

IMPACT T

Generated by Doxygen 1.8.15

Chapter 1

IMPACT-T Documentation

IMPACT-T: A 3D Parallel Particle Tracking Code in Time Domain

IMPACT-T is a fully three-dimensional program to track relativistic charged particles taking into account space charge forces, short-range longitudinal and transverse wakefields, coherent synchrotron radiation (CSR) wakefield in accelerators. IMPACT-T code can run on both massive parallel supercomputers and single processor computers such as Windows PC, Mac, and Linux system. It is one of the few codes used in the photoinjector community that has a parallel implementation, making it very useful for high statistics simulations of beam halos and beam diagnostics. It has a comprehensive set of beamline elements, and furthermore allows arbitrary overlap of their fields, which gives the IMPACT-T a capability to model both the standing wave structure and traveling wave structure. It includes mean-field space-charge solvers based on an integrated Green function to efficiently and accurately treat beams with large aspect ratio, and a shifted Green function to efficiently treat image charge effects of a cathode. It is also unique in its inclusion of energy binning in the space-charge calculation to model beams with large energy spread. It also has a direct N-body solver to calculate stochastic space-charge forces. IMPACT-T has a flexible data structure that allows particles to be stored in containers with common characteristics; for photoinjector simulations the containers represent multiple slices, but in other applications they could correspond, e.g., to particles of different species. Together, all these features make IMPACT-T a powerful and versatile tool for modeling beams in photoinjectors and other systems.

Here is the link to the home page of IMPACT-T: <https://amac.lbl.gov/~jiquiang/IMPACT-T/index.html>

Here is the link to the GitHub of IMPACT-T: <https://github.com/impact-lbl/IMPACT-T>

This is the license statement:

*** License Agreement ***

IMPACT-T, Copyright (c) 2016, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy). All rights reserved." >Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the University of California, Lawrence Berkeley National Laboratory, U.S. Dept. of Energy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

You are under no obligation whatsoever to provide any bug fixes, patches, or upgrades to the features, functionality or performance of the source code ("Enhancements") to anyone; however, if you choose to make your Enhancements available either publicly, or directly to Lawrence Berkeley National Laboratory, without imposing a separate written license agreement for such Enhancements, then you hereby grant the following license: a non-exclusive, royalty-free perpetual license to install, use, modify, prepare derivative works, incorporate into other computer software, distribute, and sublicense such enhancements or derivative works thereof, in binary and source code form.

This is the README file:

*** Copyright Notice ***

"IMPACT-T" Copyright (c) 2016, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from the U.S. Dept. of Energy). All rights reserved.

If you have questions about your rights to use or distribute this software, please contact Berkeley Lab's Innovation & Partnerships Office at IPO@lbl.gov.

NOTICE. This Software was developed under funding from the U.S. Department of Energy and the U.S. Government consequently retains certain rights. As such, the U.S. Government has been granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable, worldwide license in the Software to reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit other to do so.

V2.0

Note:

1. The current version of the code is for serial single processor computer with Fortran90 compiler. To run the code on a parallel computer with MPI, the user has to comment out the line "use mpistub" in [Contrl/Input.f90](#), [DataStruct/Data.f90](#), [DataStruct/Pgrid.f90](#), [DataStruct/PhysConst.f90](#), and [Func/Timer.f90](#). The user also has to remove the mpif.h file under the Appl, Control, DataStruct, and Func directories. The user also has to modify the Makefile to remove the mpistub.o inside the file and to use the appropriate parallel F90 compiler such as mpif90.
2. The phaseOpt.py is used to find the driven phase of a RF cavity with initial design phase. This code needs to be modified for each input ImpactT.in file in order to use it correctly.
3. The subroutines in [FFT.f90](#): realft, four1, and sinft, can be replaced with functions from the Numerical Recipe or some equivalent 1D FFT functions.

Chapter 2

Modules Index

2.1 Packages

Here are the packages with brief descriptions (if available):

accsimulatorclass	This class defines functions to set up the initial beam particle distribution, field information, computational domain, beam line element lattice and run the dynamics simulation through the system	??
beambunchclass	This class defines the charged particle beam bunch information in the accelerator	??
beamlineelemclass	This class defines the base beam line element class for different lattice element class	??
bpmclass	This class defines the different beam diagnostics at given beam position	??
ccdtlclass	This class defines the linear transfer map and RF field for the CCDTL beam line element	??
cclclass	This class defines the linear transfer map and RF field for the CCL beam line element	??
compdomclass	This class defines 3-D global and local computational domain in the parallel simulation	??
constfocclass	This class defines the linear transfer map and field for the 3d constant focusing beam line element	??
dataclass	This class stores the rf cavity data E_z , E_z' , E_z'' on the axis; Fourier coefficients of E_z on the axis; $E_z(r,z)$, $E_r(r,z)$, $H_{\theta}(r,z)$ on the r - z grid plane; and $E_x(x,y,z)$, $E_y(x,y,z)$, $E_z(x,y,z)$, $B_x(x,y,z)$, $B_y(x,y,z)$, $B_z(x,y,z)$ on uniform x , y , z grid, and $B_r(r,z)$ and $B_z(r,z)$ on the r - z grid	??
depositorclass	This class deposit the particles onto computational mesh implementation	??
dipoleclass	This class defines the linear transfer map and field for the Dipole beam line element	??
distributionclass	This class defines initial distributions for the charged particle beam bunch information in the accelerator	??
drifttubeclass	This class defines the linear transfer map and field for the drift space beam line element	??
dtlclass	This class defines the linear transfer map and RF field for the DTL beam line element	??
emfldanaclass	This class contains discrete EM field data (as a function of x,y,z) or (r,z) and analytical representation of EM field data (user can supply the function form). The linear transfer map is also computed base on the field on the axis	??

emfldcartclass	This class contains discrete EM field data (as a function of x,y,z) or (r,z) and analytical representation of EM field data (user can supply the function form). The linear transfer map is also computed base on the field on the axis	??
emfldclass	This class contains discrete EM field data (as a function of x,y,z) or (r,z) and analytical representation of EM field data (user can supply the function form). The linear transfer map is also computed base on the field on the axis	??
emfldcylclass	This class contains discrete EM field data (as a function of x,y,z) or (r,z) and analytical representation of EM field data (user can supply the function form). The linear transfer map is also computed base on the field on the axis	??
fftclass	This class defines the 3d FFT transformation subject to open or periodic conditions, Fourier Sine transformation, Complex-Complex, Complex-Real, and Real-Complex FFT	??
fieldquantclass	This class defines a 3-D field quantity in the accelerator. The field quantity can be updated at each step	??
fldmgerclass	This class defines the functions to sum up the particle contribution from neighboring processor domain, exchange the potential, exchange the field for interpolation between neighboring processors	??
inputclass	This class defines functions to input the global beam and computational parameters and the lattice input parameters in the accelerator	??
mpistub	??
multipoleclass	This class defines the linear transfer map and field for the multipole (sextupole, octupole, decapole) beam line element	??
numconstclass	This class defines the maximum size for numerical parameters in the simulation	??
outputclass	This class defines functions to print out the charged particle beam information in the accelerator	??
pgrid2dclass	This class construct a logical 2-D Cartesian processor grid	??
physconstclass	This class defines the physical constant parameters used in the simulation	??
ptclmgerclass	This class defines functions to transport particles to their local computation processor domain through an iterative neighboring processor communication process	??
quadrupoleclass	This class defines the linear transfer map and field for the quadrupole beam line element	??
rangerclass	Find the global range of computation domain class implementation	??
scclass	This class defines the linear transfer map and RF field for the SC beam line element	??
solclass	This class defines the linear transfer map and field for the Solenoid beam line element	??
solrfclass	This class defines the linear transfer map and RF field for the Sol-RF beam line element	??
timerclass	This module is to record time spent in different subroutines	??
transposeclass	This class defines a tranpose class which contains 2D and 3D tranpose functions. (Both arrays are distributed on 2D Cartesian processor array)	??

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

beamlineelemclass::assign_beamlineelem	??
compdomclass::balance_compdom	??
accsimulatorclass::construct_accsimulator	
Beam line element period	??
compdomclass::construct_compdom	??
fftclass::fftcrlcal_fft	??
fftclass::fftrclcal_fft	??
beamlineelemclass::getparam_beamlineelem	??
bpmclass::getparam_bpm	??
ccdtlclass::getparam_ccdtl	??
cclclass::getparam_ccl	??
constfocclass::getparam_constfoc	??
dipoleclass::getparam_dipole	??
drifftubeclclass::getparam_drifftube	??
dtlclass::getparam_dtl	??
emfldclass::getparam_emfld	??
emfldanaclass::getparam_emfldana	??
emfldcartclass::getparam_emfldcart	??
emfldcylclass::getparam_emfldcyl	??
multipoleclass::getparam_multipole	??
quadrupoleclass::getparam_quadrupole	??
scclass::getparam_sc	??
solclass::getparam_sol	??
solrfclass::getparam_solrf	??
rangerclass::globalrange	??
inputclass::in_input	??
mpistub::mpi_allgather	??
mpistub::mpi_allgatherv	??
mpistub::mpi_allreduce	??
mpistub::mpi_alltoallv	??
mpistub::mpi_bcast	??
mpistub::mpi_gather	??
mpistub::mpi_irecv	??
mpistub::mpi_isend	??
mpistub::mpi_recv	??

mpistub::mpi_reduce	??
mpistub::mpi_send	??
compdomclass::setlctab_compdom	??
beamlineelemclass::setparam_beamlineelem	??
bpmclass::setparam_bpm	??
ccdtlclass::setparam_ccdtl	??
cclclass::setparam_ccl	??
constfocclass::setparam_constfoc	??
dipoleclass::setparam_dipole	??
drifftube::setparam_drifftube	??
dtlclass::setparam_dtl	??
emfldclass::setparam_emfld	??
emfldana::setparam_emfldana	??
emfldcart::setparam_emfldcart	??
emfldcyl::setparam_emfldcyl	??
multipoleclass::setparam_multipole	??
quadrupoleclass::setparam_quadrupole	??
scclass::setparam_sc	??
solclass::setparam_sol	??
solrfclass::setparam_solrf	??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/ mpif.h	??
src/ mpistub.f90	??
src/ Appl/ BeamBunch.f90	??
src/ Appl/ BeamLineElem.f90	??
src/ Appl/ BPM.f90	??
src/ Appl/ CCDTL.f90	??
src/ Appl/ CCL.f90	??
src/ Appl/ CompDom.f90	??
src/ Appl/ ConstFoc.f90	??
src/ Appl/ Depositor.f90	??
src/ Appl/ Dipole.f90	??
src/ Appl/ Distribution.f90	??
src/ Appl/ DriftTube.f90	??
src/ Appl/ DTL.f90	??
src/ Appl/ EMfld.f90	??
src/ Appl/ EMfldAna.f90	??
src/ Appl/ EMfldCart.f90	??
src/ Appl/ EMfldCyl.f90	??
src/ Appl/ Field.f90	??
src/ Appl/ mpif.h	??
src/ Appl/ Multipole.f90	??
src/ Appl/ Quadrupole.f90	??
src/ Appl/ Ranger.f90	??
src/ Appl/ SC.f90	??
src/ Appl/ Sol.f90	??
src/ Appl/ SolRF.f90	??
src/ Contrl/ AccSimulator.f90	??
src/ Contrl/ Input.f90	??
src/ Contrl/ main.f90	??
src/ Contrl/ mpif.h	??
src/ Contrl/ Output.f90	??
src/ DataStruct/ Data.f90	??
src/ DataStruct/ mpif.h	??
src/ DataStruct/ NumConst.f90	??
src/ DataStruct/ Pgrid.f90	??

src/DataStruct/ PhysConst.f90	??
src/Func/ FFT.f90	??
src/Func/ Fldmger.f90	??
src/Func/ mpif.h	??
src/Func/ Ptclmger.f90	??
src/Func/ Timer.f90	??
src/Func/ Transpose.f90	??

Chapter 5

Module Documentation

5.1 accsimulatorclass Module Reference

This class defines functions to set up the initial beam particle distribution, field information, computational domain, beam line element lattice and run the dynamics simulation through the system.

Data Types

- interface [construct_accsimulator](#)
beam line element period.

Functions/Subroutines

- subroutine [init_accsimulator](#) (time)
set up objects and parameters.
- subroutine [run_accsimulator](#) ()
Run beam dynamics simulation through accelerator.
- subroutine [rebin_utility](#) (this, Nbunch, ibunch, dGspread)
- subroutine [destruct_accsimulator](#) (time)

Variables

- integer [nbunch](#)
initial # of bunches/bins
- type(pgrid2d) [grid2d](#)
2d logical processor array
- type(beam bunch), dimension(nbunchmax) [ebunch](#)
beam particle object and array.
- type(fieldquant) [potential](#)
beam charge density and field potential arrays.
- type(compdom) [ageom](#)
geometry object.
- type(fielddata), dimension(maxoverlap) [fldmp](#)
overlaped external field data array

- double precision `temission`
maximum e- emission time
- integer `nemission`
number of steps for emission
- double precision `zimage`
distance after that to turn off image space-charge
- double precision, dimension(2, nblemtmax) `zblnelem`
longitudinal position of each element (min and max).

of phase dim., num. total and local particles, int. dist. and restart switch, error study switch, substep for space-charge switch, # of time step

- integer `dim`
- integer `flagdist`
- integer `rstartflg`
- integer `flagerr`
- integer `flagsubstep`
- integer `ntstep`
- integer, dimension(nbunchmax) `np`
- integer, dimension(nbunchmax) `nplocal`

of num. total x, total and local y mesh pts., type of BC, # of beam elems, type of integrator. FlagImage: switch flag for image space-charge force calculation: "1" for yes, otherwise for no.

- integer `nx`
- integer `ny`
- integer `nz`
- integer `nxlocal`
- integer `nylocal`
- integer `nzlocal`
- integer `flagbc`
- integer `nblem`
- integer `flagmap`
- integer `flagdiag`
- integer `flagimage`

of processors in column and row direction.

- integer `npcol`
- integer `nprow`

beam current, kin. energy, part. mass, charge, ref. freq., period length, time step size

- double precision [bcurr](#)
- double precision [bkenergy](#)
- double precision [bmass](#)
- double precision [bcharge](#)
- double precision [bfreq](#)
- double precision [perdlen](#)
- double precision [dt](#)
- double precision [xrad](#)
- double precision [yrad](#)

conts. in init. dist.

- integer, parameter [ndistparam](#) = 21
- double precision, dimension([ndistparam](#)) [distparam](#)

restart time and step

- double precision [tend](#)
- double precision [dtlessend](#)
- integer [iend](#)
- integer [nfileout](#)
- integer [ioutend](#)
- integer [itszend](#)
- integer [isteerend](#)
- integer [isloutend](#)

beam line element array.

- type(bpm), dimension([nbpmmax](#)), target [beamln0](#)
- type(drifttube), dimension([ndriftmax](#)), target [beamln1](#)
- type(quadropole), dimension([nquadmax](#)), target [beamln2](#)
- type(dtl), dimension([ndtlmax](#)), target [beamln3](#)
- type(ccdtl), dimension([nccdtlmax](#)), target [beamln4](#)
- type(ccl), dimension([ncclmax](#)), target [beamln5](#)
- type(sc), dimension([nscmax](#)), target [beamln6](#)
- type(constfoc), dimension([ncfmax](#)), target [beamln7](#)
- type(solrf), dimension([nslrfmax](#)), target [beamln8](#)
- type(sol), dimension([nslmax](#)), target [beamln9](#)
- type(dipole), dimension([ndipolemax](#)), target [beamln10](#)
- type(emfld), dimension([ncclmax](#)), target [beamln11](#)
- type(emfldcart), dimension([ncclmax](#)), target [beamln12](#)
- type(emfldcyl), dimension([ncclmax](#)), target [beamln13](#)
- type(emfldana), dimension([ncclmax](#)), target [beamln14](#)
- type(multipole), dimension([nquadmax](#)), target [beamln15](#)
- type(beamlineelem), dimension([nblemtmax](#)) [blelem](#)

5.1.1 Detailed Description

This class defines functions to set up the initial beam particle distribution, field information, computational domain, beam line element lattice and run the dynamics simulation through the system.

Author

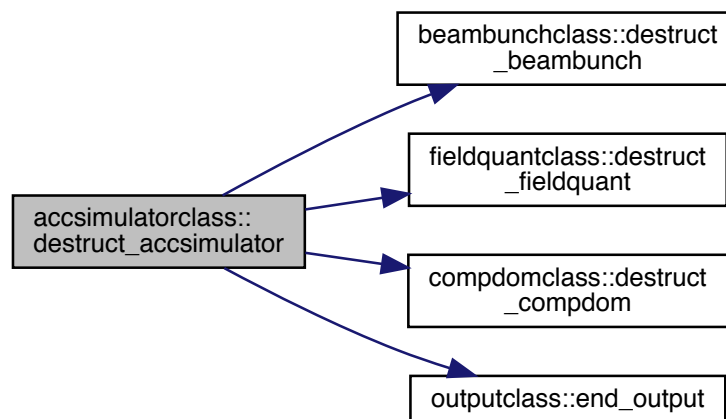
Ji Qiang

5.1.2 Function/Subroutine Documentation

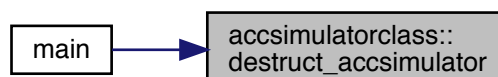
5.1.2.1 destruct_accsimulator()

```
subroutine accsimulatorclass::destruct_accsimulator (
    double precision time )
```

Here is the call graph for this function:



Here is the caller graph for this function:

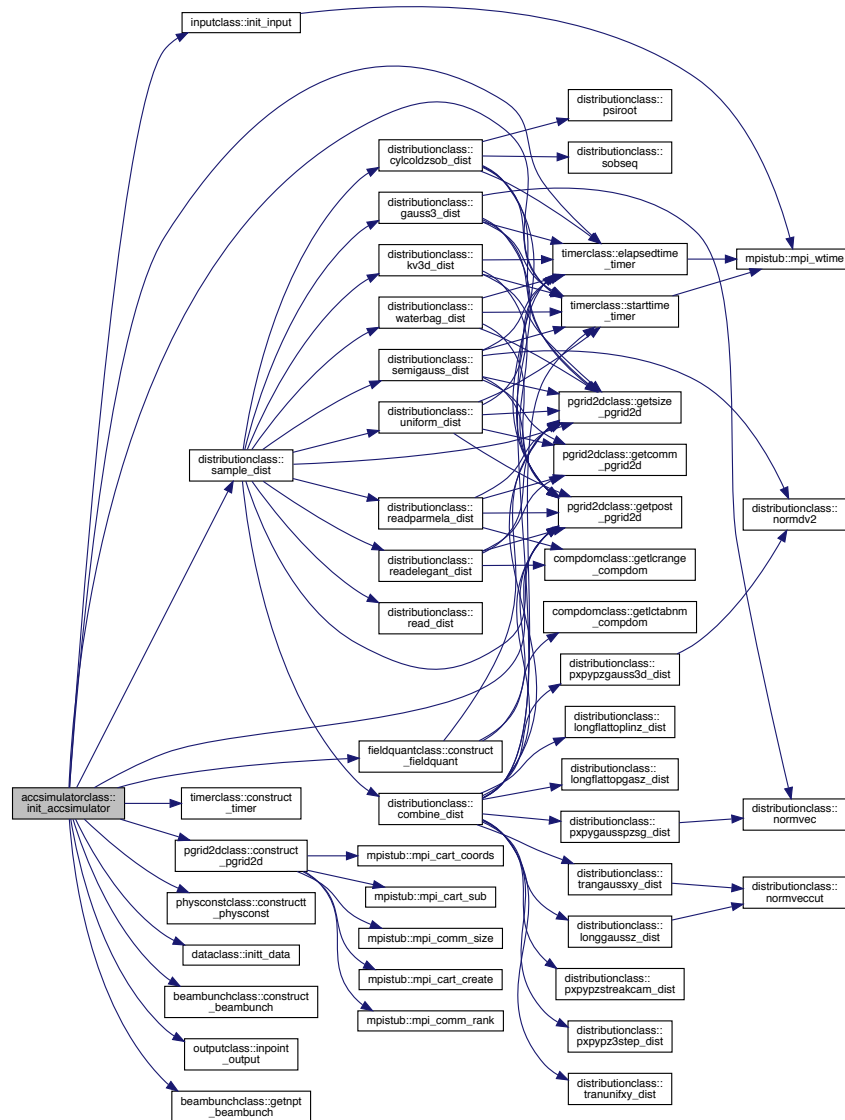


5.1.2.2 init_accsimulator()

```
subroutine accsimulatorclass::init_accsimulator (
    double precision time )
```

set up objects and parameters.

Here is the call graph for this function:



5.1.2.3 rebin_utility()

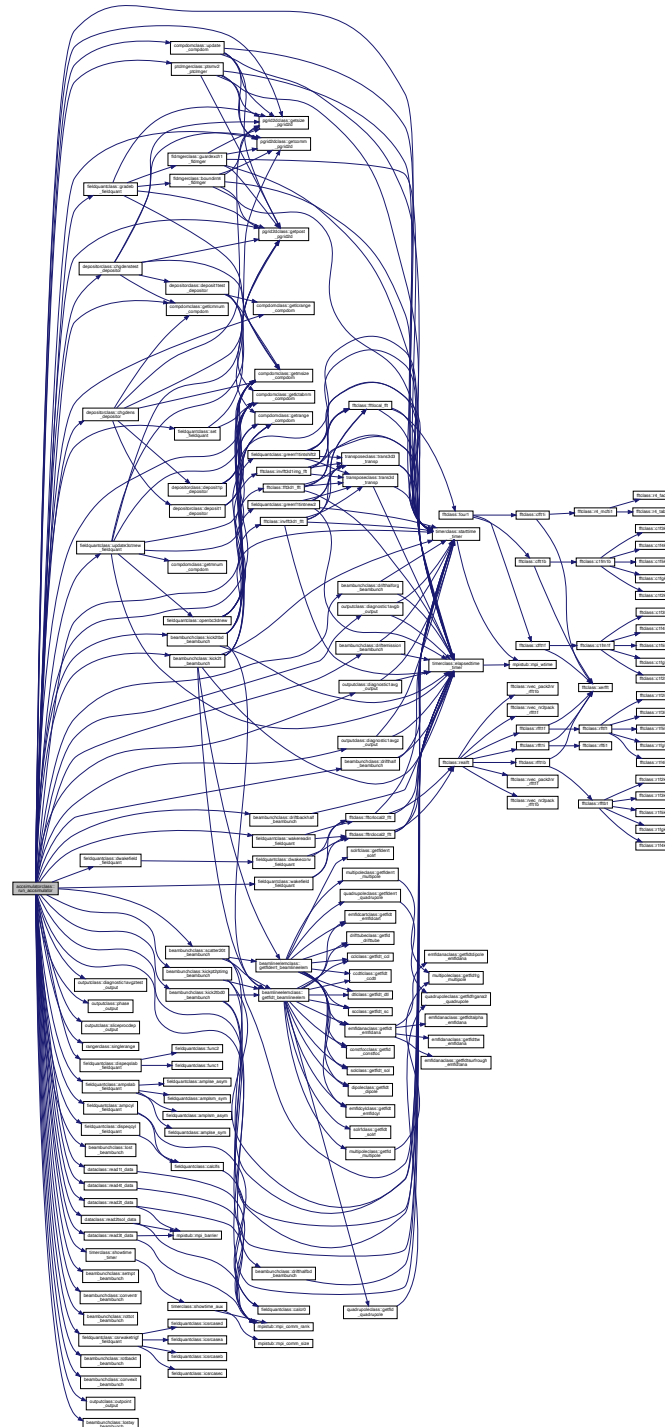
```
subroutine accsimulatorclass::rebin_utility (
    type (beambunch), dimension(:), intent(inout) this,
    integer, intent(in) Nbunch,
    integer, intent(out) ibunch,
    double precision, intent(in) dGspread )
```

5.1.2.4 run_accsimulator()

```
subroutine accsimulatorclass::run_accsimulator ( )
```

Run beam dynamics simulation through accelerator.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3 Variable Documentation

5.1.3.1 ageom

`type (compdom) accsimulatorclass::ageom`

geometry object.

5.1.3.2 bcharge

`double precision accsimulatorclass::bcharge`

5.1.3.3 bcurr

`double precision accsimulatorclass::bcurr`

5.1.3.4 beamln0

`type (bpm), dimension(nbpmmx), target accsimulatorclass::beamln0`

5.1.3.5 beamln1

`type (drifftube), dimension(ndriftmax), target accsimulatorclass::beamln1`

5.1.3.6 beamln10

```
type (dipole), dimension(ndipolemax), target accsimulatorclass::beamln10
```

5.1.3.7 beamln11

```
type (emfld), dimension(ncclmax), target accsimulatorclass::beamln11
```

5.1.3.8 beamln12

```
type (emfldcart), dimension(ncclmax), target accsimulatorclass::beamln12
```

5.1.3.9 beamln13

```
type (emfldcyl), dimension(ncclmax), target accsimulatorclass::beamln13
```

5.1.3.10 beamln14

```
type (emfldana), dimension(ncclmax), target accsimulatorclass::beamln14
```

5.1.3.11 beamln15

```
type (multipole), dimension(nquadmax), target accsimulatorclass::beamln15
```

5.1.3.12 beamln2

```
type (quadrupole), dimension(nquadmax), target accsimulatorclass::beamln2
```

5.1.3.13 beamln3

```
type (dtl), dimension(ndtlmax), target accsimulatorclass::beamln3
```

5.1.3.14 beamln4

```
type (ccdtl), dimension(nccdtlmax), target accsimulatorclass::beamln4
```

5.1.3.15 beamln5

```
type (ccl), dimension(ncclmax), target accsimulatorclass::beamln5
```

5.1.3.16 beamln6

```
type (sc), dimension(nscmax), target accsimulatorclass::beamln6
```

5.1.3.17 beamln7

```
type (constfoc), dimension(ncfmax), target accsimulatorclass::beamln7
```

5.1.3.18 beamln8

```
type (solrf), dimension(nslrfmax), target accsimulatorclass::beamln8
```

5.1.3.19 beamln9

```
type (sol), dimension(nslmax), target accsimulatorclass::beamln9
```

5.1.3.20 bfreq

```
double precision accsimulatorclass::bfreq
```

5.1.3.21 bkenergy

```
double precision accsimulatorclass::bkenergy
```

5.1.3.22 bnelem

```
type (beamlineelem), dimension(nblemtmax) accsimulatorclass::bnelem
```

5.1.3.23 bmass

```
double precision accsimulatorclass::bmass
```

5.1.3.24 dim

```
integer accsimulatorclass::dim
```

5.1.3.25 distparam

```
double precision, dimension(ndistparam) accsimulatorclass::distparam
```

5.1.3.26 dt

```
double precision accsimulatorclass::dt
```

5.1.3.27 dtlessend

```
double precision accsimulatorclass::dtlessend
```

5.1.3.28 ebunch

```
type (beambunch), dimension(nbunchmax) accsimulatorclass::ebunch
```

beam particle object and array.

5.1.3.29 flagbc

```
integer accsimulatorclass::flagbc
```

5.1.3.30 flagdiag

```
integer accsimulatorclass::flagdiag
```

5.1.3.31 flagdist

```
integer accsimulatorclass::flagdist
```

5.1.3.32 flagerr

```
integer accsimulatorclass::flagerr
```

5.1.3.33 flagimage

```
integer accsimulatorclass::flagimage
```

5.1.3.34 flagmap

```
integer accsimulatorclass::flagmap
```

5.1.3.35 flagsubstep

```
integer accsimulatorclass::flagsubstep
```

5.1.3.36 fldmp

```
type (felddata), dimension(maxoverlap) accsimulatorclass::fldmp
```

overlaped external field data array

5.1.3.37 grid2d

```
type (pgrid2d) accsimulatorclass::grid2d
```

2d logical processor array

5.1.3.38 iend

```
integer accsimulatorclass::iend
```

5.1.3.39 ioutend

```
integer accsimulatorclass::ioutend
```

5.1.3.40 isloutend

```
integer accsimulatorclass::isloutend
```

5.1.3.41 istearend

```
integer accsimulatorclass::istearend
```

5.1.3.42 itszend

```
integer accsimulatorclass::itszend
```

5.1.3.43 nblem

```
integer accsimulatorclass::nblem
```

5.1.3.44 nbunch

```
integer accsimulatorclass::nbunch
```

initial # of bunches/bins

5.1.3.45 ndistparam

```
integer, parameter accsimulatorclass::ndistparam = 21
```

5.1.3.46 nemission

```
integer accsimulatorclass::nemission
```

number of steps for emission

5.1.3.47 nfileout

```
integer accsimulatorclass::nfileout
```

5.1.3.48 np

```
integer, dimension(nbunchmax) accsimulatorclass::np
```

5.1.3.49 npcol

```
integer accsimulatorclass::npcol
```

5.1.3.50 nplocal

```
integer, dimension(nbunchmax) accsimulatorclass::nplocal
```

5.1.3.51 nprow

```
integer accsimulatorclass::npro
```

5.1.3.52 ntstep

```
integer accsimulatorclass::ntstep
```

5.1.3.53 nx

```
integer accsimulatorclass::nx
```

5.1.3.54 nxlocal

```
integer accsimulatorclass::nxlocal
```

5.1.3.55 ny

```
integer accsimulatorclass::ny
```

5.1.3.56 nylocal

```
integer accsimulatorclass::nylocal
```

5.1.3.57 nz

```
integer accsimulatorclass::nz
```


5.1.3.58 nzlocal

```
integer accsimulatorclass::nzlocal
```

5.1.3.59 perdlen

```
double precision accsimulatorclass::perdlen
```

5.1.3.60 potential

```
type (fieldquant) accsimulatorclass::potential
```

beam charge density and field potential arrays.

5.1.3.61 rstartflg

```
integer accsimulatorclass::rstartflg
```

5.1.3.62 temission

```
double precision accsimulatorclass::temission
```

maximum e- emission time

5.1.3.63 tend

```
double precision accsimulatorclass::tend
```

5.1.3.64 xrad

```
double precision accsimulatorclass::xrad
```

5.1.3.65 yrad

double precision accsimulatorclass::yrad

5.1.3.66 zblnelem

double precision, dimension(2,nblemtmax) accsimulatorclass::zblnelem

longitudinal position of each element (min and max).

5.1.3.67 zimage

double precision accsimulatorclass::zimage

distance after that to turn off image space-charge

5.2 beambunchclass Module Reference

This class defines the charged particle beam bunch information in the accelerator.

Data Types

- type [beambunch](#)

Functions/Subroutines

- subroutine [construct_beambunch](#) (this, incurr, inkin, inmass, incharge)
Initialize Beambunch class.
- subroutine [setnpt_beambunch](#) (this, innpt)
Set local # of particles.
- subroutine [getnpt_beambunch](#) (this, outnpt)
Get local # of particles.
- subroutine [drifthalf_beambunch](#) (this, t, tau, betazini)
*Drift half step in positions. Here, x, y, z are normalized by $C * Dt$ tau - normalized step size (by Dt). Only particle with $z > 0$ is drifted.*
- subroutine [driftemission_beambunch](#) (this, t, tau, betazini)
Particle emission For particle with $z < 0$, they are just shifted long z This is used to simulate the process of emission from photocathod.
- subroutine [drifthalforg_beambunch](#) (this, t, tau)
*Drift half step in positions. Here, x, y, z are normalized by $C * Dt$ tau - normalized step size (by Dt).*
- subroutine [driftz_beambunch](#) (this, dz)
- subroutine [kick1t_beambunch](#) (this, beamelem, zbeamelem, idrfile, nbea)