchFeedbackFlag | Int |  | Tracking includes bunch-by-bunch feedback or not |
| runLongRangeWake | Int |  | Tracking includes the long-range wakes or not |
| runShortRangeWake | Int |  | Tracking includes the short-range wakes or not |
| runBBIFlag | Int |  | Tracking includes the board-band impedance effect or not |
| runRamping | Int |  | Trcking including ramping parameter or not |

Notice:

1. *runcalSetting*, if it is 1, it means that each bunch only contains one macro-particle. The *bunchMacroEleNumPerBunch* in &Bunch namelist has to be set as one as well.
2. *runTBTBunchDisData,* this parameter specifies if the bunch distribution is printed out or not. It only works when the macro-particle number per bunch is larger than 1.
3. *runBunchInfoPrintInterval* refers the tracking turn interval to print out the bunch info.
4. *runTBTBunchDisDataBunchIndex* specifies the bunch index to be printed out. It refers to the bunch index but not the rf bucket index.
5. *runTBTBunchProfile,* to print out the beam profile or not. It only works only when the particle number per bunch is larger than 1.
6. *runCBMGR* specifies whether the coupled bunch growth rate is calculated or not. If it is set as 1, the coupled bunch growth rate is obtained by using the bunch-by-bunch data from *runGrowthRateFittingStart* to *runGrowthRateFittingEnd* turns.The algrithem applied from coupled bunch growth rate calculation in CETASim is only available for a uniform filling pattern scheme.

# Bibliography

|  |  |
| --- | --- |
| [1] | "https://github.com/ESR-BeamSimulation/CETASIM," [Online]. |