**foundation module**

The foundation module contains classes fundamental to accelerator physics and accelerator physics simulation.

**Classes**

*class* **Diagnostics\_write\_helper**

The [*Diagnostics\_write\_helper*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_diagnostics__write__helper) is a helper class for [*Diagnostics*](http://compacc.fnal.gov/~amundson/html/bunch.html#project0class_diagnostics) objects.

Serial Diagnostics\_write\_helpers write many updates to a single file. Non-serial Diagnostics\_write\_helpers write each update to a new file.

*Public Functions*

**Diagnostics\_write\_helper**(std::string const & filename, bool serial, [*Commxx\_sptr*](http://compacc.fnal.gov/~amundson/html/utils.html#project0commxx_8h_1a8e4a45ebebd319814e88c81a65f0dbd9) commxx\_sptr, std::string const & local\_dir, int writer\_rank = [*default\_rank*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_diagnostics__write__helper_1af8743503c95a93a5af773ae058bdac8a))

**Diagnostics\_write\_helper**()

int **get\_count**()

Get the count for non-serial writers.

void **set\_count**(int count)

Set the count for non-serial writers.

**Parameters**

* count -

the count

void **increment\_count**()

bool **write\_locally**()

int **get\_writer\_rank**()

[*Hdf5\_file\_sptr*](http://compacc.fnal.gov/~amundson/html/utils.html#project0hdf5__file_8h_1aabf20c82608aa7edaf304a96e5eb56d9) **get\_hdf5\_file\_sptr**()

void **finish\_write**()

template < class Archive >

void **serialize**(Archive & ar, const unsigned int version)

**~Diagnostics\_write\_helper**()

*Public Static Attributes*

const int **default\_rank**

const int **flush\_period**

*class* **Distribution**

[*Distribution*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_distribution) is a virtual base class for obtaining the next number or set of numbers from a sequence according to a limited set of shapes.

*Public Functions*

double **get**()

Get the next number in the sequence (between 0 and 1).

void **fill\_uniform**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) array, double min, double max)

Fill a one-dimensional array uniformly between min and max.

void **fill\_uniform**([*MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1a9326a326b0bb8363bac5c6dc1feba89b) array, double min, double max)

Alternate form for type compatibility.

void **fill\_unit\_gaussian**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) array)

Fill a one-dimensional array with Gaussian distribution of zero mean and unit standard deviation.

void **fill\_unit\_gaussian**([*MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1a9326a326b0bb8363bac5c6dc1feba89b) array)

Alternate form for type compatibility.

void **fill\_unit\_disk**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) x\_array, [*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) y\_array)

Fill two one-dimensional arrays such that (x,y) are distributed uniformly in the unit disk.

void **fill\_unit\_disk**([*MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1a9326a326b0bb8363bac5c6dc1feba89b) x\_array, [*MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1a9326a326b0bb8363bac5c6dc1feba89b) y\_array)

Alternate form for type compatibility.

**~Distribution**()

*class* **Four\_momentum**

[*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) provides conversion between various relativistic kinematic parameters.

*Public Functions*

**Four\_momentum**()

Default constructor for internal use only.

**Four\_momentum**(double mass)

Construct a [*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) in the rest frame.

**Parameters**

* mass -

in GeV/c^2

**Four\_momentum**(double mass, double total\_energy)

Construct a [*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) with the given total energy.

**Parameters**

* mass -

in GeV/c^2

* total\_energy -

in GeV

void **set\_total\_energy**(double total\_energy)

Set the total energy.

**Parameters**

* total\_energy -

in GeV

void **set\_kinetic\_energy**(double kinetic\_energy)

Set the kinetic energy.

**Parameters**

* kinetic\_energy -

in GeV

void **set\_momentum**(double momentum)

Set the momentum.

**Parameters**

* momentum -

in GeV/c

void **set\_gamma**(double gamma)

Set the relativistic gamma factor.

**Parameters**

* gamma -

unitless

void **set\_beta**(double beta)

Set the relativistic beta factor.

**Parameters**

* beta -

unitless

double **get\_mass**()

Get the mass in GeV/c^2.

double **get\_total\_energy**()

Get the total energy in GeV.

double **get\_kinetic\_energy**()

Get the kinetic energy in GeV.

double **get\_momentum**()

Get momentum in GeV/c.

double **get\_gamma**()

Get the relativistic gamma factor.

double **get\_beta**()

Get the relativistic beta factor.

bool **equal**([*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) const & four\_momentum, double tolerance)

Check equality to the given tolerance.

**Parameters**

* four\_momentum -

another [*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum)

* tolerance -

fractional accuracy for beta and gamma

template < class Archive >

void **serialize**(Archive & ar, const unsigned int version)

Serialization support.

*class* **Multi\_diagnostics**

[*Multi\_diagnostics*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_multi__diagnostics) contains a list of Diagnostics\_sptrs.

*Public Type*

typedef std::list< [*Diagnostics\_sptr*](http://compacc.fnal.gov/~amundson/html/bunch.html#project0diagnostics_8h_1a27a65c14c4e8fc1bc909d405ff0a0bf7) >:: [*iterator*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_multi__diagnostics_1a2695d6896dc135d9b4d9dc87e8559aeb) **iterator**

A convenience definition of the list iterator.

Not relevant for Python.

*Public Functions*

**Multi\_diagnostics**()

Construct an empty list of Diagnostics\_sptrs.

void **append**([*Diagnostics\_sptr*](http://compacc.fnal.gov/~amundson/html/bunch.html#project0diagnostics_8h_1a27a65c14c4e8fc1bc909d405ff0a0bf7) diagnostics\_sptr)

Append a Diagnostics\_sptr to the list.

**Parameters**

* diagnostics\_sptr -

the Diagnostics\_sptr

void **push\_back**([*Diagnostics\_sptr*](http://compacc.fnal.gov/~amundson/html/bunch.html#project0diagnostics_8h_1a27a65c14c4e8fc1bc909d405ff0a0bf7) diagnostics\_sptr)

The same as append included for notational consistency with C++.

**Parameters**

* diagnostics\_sptr -

the Diagnostics\_sptr

[*iterator*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_multi__diagnostics_1a2695d6896dc135d9b4d9dc87e8559aeb) **begin**()

A convenience definition of the list iterator begin.

Not relevant for Python.

[*iterator*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_multi__diagnostics_1a2695d6896dc135d9b4d9dc87e8559aeb) **end**()

A convenience definition of the list iterator end.

Not relevant for Python.

*class* **Random\_distribution**

[*Random\_distribution*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_random__distribution) provides a [*Distribution*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_distribution) of random numbers.

The random seed is maintained across multiple processors. The implementation uses random numbers from the GNU Scientific Library.

*Public Type*

**Generator enum**

*Values:*

* ranlxd2 -
* mt19937 -

*Public Functions*

**Random\_distribution**(unsigned long int seed, [*Commxx*](http://compacc.fnal.gov/~amundson/html/utils.html#project0class_commxx) const & comm, [*Generator*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_random__distribution_1a4611c834f746008000c0a17f98ac407e) generator = ranlxd2)

Construct a [*Random\_distribution*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_random__distribution).

**Parameters**

* seed -

The random number seed. If seed == 0, the seed is obtained from [*Random\_distribution::get\_default\_seed()*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_random__distribution_1a1c7faa6f961fe9551d81fdd7925cdb79).

* comm -

Distribute the seed across the processors in this communicator.

* generator -

The underlying random number generator to be used.

void **set\_seed**(unsigned long int seed)

Set the random number generator seed.

**Parameters**

* seed -

The seed.

unsigned long int **get\_original\_seed**()

Get the seed used to start the random number generator.

double **get**()

Get the next random number between 0 and 1.

void **fill\_uniform**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) array, double min, double max)

Fill a one-dimensional array uniformly between min and max.

void **fill\_unit\_gaussian**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) array)

Fill a one-dimensional array with Gaussian distribution of zero mean and unit standard deviation.

void **fill\_unit\_disk**([*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) x\_array, [*MArray1d\_view*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1af05e76759f25a0763a23cdd2129f4b34) y\_array)

Fill two one-dimensional arrays such that (x,y) are distributed uniformly in the unit disk.

**~Random\_distribution**()

*Public Static Functions*

unsigned long int **get\_default\_seed**(const char \* device = “/dev/urandom”)

Generate a random seed.

Attempt to read from device if present. Otherwise, use the system clock.

**Parameters**

* device -

Read from pathname device.

*class* **Reference\_particle**

[*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) stores the four momentum of the reference frame with respect to the lab frame (defined to be along the axis of the accelerator) as well as the six-dimensional state vector of the the reference particle in the reference frame.

Reference particle also keeps track of the total path length of the reference particle trajectory.

*Public Functions*

**Reference\_particle**()

Default constructor for internal use only.

**Reference\_particle**(int charge, double mass, double total\_energy)

Construct a [*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) with a given mass and total energy.

**Parameters**

* mass -

in GeV/c^2

* charge -

in units of e

* total\_energy -

in GeV in the lab frame

**Reference\_particle**(int charge, [*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) const & four\_momentum)

Construct a [*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) with a given four momentum.

**Parameters**

* charge -

in units of e

* four\_momentum -

in the lab frame

**Reference\_particle**(int charge, [*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) const & four\_momentum, [*Const\_MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1ae8c173d01855d23993771d2fd98632e9) state)

Construct a [*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) with a given four momentum and state in the reference frame.

**Parameters**

* charge -

in units of e

* four\_momentum -

in the lab frame

* state -

is a six-dimensional state vector

void **set\_four\_momentum**([*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) const & four\_momentum)

Set the four momentum.

**Parameters**

* four\_momentum -

in the lab frame

void **set\_state**([*Const\_MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1ae8c173d01855d23993771d2fd98632e9) state)

Set the state vector in the reference frame.

**Parameters**

* state -

is a six-dimensional state vector

void **set\_total\_energy**(double total\_energy)

Set the total energy.

**Parameters**

* total\_energy -

in GeV in the lab frame

void **increment\_trajectory**(double length)

Increment the trajectory length.

**Parameters**

* length -

in m

void **start\_repetition**()

Start a new repetition.

void **set\_trajectory**(int repetition, double repetition\_length, double s)

Manually set trajectory parameters.

**Parameters**

* repetition -

starting at 0

* repetition\_length -

in m

* s -

in m

int **get\_charge**()

Return the [*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) charge in units of e.

[*Four\_momentum*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_four__momentum) const & **get\_four\_momentum**()

Get the four momentum in the lab frame.

[*Const\_MArray1d\_ref*](http://compacc.fnal.gov/~amundson/html/utils.html#project0multi__array__typedefs_8h_1ae8c173d01855d23993771d2fd98632e9) **get\_state**()

Get the six-dimensional state vector in the reference frame.

double **get\_beta**()

Get the relativistic beta in the lab frame.

double **get\_gamma**()

Get the relativistic gamma in the lab frame.

double **get\_momentum**()

Get the momentum in GeV/c in the lab frame.

double **get\_total\_energy**()

Get the total energy in GeV in the lab frame.

double **get\_s**()

Get the total path length in m of the reference particle trajectory.

double **get\_s\_n**()

Get the distance traveled in m since the beginning of the current repetition.

int **get\_repetition**()

Get the number of repetition.

double **get\_repetition\_length**()

Get the repetition length in m.

bool **equal**([*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle) const & reference\_particle, double tolerance)

Check equality to the given tolerance.

**Parameters**

* reference\_particle -

another [*Reference\_particle*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_reference__particle)

* tolerance -

fractional accuracy

template < class Archive >

void **serialize**(Archive & ar, const unsigned int version)

Serialization support.

**Typedefs**

typedef boost::shared\_ptr< [*Diagnostics\_write\_helper*](http://compacc.fnal.gov/~amundson/html/foundation.html#project0class_diagnostics__write__helper) > **Diagnostics\_write\_helper\_sptr**