# 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/~jiqiang/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:

## This is the README file:

*** Copyright Notice ***



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 parall 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 equavilent 1D FFT functions.

# Chapter 2

# Modules Index

## 2.1 Packages

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

Chapter 3

# Data Type Index

## 3.1 Data Types List

Here are the data types with brief descriptions:

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:

# 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(beambunch), 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(quadrupole), 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) blnelem

### 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**

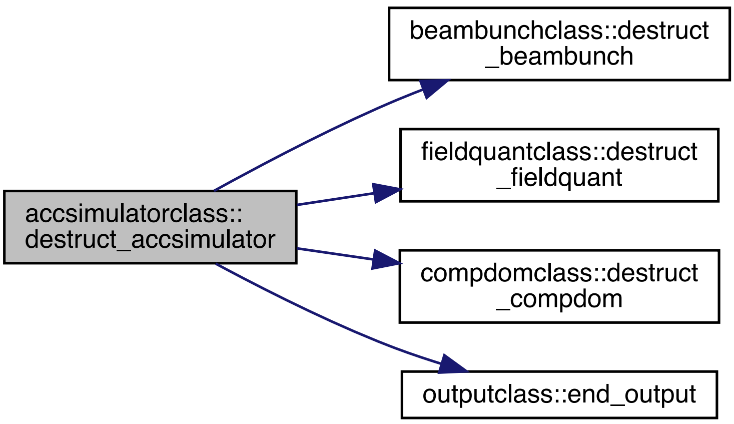
Ji Qiang

### 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:



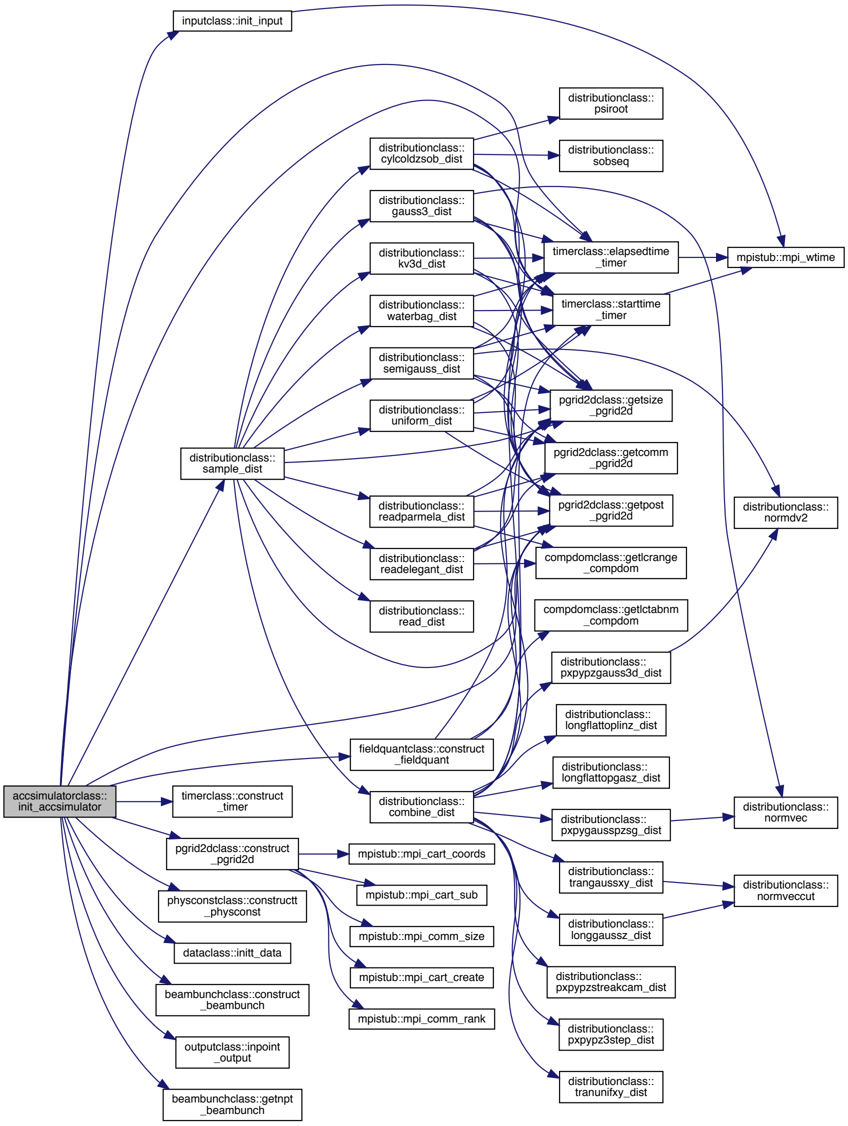

Here is the caller graph for this function:

#### 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()

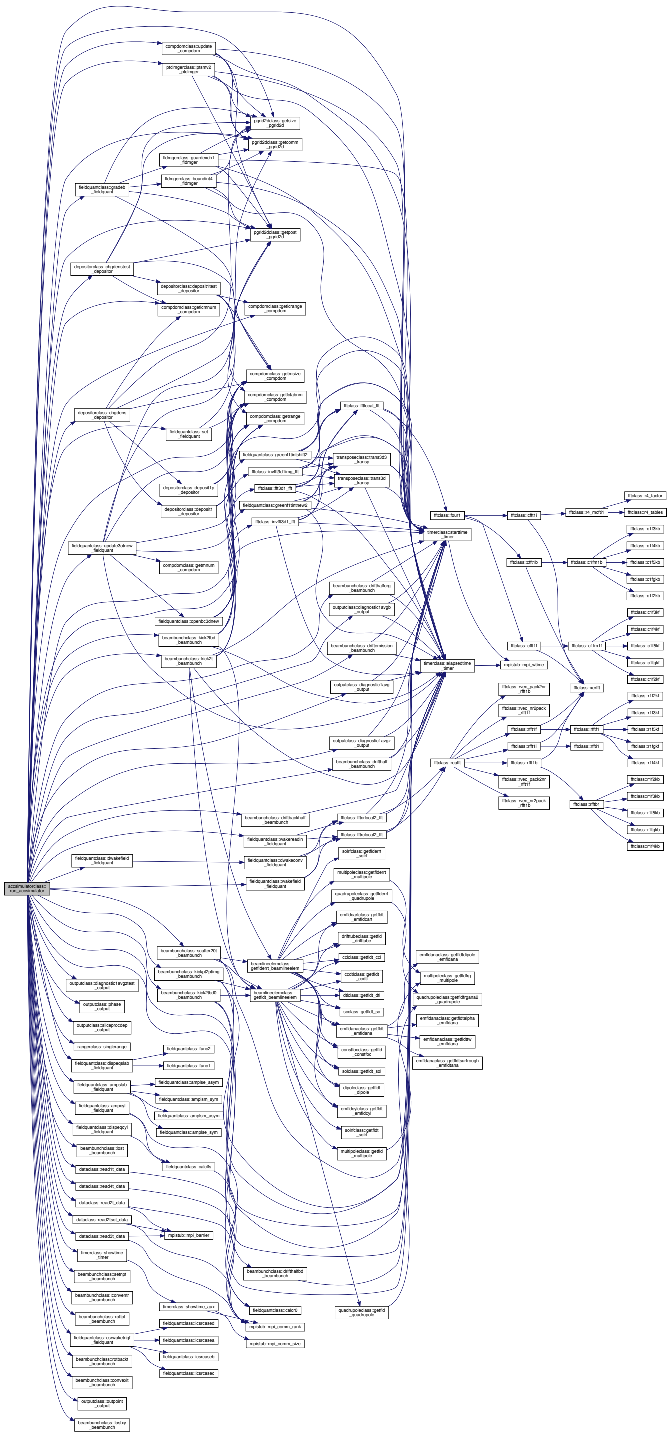
```
subroutine accsimulatorclass::rebin_utility (
            type (beambunch), dimension(:), intent(inout) this,
            integer, intent(in) Nbunch,
            integer, intent(out) ibunch,
            double precision, intent(in) dGspread )
```

#### 5.1.2.4 run_accsimulator()

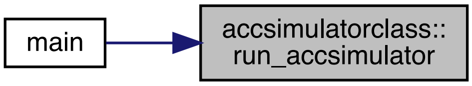
```
subroutine accsimulatorclass::run_accsimulator ( )
```

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(nbpmmax), target accsimulatorclass::beamln0
```

#### 5.1.3.5 beamln1

```
type (drifttube), 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 blnelem

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

#### 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 (fielddata), 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 isteerend

```
integer accsimulatorclass::isteerend
```

#### 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::nprow
```

#### 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