

# Package ‘libamtrack’

December 12, 2015

**Version** 0.6.3

**Date** 2015-12-12

**Title** Computational Routines for Proton and Ion Radiotherapy

**Author** Steffen Greilich and the libamtrack team

**Maintainer** Steffen Greilich <s.greilich@dkfz.de>

**Depends** R (>= 2.11.0)

**Suggests** lattice

**SystemRequirements** Gnu Scientific Library version >= 1.8

**Description** R interface to the open-source, ANSI C library 'libamtrack' (<http://libamtrack.dkfz.org>). 'libamtrack' provides computational routines for the prediction of detector response and radiobiological efficiency in heavy charged particle beams. It is designed for research in proton and ion dosimetry and radiotherapy. 'libamtrack' also includes many auxiliary physics routines for proton and ion beams. Original package and C-to-R conversion routines developed by Felix A. Klein.

**URL** [libamtrack.dkfz.org](http://libamtrack.dkfz.org)

**License** GPL (>= 3)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2015-12-12 18:49:11

## R topics documented:

libamtrack-package . . . . .	4
AT.A.from.particle.no . . . . .	8
AT.add.leading.zeros . . . . .	9
AT.add.trailing.zeros . . . . .	10
AT.atomic.weight.from.Z . . . . .	10
AT.average.A.from.composition . . . . .	11
AT.average.Z.from.composition . . . . .	11
AT.beam.par.physical.to.technical . . . . .	12
AT.beam.par.technical.to.physical . . . . .	13

AT.beta.from.E . . . . .	14
AT.CPPSC.alpha.and.beta . . . . .	15
AT.CSDA.range.g.cm2 . . . . .	16
AT.D.RDD.Gy . . . . .	17
AT.dose.Gy.from.fluence.cm2 . . . . .	18
AT.dose.weighted.E.MeV.u . . . . .	19
AT.dose.weighted.LET.MeV.cm2.g . . . . .	20
AT.E.from.beta . . . . .	21
AT.E.MeV.u.from.momentum.MeV.c.u . . . . .	22
AT.effective.charge.from.E.MeV.u . . . . .	23
AT.effective.Z.from.composition . . . . .	24
AT.electron.density.m3 . . . . .	24
AT.electron.density.m3.from.composition . . . . .	25
AT.electron.density.m3.from.material.no . . . . .	26
AT.energy.loss.from.lambda.gauss . . . . .	26
AT.energy.loss.from.lambda.landau . . . . .	27
AT.energy.loss.from.lambda.vavilov . . . . .	28
AT.energy.stragglng.after.slab.E.MeV.u . . . . .	28
AT.energy.stragglng.MeV2.cm2.g . . . . .	29
AT.fluence.cm2.from.dose.Gy . . . . .	30
AT.fluence.weighted.E.MeV.u . . . . .	31
AT.fluence.weighted.LET.MeV.cm2.g . . . . .	32
AT.FLUKA.particle.name.to.libamtrack.particle.name . . . . .	33
AT.FLUKA.read.USRBIN.mesh . . . . .	33
AT.FLUKA.read.USRBIN.regs . . . . .	34
AT.FLUKA.read.USRTRACK . . . . .	35
AT.gamma.from.E . . . . .	36
AT.gamma.response . . . . .	37
AT.Gauss.IDF . . . . .	39
AT.Gauss.PDF . . . . .	39
AT.get.materials.data . . . . .	40
AT.I.eV.from.composition . . . . .	41
AT.inverse.gamma.response . . . . .	41
AT.kappa . . . . .	42
AT.lambda.landau.from.energy.loss . . . . .	43
AT.lambda.max . . . . .	43
AT.lambda.mean . . . . .	44
AT.lambda.vavilov.from.energy.loss . . . . .	45
AT.Landau.IDF . . . . .	45
AT.Landau.PDF . . . . .	46
AT.Mass.Stopping.Power . . . . .	47
AT.Mass.Stopping.Power.with.no . . . . .	48
AT.material.name.from.material.no . . . . .	49
AT.material.no.from.material.name . . . . .	50
AT.max.E.transfer.MeV . . . . .	50
AT.max.E.transfer.MeV.new . . . . .	51
AT.max.electron.ranges.m . . . . .	52
AT.mean.energy.loss.keV . . . . .	53

AT.mean.number.of.tracks.contrib . . . . .	53
AT.momentum.MeV.c.u.from.E.MeV.u . . . . .	54
AT.nuclear.spin.from.particle.no . . . . .	55
AT.particle.name.from.particle.no . . . . .	55
AT.particle.no.from.particle.name . . . . .	56
AT.particle.no.from.Z.and.A . . . . .	57
AT.r.RDD.m . . . . .	57
AT.run.CPPSC.method . . . . .	58
AT.run.GSM.method . . . . .	60
AT.run.IGK.method . . . . .	63
AT.Rutherford.SDCS . . . . .	65
AT.set.user.material . . . . .	65
AT.set.user.material.from.composition . . . . .	66
AT.SPC.convert.to.DDD . . . . .	67
AT.SPC.export.DEDX . . . . .	67
AT.SPC.get . . . . .	68
AT.SPC.get.list . . . . .	69
AT.SPC.interpolate . . . . .	70
AT.SPC.read . . . . .	71
AT.SPC.spectrum.at.depth.g.cm2 . . . . .	75
AT.SPC.spectrum.at.depth.step . . . . .	76
AT.SPC.tapply . . . . .	76
AT.Stopping.Power . . . . .	79
AT.stopping.power.ratio . . . . .	80
AT.Stopping.Power.with.no . . . . .	81
AT.total.D.Gy . . . . .	82
AT.total.fluence.cm2 . . . . .	83
AT.translate.dose.into.DSB.distribution . . . . .	84
AT.Vavilov.IDF . . . . .	85
AT.Vavilov.PDF . . . . .	86
AT.xi.keV . . . . .	86
AT.Z.from.particle.no . . . . .	87
E.MeV.u . . . . .	87
er.model . . . . .	88
fluence.cm2.or.dose.Gy . . . . .	88
gamma.model . . . . .	89
material.name . . . . .	90
material.no . . . . .	90
number.of.field.components . . . . .	91
particle.name . . . . .	91
particle.no . . . . .	92
rdd.model . . . . .	92
stopping.power.source . . . . .	93

---

libamtrack-package      *libamtrack package*


---

## Description

This package is the R interface to the open-source, ANSI C library libamtrack. libamtrack provides computational routines for the prediction of detector response and relative biological efficiency in heavy charged particle beams. It is designed for research in proton and ion dosimetry and radiotherapy. libamtrack also provides many auxiliary physics routines for proton and ion beams. Please note that libamtrack is still under heavy development and so is the R interface. Function can be unstable especially when arguments are pushed out of their scope. If you experience any trouble your feed back is very appreciated: <s.greilich@dkfz.de>.

## Details

Package:	libamtrack
Version:	0.6.3 (Green Armadillo)
Date:	2015-12-12
Depends:	R (>= 2.12.0)
Suggests:	lattice
SystemRequirements:	gsl (optionally: cernlib to enable energy loss distributions)
License:	GPL (version 3 or later)

**FUNCTION INDEX: Efficiency / RBE routines:** These functions compute the relative efficiency / RBE of a mixed particle field according to a specific amorphous track model flavour and physics

<a href="#">AT.run.GSM.method</a>	Grid-summation ('checkerboard') method
<a href="#">AT.run.IGK.method</a>	Ion-gamma-kill ('Katz') method
<a href="#">AT.run.CPPSC.method</a>	Compound-Poison process with successive convolutions (CPP-SC, 'SPIFF') method
<a href="#">AT.CPPSC.alpha.and.beta</a>	Alpha, beta for ion beams as predicted by CPP-SC method

**Track-structure routines:** These functions handle underlying physic used in amorphous track modeling:

<a href="#">AT.D.RDD.Gy</a>	Dose distribution around a particle track
<a href="#">AT.r.RDD.m</a>	Inverse dose distribution around a particle track
<a href="#">AT.max.electron.ranges.m</a>	Maximum electron range / track width
<a href="#">AT.gamma.response</a>	Gamma / X ray response of a system
<a href="#">AT.inverse.gamma.response</a>	Inverse Gamma / X ray response of a system

**SPC routines:** These functions provide an interface to spectral depth data:

<a href="#">AT.SPC.read</a>	Read in spc data from file
<a href="#">AT.SPC.spectrum.at.depth.step</a>	Extract spectrum at given depth step
<a href="#">AT.SPC.spectrum.at.depth.g.cm2</a>	Extract spectrum at given depth (will use closest step)
<a href="#">AT.SPC.tapply</a>	Applies a function to all depths of spc data

**FLUKA routines:** These function provide an interface to FLUKA output files:

<code>AT.FLUKA.read.USRBIN.mesh</code>	Reads USRBIN output (Cartesian mesh, also for multiple runs)
<code>AT.FLUKA.read.USRBIN.regs</code>	Reads USRBIN output (regions, also for multiple runs).
<code>AT.FLUKA.read.USRTRACK</code>	Reads USRTRACK output (energy spectra, also for multiple runs)
<code>AT.FLUKA.particle.name.to.libamtrack.particle.name</code>	Converts FLUKA particle names to libamtrack conventions.

**Physics routines:** These functions handle the physics of proton and ion beams needed in libamtrack. Stopping-power routines:

<code>AT.Mass.Stopping.Power</code>	Electronic mass stopping power (data source by name)
<code>AT.Stopping.Power</code>	Electronic stopping power (data source by name)
<code>AT.Mass.Stopping.Power.with.no</code>	Electronic mass stopping power (data source by number)
<code>AT.Stopping.Power.with.no</code>	Electronic stopping power (data source by number)
<code>AT.stopping.power.ratio</code>	Computes stopping power ratios, also for mixed fields
<code>TODO:</code>	Re-enable CSDA functions
<code>AT.CSDA.range.g.cm2</code>	CSDA range using PSTAR data

Mean LET / energy in mixed fields routines:

<code>AT.fluence.weighted.LET.MeV.cm2.g</code>	Computes fluence-weighted LET
<code>AT.dose.weighted.LET.MeV.cm2.g</code>	Computes dose-weighted LET
<code>AT.fluence.weighted.E.MeV.u</code>	Computes fluence-weighted mean energy
<code>AT.dose.weighted.E.MeV.u</code>	Computes dose-weighted mean energy

Dose / fluence conversions:

<code>AT.dose.Gy.from.fluence.cm2</code>	Compute dose(s) for given fluence(s) and particle(s)
<code>AT.fluence.cm2.from.dose.Gy</code>	Compute fluence(s) given dose(s) and particle(s)
<code>AT.total.D.Gy</code>	Computes total dose for a mixed field
<code>AT.total.fluence.cm2</code>	Computes total fluence for a mixed field

Beam related routines:

<code>AT.beam.par.technical.to.physical</code>	For double Gaussian beam, converts FWHM and particle number into fluence and sigma
<code>AT.beam.par.physical.to.technical</code>	Inverse, converts fluence and sigma width into FWHM and particle number

Misc physics routines:

<code>AT.momentum.MeV.c.u.from.E.MeV.u</code>	Momentum from kinetic energy
<code>AT.E.MeV.u.from.momentum.MeV.c.u</code>	Kinetic energy from momentum
<code>AT.effective.charge.from.E.MeV.u</code>	Effective charge of an ion depending on its kinetic energy
<code>AT.max.E.transfer.MeV</code>	Max energy transferred from an ion to secondary electrons
<code>AT.mean.number.of.tracks.contrib</code>	Mean number of tracks that desoposite dose in a representative point
<code>AT.Rutherford.SDCS</code>	Rutherford cross section of energy transferred to sec. electrons
<code>AT.beta.from.E</code>	Relativistic beta of an ion with
<code>AT.E.from.beta</code>	Kinetic energy for given beta
<code>AT.gamma.from.E</code>	Relativistic gamma of an ion

## Other routines:

AT.particle.name.from.particle.no	Converts particle index numbers into particle names
AT.particle.no.from.particle.name	Converts particle names into particle index numbers
AT.particle.no.from.Z.and.A	Returns particle index number for given mass and atomic number
AT.A.from.particle.no	Returns mass number for given particle number
AT.Z.from.particle.no	Returns atomic number for given particle number
AT.nuclear.spin.from.particle.no	Returns nuclear spin for given particle number
AT.electron.density.m3	Returns electron density from average Z and A
AT.electron.density.m3.from.material.no	Returns electron density for given material
AT.material.name.from.material.no	Converts material index numbers into material names
AT.material.no.from.material.name	Converts material names into material index numbers
AT.electron.density.m3.from.composition	Computes electron density from material composition
AT.average.A.from.composition	Computes average mass number from material composition
AT.average.Z.from.composition	Computes average atomic number from material composition
AT.effective.Z.from.composition	Computes effective atomic number from material composition
AT.I.eV.from.composition	Computes average I value from material composition
AT.set.user.material	Sets properties of user defined material (CAVE: only valid until library is f
AT.set.user.material.from.composition	Sets properties of user defined material from elemental composition (CAV
AT.get.materials.data	Returns properties of pre-defined materials

## Helper routines shipped with libamtrack:

AT.add.leading.zeros	Adds leading zeros to a character string representing a number
AT.add.trailing.zeros	Adds trailing zeros to a character string representing a number

**Author(s)**

C2R autoconversion developed by: Felix Klein <fklein@embl.de> Package maintainer: Steffen Greilich <s.greilich@dkfz.de>

**References**

*Greilich, Grzanka, Bassler, Andersen and Jakel, Amorphous track models: A numerical comparison study, doi:10.1016/j.radmeas.2010.05.039*

**See Also**

<http://libamtrack.dkfz.org>

**Examples**

```
#####
##### 1. LET #####
cat("Compute the LET (in keV/um) of a 270 MeV/u carbon ion in Aluminum\n")
cat("using the PSTAR stopping power data:\n")
AT.Stopping.Power( E.MeV.u      = 270,
                  particle.no    =
AT.particle.no.from.particle.name("12C"),
                  material.no    =
```

```

AT.material.no.from.material.name("Aluminum"),
      stopping.power.source = "PSTAR")

cat("... and in water:\n")
AT.Stopping.Power( E.MeV.u           = 270,
                  particle.no        =
AT.particle.no.from.particle.name("12C"),
                  material.no        =
AT.material.no.from.material.name("Water, Liquid"),
                  stopping.power.source = "PSTAR")
#####

# 2. DOSE AROUND A TRACK
cat
("Compare the Geiss parametrization for protons and Carbon at different energies:\n"
)
df <- expand.grid( E.MeV.u           = 10^seq(0, 3, length.out = 4),
                  particle.no        = c(1001,6012),
                  # protons and carbons
                  r.m                 = 10^seq(-9, -2, length.out =
100), # from 1 nm to 1 cm in 100 steps
                  material.no        = 2,
                  # Aluminium Oxide
                  rdd.model           = 3,
                  # Geiss parametrization
                  rdd.parameter       = 5e-8,
                  # Fixed core size of 50 nm
                  er.model            = 4,
                  # Geiss track width parametrization
                  D.Gy                = 0)

ii <- df$particle.no == 1001
# Add particle names
df$particle.name <- "Carbon-12"
df$particle.name[ii] <- "Protons"
for (i in 1:nrow(df)){
  # Loop through particles/energies
  df$D.Gy[i] <- AT.D.RDD.Gy( r.m           = df$r.m[i],
                           E.MeV.u       = df$E.MeV.u[i],
                           particle.no    = df$particle.no[i],
                           material.no    = df$material.no[i],
                           rdd.model      = df$rdd.model[i],
                           rdd.parameter  =
df$rdd.parameter[i],
                           er.model       = df$er.model[i],
                           stopping.power.source.no
= 2)[[1]] # use PSTAR data
}
lattice::xyplot( log10(D.Gy) ~ log10(r.m)|particle.name,
                # Plot
                df,
                type = 'l',

```

```

groups      = E.MeV.u,
auto.key    = TRUE)[c(2,1)]

#####
##### 3. DETECTOR EFFICIENCY #####
cat("Compute the relative efficiency of Alanine in 10 MeV protons\n")
cat("Waligorskis version of the Katz' model\n")
AT.run.IGK.method( particle.no           = 1001,
  # namely protons with
    E.MeV.u                             = 10,
  # 10 MeV/u
    fluence.cm2.or.dose.Gy              = c(-1.0),
  # delivering 1 Gy
    material.no                          = 5,
  # i.e. Alanine
    rdd.model                             = 4,

# Katz parametrization of radial dose distribution with simplified extended
# targets
    rdd.parameter                        = c(5e-8,1e-10),
  # with 50 nm target size and 1e-10 dose minimum
    er.model                             = 2,
  # Butts&Katz parametrization of track radius
    gamma.model                          = 2,
  # Use general target/hit model but here...
    gamma.parameters                     = c(1,500,1,1,0),
  # ...as exponential saturation with characteristic dose 500 Gy
    saturation.cross.section.factor      = 1.4,
  # factor to take 'brush' around track into account
    write.output                          = TRUE,
  # write a log file
    stopping.power.source.no             = 2)
# use PSTAR data

```

---

AT.A.from.particle.no *AT.A.from.particle.no*

---

## Description

Returns mass number for given particle number

## Usage

AT.A.from.particle.no(particle.no)

## Arguments

particle.no      particle index number (array of size n) (see also [particle.no](#)).



**Value**

A	mass number (array of size n)
return	return

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataParticle.c#L53](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataParticle.c#L53)

---

`AT.add.leading.zeros`    *AT.add.leading.zeros*

---

**Description**

Adds leading zeros to a number and represent it in a string of fixed length.

**Usage**

```
AT.add.leading.zeros( x, digits = 5)
```

**Arguments**

x	number
digits	length of resulting string. Should be larger that log10(x).

**Value**

Character string with digits characters.

**Examples**

```
# Represent 99 as '00099'  
AT.add.leading.zeros(x = 9, digits = 5)
```

---

AT.add.trailing.zeros *AT.add.trailing.zeros*

---

### Description

Adds trailing zeros to a number and represent it in a string of fixed length.

### Usage

```
AT.add.trailing.zeros( x, digits = 5)
```

### Arguments

x	number
digits	length of resulting string.

### Value

Character string with digits characters.

### Examples

```
# Represent 99.1 as '99.100'
AT.add.trailing.zeros(x = 99.1, digits = 5)
```

---

AT.atomic.weight.from.Z  
*AT.atomic.weight.from.Z*

---

### Description

Returns atomic weight for given Z

### Usage

```
AT.atomic.weight.from.Z(Z)
```

### Arguments

Z	atomic number (array of size n).
---	----------------------------------

### Value

atomic.weight	atomic weight (array of size n)
return	return

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataParticle.c#L109](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataParticle.c#L109)

---

AT.average.A.from.composition  
*AT.average.A.from.composition*

---

**Description**

Computes the average mass number for a given material composition

**Usage**

AT.average.A.from.composition(A, weight.fraction)

**Arguments**

A                    mass numbers of constituents (array of size n).  
weight.fraction    relative fractions of weight of constituents (array of size n).

**Value**

average.A          average A

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L386](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L386)

---

AT.average.Z.from.composition  
*AT.average.Z.from.composition*

---

**Description**

Computes the average atomic number for a given material composition

**Usage**

AT.average.Z.from.composition(Z, weight.fraction)

**Arguments**

Z                    atomic numbers of constituents (array of size n).  
 weight.fraction       relative fractions of weight of constituents (array of size n).

**Value**

average.Z            average Z

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L403](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L403)

---

AT.beam.par.physical.to.technical

*AT.beam.par.physical.to.technical*

---

**Description**

Converts physical beam parameters of a symmetric, double lateral Gaussian shape beam, i.e. central (=peak) fluence and width (= 1 standard deviation) to technical, accelerator parameters, i.e. total number of particles and FWHM

**Usage**

AT.beam.par.physical.to.technical(fluence.cm2, sigma.cm)

**Arguments**

fluence.cm2        fluence in beam center (array of size n).  
 sigma.cm            beam width stdev (array of size n).

**Value**

N                    resulting absolute particle numbers (array of size n)  
 FWHM.mm            resulting FWHMs (in mm) (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L430](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L430)

**Examples**

```
# Technical parameters for a double Gaussian beam
# of Carbon ions at 270 MeV/u in water with
# 0.5 cm sigma width and 1 Gy in the peak
AT.beam.par.physical.to.technical( fluence.cm2 = AT.fluence.cm2.from.dose.Gy(
  E.MeV.u      = 270,

  D.Gy        = 1.0,

  particle.no  = AT.particle.no.from.particle.name("12C"),

  material.no  = AT.material.no.from.material.name("Water, Liquid"),

  stopping.power.source.no = 2),

  sigma.cm    = 0.5)
```

---

```
AT.beam.par.technical.to.physical
    AT.beam.par.technical.to.physical
```

---

**Description**

Converts technical, accelerator parameters of a symmetric, double lateral Gaussian shape beam, i.e. total number of particles and FWHM to physical beam parameters, i.e. central (=peak) fluence and width (= 1 standard deviation)

**Usage**

```
AT.beam.par.technical.to.physical(N, FWHM.mm)
```

**Arguments**

N	absolute particle numbers (array of size n).
FWHM.mm	FWHMs (in mm) (array of size n).

**Value**

fluence.cm2	resulting fluence in beam center (array of size n)
sigma.cm	resulting beam width stdev (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L443](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L443)

**Examples**

```
# Get peak dose of a 142.66 MeV protons in Alox
# from technical beam parameters
peak.fluence.cm2 <- AT.beam.par.technical.to.physical( N      = 3.2e8,
                                                    FWHM.mm = 15.2)[1]

AT.dose.Gy.from.fluence.cm2( E.MeV.u      = 142.66,
                             particle.no =
  AT.particle.no.from.particle.name("1H"),
                             material.no =
  AT.material.no.from.material.name("Aluminum Oxide"),
                             fluence.cm2 = peak.fluence.cm2,
                             stopping.power.source.no = 2)
```

---

AT.beta.from.E	<i>AT.beta.from.E</i>
----------------	-----------------------

---

**Description**

Returns relativistic speed for many particles

**Usage**

```
AT.beta.from.E(E.MeV.u)
```

**Arguments**

E.MeV.u            vector of energies of particle per nucleon [MeV] (array of size n) (see also [E.MeV.u](#)).

**Value**

beta                vector of relative particle speed  $\beta = v/c$  (array of size n)  
status               status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L34](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L34)

**Examples**

```
# Compute beta between 1 and 1000 MeV/u:
AT.beta.from.E( E.MeV.u      = c(1,10,100,1000))
```

---

 AT.CPPSC.alpha.and.beta

*AT.CPPSC.alpha.and.beta*


---

### Description

Returns alpha and beta for an ion survival curve predicted by the CPPSC model. The arguments are directly passed on to the `AT.run.CPPSC.method` routine, some are provided default values here to simplify the use. !! THIS ROUTINE IS LIMITED TO THE LIN-QUAD MODEL BUT SHOULD BE GENERALIZED FOR ANY GAMMA REPNSE, BOTH IN TERMS OF INPUT X RAY PARAMETERS \*AND\* OF RETURNED (FITTED) ION PARAMETERS!! Cave: Even if input X ray response is LQ with cut dose, the resulting LQ will be purely linear-quadratic without high-dose linear characteristic.

### Usage

```
AT.CPPSC.alpha.and.beta(E.MeV.u, particle.no, fluence.cm2.or.dose.Gy,
  material.no, RDD.model, RDD.parameters, ER.model, gamma.model,
  gamma.parameters, N2 = 20, fluence.factor = 1.0, write.output = FALSE,
  shrink.tails = TRUE, shrink.tails.under = 1e-30, adjust.N2 = TRUE)
```

### Arguments

E.MeV.u	Energy
particle.no	Particle index
fluence.cm2.or.dose.Gy	Mean dose - will be varied for survival / response in a range 0.1x-10x.
material.no	Material index
RDD.model	RDD index
RDD.parameters	RDD parameters
ER.model	ER index
gamma.model	gamma model index, must be LQ = 5
gamma.parameters	gamma parameters, here alpha, beta, Dcut
N2	defaulted
fluence.factor	defaulted
write.output	defaulted
shrink.tails	defaulted
shrink.tails.under	defaulted
adjust.N2	defaulted

**Value**

List with following items

alpha	alpha for ion response curve
beta	beta for ion response curve
df	data frame with complete ion response curve as computed to find alpha and beta including residuals fo that fit.

**Examples**

```
# None yet
```

---

```
AT.CSDA.range.g.cm2  AT.CSDA.range.g.cm2
```

---

**Description**

Computes the CSDA range using the PSTAR data

**Usage**

```
AT.CSDA.range.g.cm2(E.initial.MeV.u, E.final.MeV.u, particle.no,
  material.no)
```

**Arguments**

E.initial.MeV.u	initial energy of particle per nucleon (array of size n).
E.final.MeV.u	final energy of particle per nucleon (array of size n).
particle.no	particle index (array of size n) (see also <a href="#">particle.no</a> ).
material.no	material index (see also <a href="#">material.no</a> ).

**Value**

```
CSDA.range.cm2.g
  resulting range (array of size n)
```

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataRange.c#L101](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataRange.c#L101)



**Examples**

```
# Range of 270 MeV/u carbon ions and 142 MeV/u protons in water
#AT.CSDA.range.g.cm2( E.MeV.u           = c(270, 142),

#           particle.no           =
# AT.particle.no.from.particle.name(c("12C", "1H")),

#           material.no           =
# AT.material.no.from.material.name("Water, Liquid"))
```

---

AT.D.RDD.Gy

*AT.D.RDD.Gy*


---

**Description**

Returns local dose as a function of distance `r_m` for a given radial dose distribution model

**Usage**

```
AT.D.RDD.Gy(r.m, E.MeV.u, particle.no, material.no, rdd.model,
rdd.parameter, er.model, stopping.power.source.no)
```

**Arguments**

<code>r.m</code>	distance [m] (array of size n).
<code>E.MeV.u</code>	particle (ion) energy per nucleon [MeV/u] (single number, no mixed fields) (see also <a href="#">E.MeV.u</a> ).
<code>particle.no</code>	particle code number (single number, no mixed fields) (see also <a href="#">particle.no</a> ).
<code>material.no</code>	material code number (single number, no mixed fields) (see also <a href="#">material.no</a> ).
<code>rdd.model</code>	radial dose distribution model index (see also <a href="#">rdd.model</a> ).
<code>rdd.parameter</code>	radial dose distribution model parameters (array of size 4).
<code>er.model</code>	electron range / track with model index (see also <a href="#">er.model</a> ).
<code>stopping.power.source.no</code>	TODO (see also <a href="#">stopping.power.source.no</a> ).

**Value**

<code>D.RDD.Gy</code>	dose [Gy] (array of size n)
<code>status</code>	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_RDD.c#L485](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_RDD.c#L485)

**Examples**

```

# Compute dose in several distances of an 100 MeV/u neon ion in water
# according to 'Site' parametrization
AT.D.RDD.Gy(  r.m          = 10^(-9:-4),
             E.MeV.u      = 100,
             particle.no   = 10020,
             material.no   = 1,
             rdd.model     = 4,
             rdd.parameter = c(5e-8, 1e-10),
             er.model      = 2,
             stopping.power.source.no = 2)

# Compare the Geiss parametrization of RDD for protons and Carbon ions at
# different energies:
df <- expand.grid( E.MeV.u      = 10^seq(0, 3, length.out = 4),
                  # from 1 to 1000 MeV/u in 4 steps
                  particle.no   = c(1001,6012),

                  # protons and carbons
                  r.m           = 10^seq(-9, -2, length.out = 100),
                  # from 1 nm to 1 cm in 100 steps
                  material.no   = 2,

                  # Aluminium Oxide
                  rdd.model     = 3,

                  # Geiss parametrization
                  rdd.parameter = 5e-8,

                  # Fixed core size of 50 nm
                  er.model      = 4,

                  # Geiss track width parametrization
                  D.Gy          = 0)

# For later use
ii <- df$particle.no == 1001
# Add particle names
df$particle.name <- "Carbon-12"
df$particle.name[ii] <- "Protons"
for (i in 1:nrow(df)){
  # Loop through particles/energies
  df$D.Gy[i] <- AT.D.RDD.Gy( r.m          = df$r.m[i],
                           E.MeV.u      = df$E.MeV.u[i],
                           particle.no   = df$particle.no[i],
                           material.no   = df$material.no[i],
                           rdd.model     = df$rdd.model[i],
                           rdd.parameter = df$rdd.parameter[i],
                           er.model      = df$er.model[i],
                           stopping.power.source.no = 2)$D.RDD.Gy
}

```

**Description**

Returns dose in Gy for each given particle

**Usage**

```
AT.dose.Gy.from.fluence.cm2(E.MeV.u, particle.no, fluence.cm2,
    material.no, stopping.power.source.no)
```

**Arguments**

E.MeV.u            energy of particles in the mixed particle field (array of size n) (see also [E.MeV.u](#)).

particle.no        type of the particles in the mixed particle field (array of size n) (see also [particle.no](#)).

fluence.cm2       fluence for each particle type (array of size n).

material.no        material index (see also [material.no](#)).

stopping.power.source.no    stopping power source index (see also [stopping.power.source.no](#)).

**Value**

dose.Gy            be allocated by the user which will be used to return the results (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L344](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L344)

**Examples**

```
# Compute dose from protons, He-3, C-12 and O-16 at
# same energy and fluence in air
AT.dose.Gy.from.fluence.cm2( E.MeV.u        = c(10, 10, 10, 10),
                             fluence.cm2    = c(1e7, 1e7, 1e7, 1e7),
                             particle.no    = c(1001, 2003, 6012, 8016),
                             material.no    = 7,
                             stopping.power.source.no = 2)
```

---

AT.dose.weighted.E.MeV.u

*AT.dose.weighted.E.MeV.u*

---

**Description**

Computes the dose-weighted average energy of a particle field Needed by SuccessiveConvolutions

**Usage**

```
AT.dose.weighted.E.MeV.u(E.MeV.u, particle.no, fluence.cm2,
    material.no, stopping.power.source.no)
```

**Arguments**

E.MeV.u            energy of particles in the mixed particle field (array of size `number.of.field.components`) (see also `E.MeV.u`).

particle.no        particle index (array of size `number.of.field.components`) (see also `particle.no`).

fluence.cm2        fluences of particles in the mixed particle field (array of size `number.of.field.components`).

material.no        material index (see also `material.no`).

stopping.power.source.no    **TODO** (see also `stopping.power.source.no`).

**Value**

dose-weighted    dose-weighted

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L600](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L600)

**Examples**

```
# fluence- and dose-weighted mean energy for a simple mixed field
# of high and low (99/1) energy protons
AT.fluence.weighted.E.MeV.u( E.MeV.u      = c(100, 1),
                             fluence.cm2  = c(99e8, 1e8))
AT.dose.weighted.E.MeV.u( E.MeV.u      = c(100, 1),
                          particle.no    = c(1001, 1001),
                          fluence.cm2    = c(99e8, 1e8),
                          material.no    = 1,                               # water
                          stopping.power.source.no = 2)
```

---

AT.dose.weighted.LET.MeV.cm2.g

*AT.dose.weighted.LET.MeV.cm2.g*

---

**Description**

Computes the dose-weighted average LET of a particle field

**Usage**

```
AT.dose.weighted.LET.MeV.cm2.g(E.MeV.u, particle.no, fluence.cm2,
                                material.no, stopping.power.source.no)
```

**Arguments**

E.MeV.u            energy of particles in the mixed particle field (array of size `number.of.field.components`) (see also `E.MeV.u`).

particle.no        particle index (array of size `number.of.field.components`) (see also `particle.no`).

fluence.cm2        fluences of particles in the mixed particle field (array of size `number.of.field.components`).

material.no        material index (see also `material.no`).

stopping.power.source.no    TODO (see also `stopping.power.source.no`).

**Value**

dose-weighted    dose-weighted

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L674](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L674)

**Examples**

```
# fluence- and dose-weighted LET for a simple mixed field
# of high and low (99/1) energy protons
AT.fluence.weighted.LET.MeV.cm2.g( E.MeV.u      = c(100, 5),
                                   particle.no   = c(1001, 1001),
                                   fluence.cm2    = c(99e8, 1e8),
                                   material.no    = 1,                                # water
                                   stopping.power.source.no = 2)
AT.dose.weighted.LET.MeV.cm2.g( E.MeV.u      = c(100, 5),
                                 particle.no   = c(1001, 1001),
                                 fluence.cm2    = c(99e8, 1e8),
                                 material.no    = 1,                                # water
                                 stopping.power.source.no = 2)
```

---

AT.E.from.beta

*AT.E.from.beta*


---

**Description**

Returns energy per nucleon of particle with relativistic speed beta

**Usage**

AT.E.from.beta(beta)

**Arguments**

beta                vector of relative particle speed beta = v/c (array of size n).

**Value**

E.MeV.u	vector of energies of particle per nucleon [MeV] (array of size n)
status	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L71](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L71)

**Examples**

```
# Compute energies for betas between 0.1 and 0.99:
AT.E.from.beta( beta = c(0.1, 0.1*(1:9), 0.99))
```

---

```
AT.E.MeV.u.from.momentum.MeV.c.u
      AT.E.MeV.u.from.momentum.MeV.c.u
```

---

**Description**

Returns energy per nucleon for particles with given momentum per nucleon

**Usage**

```
AT.E.MeV.u.from.momentum.MeV.c.u(momentum.MeV.c.u)
```

**Arguments**

momentum.MeV.c.u	vector of particle momenta per nucleon [MeV/c], (array of size n).
------------------	--

**Value**

E.MeV.u	vector of energies of particle per nucleon [MeV], (array of size n)
status	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L112](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L112)

**Examples**

```
# Get kinetic energy for a 502 MeV/c antiproton beam (CERN AD/ACE experiment):
AT.E.MeV.u.from.momentum.MeV.c.u( momentum.MeV.c.u = 502)
```

---

 AT.effective.charge.from.E.MeV.u

*AT.effective.charge.from.E.MeV.u*


---

## Description

Effective charge according to Barkas-Bethe-approximation: for particles with given kinetic energy per nucleon

## Usage

```
AT.effective.charge.from.E.MeV.u(E.MeV.u, particle.no)
```

## Arguments

E.MeV.u            vector of energies of particle per nucleon [MeV] (array of size n) (see also [E.MeV.u](#)).

particle.no        type of the particles in the mixed particle field (array of size n) (see also [particle.no](#)).

## Value

effective.charge            Effective charge according to Barkas-Bethe-approximation (array of size n)

status                      status

## See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L149](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L149)

## Examples

```
# Charge pick-up of several nuclids depending on ion energy
df <- data.frame( E.MeV.u      = 10^seq(-1, 2, length.out = 50),
                 particle.name = c("1H", "3He", "6Li", "12C",
                                   "16O"),
                 effective.charge = 0)
for(i in 1:nrow(df)){
  df$effective.charge[i] <- AT.effective.charge.from.E.MeV.u( E.MeV.u      =
df$E.MeV.u[i],
                                                             particle.no =
AT.particle.no.from.particle.name(df$particle.name[i])[1])
}
```

---

AT.effective.Z.from.composition

*AT.effective.Z.from.composition*

---

### Description

Computes the effective atomic number for a given material composition

### Usage

```
AT.effective.Z.from.composition(Z, weight.fraction,
    electron.densities.cm3, exponent)
```

### Arguments

Z	atomic numbers of constituents (array of size n).
weight.fraction	relative fractions of weight of constituents (array of size n).
electron.densities.cm3	if not zero, weight fractions will additionally include electron densities per volume (array of size n).
exponent	exponent for additivity rule reflecting the photon energy regime (usually 3.5 at ~ 100 kV).

### Value

effective.Z	effective Z
-------------	-------------

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L420](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L420)

---

AT.electron.density.m3

*AT.electron.density.m3*

---

### Description

Returns electron density from average A and Z

### Usage

```
AT.electron.density.m3(density.g.cm3, average.A, average.Z)
```



**Arguments**

density.g.cm3 material density in g/cm<sup>3</sup> (array of size n).  
 average.A average mass number (array of size n).  
 average.Z average atomic number (array of size n).

**Value**

electron.density.m3  
 electron density in 1/m<sup>3</sup> (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L350](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L350)

---

AT.electron.density.m3.from.composition  
*AT.electron.density.m3.from.composition*

---

**Description**

Computes the electron density for a given material composition

**Usage**

AT.electron.density.m3.from.composition(density.g.cm3, Z, A,  
 weight.fraction)

**Arguments**

density.g.cm3 physical density (in g per cm<sup>3</sup>) of material.  
 Z atomic numbers of constituents (array of size n).  
 A mass numbers of constituents (array of size n).  
 weight.fraction relative fractions of weight of constituents (array of size n).

**Value**

electron.density.m3  
 electron density per m<sup>3</sup>

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L364](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L364)

---

AT.electron.density.m3.from.material.no  
*AT.electron.density.m3.from.material.no*

---

**Description**

Get electron density [1/m<sup>3</sup>] for materials

**Usage**

AT.electron.density.m3.from.material.no(material.no)

**Arguments**

material.no      material indices (array of size n) (see also [material.no](#)).

**Value**

electron.density.m3  
 electron densities per m<sup>3</sup> (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L327](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L327)

---

AT.energy.loss.from.lambda.gauss  
*AT.energy.loss.from.lambda.gauss*

---

**Description**

Computes the energy loss from the lambda parameter of the Gauss distribution for compatibility with CERN W5013 No effective projectile charge is considered!

**Usage**

AT.energy.loss.from.lambda.gauss(lambda.gauss, E.MeV.u, particle.no,  
 material.no, slab.thickness.um)

**Arguments**

lambda.gauss      Gauss lambda (array of size n).  
 E.MeV.u          energy of particle per nucleon (array of size n) (see also [E.MeV.u](#)).  
 particle.no      particle index (array of size n) (see also [particle.no](#)).  
 material.no      material index (see also [material.no](#)).  
 slab.thickness.um      slab thickness in um (array of size n).

**Value**

energy.loss.keV  
(array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L607](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L607)

---

AT.energy.loss.from.lambda.landau

*AT.energy.loss.from.lambda.landau*

---

**Description**

Computes the energy loss from the lambda parameter of the Landau distribution acc. to CERN W5013 No effective projectile charge is considered!

**Usage**

AT.energy.loss.from.lambda.landau(lambda.landau, E.MeV.u, particle.no,  
material.no, slab.thickness.um)

**Arguments**

lambda.landau Landau lambda (array of size n).  
E.MeV.u energy of particle per nucleon (array of size n) (see also [E.MeV.u](#)).  
particle.no particle index (array of size n) (see also [particle.no](#)).  
material.no material index (see also [material.no](#)).  
slab.thickness.um  
slab thickness in um (array of size n).

**Value**

energy.loss.keV  
(array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L181](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L181)

---

AT.energy.loss.from.lambda.vavilov

*AT.energy.loss.from.lambda.vavilov*

---

### Description

Computes the energy loss from the lambda parameter of the Vavilov distribution acc. to CERN W5013 No effective projectile charge is considered!

### Usage

AT.energy.loss.from.lambda.vavilov(lambda.vavilov, E.MeV.u,  
particle.no, material.no, slab.thickness.um)

### Arguments

lambda.vavilov Vavilov lambda (array of size n).  
E.MeV.u energy of particle per nucleon (array of size n) (see also [E.MeV.u](#)).  
particle.no particle index (array of size n) (see also [particle.no](#)).  
material.no material index (see also [material.no](#)).  
slab.thickness.um  
slab thickness in um (array of size n).

### Value

energy.loss.keV  
(array of size n)

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L218](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L218)

---

AT.energy.straggling.after.slab.E.MeV.u

*AT.energy.straggling.after.slab.E.MeV.u*

---

### Description

Get energy spread of an ion beam after traversing a material slab according to Bohr's classical theory. Bohr, N. (1915), Phil. Mag. 30, 581ff, see also Evans, R.D. (1955), The atomic nucleus, McGraw Hill, New York, p. 661 Please note that the effective charge is assumed to be constant over the material slab If this is not the case you should apply this routine multiple times to subslices

**Usage**

AT.energy.stragglng.after.slabs.E.MeV.u(E.MeV.u, particle.no,  
material.no, slab.thickness.m, initial.sigma.E.MeV.u)

**Arguments**

E.MeV.u            vector of energies of particle per nucleon [MeV] (array of size n) (see also [E.MeV.u](#)).

particle.no        type of the particles in the mixed particle field (array of size n) (see also [particle.no](#)).

material.no        index number for slab material (see also [material.no](#)).

slab.thickness.m    thickness of slab in m.

initial.sigma.E.MeV.u    energy spread - 1 sigma - before traversing the slab - can be 0 (array of size n).

**Value**

sigma.E.MeV.u    energy spread - 1 sigma - after traversing the slab (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L318](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L318)

---

AT.energy.stragglng.MeV2.cm2.g

*AT.energy.stragglng.MeV2.cm2.g*

---

**Description**

Get energy spread with depth according to Bohr's classical theory Bohr, N. (1915), Phil. Mag. 30, 581ff, see also Evans, R.D. (1955), The atomic nucleus, McGraw Hill, New York, p. 661 In the literature  $dsE^2dz$  is often given in units  $\text{ergs}^2/\text{cm}$ . Here we report it mass-normalized  $\text{MeV}^2 \cdot \text{cm}^2/\text{g}$  Since the effective charge of the particle enters the equation, particle types and energies have to be given The equation is however limited to energies  $> 10 \text{ MeV/u}$  and not too heavy ions TODO: add William extension for relativistic effects (Williams, E.J. (1945), Revs. Mod. Phys. 17, 217ff)

**Usage**

AT.energy.stragglng.MeV2.cm2.g(E.MeV.u, particle.no, material.no)

**Arguments**

E.MeV.u            vector of energies of particle per nucleon [MeV] (array of size n) (see also [E.MeV.u](#)).

particle.no        type of the particles in the mixed particle field (array of size n) (see also [particle.no](#)).

material.no        index number for slab material (see also [material.no](#)).

**Value**

dsE2dz.MeV2.cm2.g

Increase of energy straggling variance  $\sigma.E^2$  per unit length of material (array of size n)**See Also**View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L301](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L301)

---

AT.fluence.cm2.from.dose.Gy*AT.fluence.cm2.from.dose.Gy*

---

**Description**

Returns fluence in 1/cm2 for each given particle

**Usage**

AT.fluence.cm2.from.dose.Gy(E.MeV.u, particle.no, D.Gy, material.no, stopping.power.source.no)

**Arguments**

E.MeV.u	energy of particles in the mixed particle field (array of size n) (see also <a href="#">E.MeV.u</a> ).
particle.no	type of the particles in the mixed particle field (array of size n) (see also <a href="#">particle.no</a> ).
D.Gy	dose / Gy for each particle type (array of size n).
material.no	material index (see also <a href="#">material.no</a> ).
stopping.power.source.no	TODO (see also <a href="#">stopping.power.source.no</a> ).

**Value**

fluence.cm2 to be allocated by the user which will be used to return the results (array of size n)

**See Also**View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L380](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L380)



---

 AT.fluence.weighted.LET.MeV.cm2.g

*AT.fluence.weighted.LET.MeV.cm2.g*


---

## Description

Computes the fluence-weighted average LET of a particle field

## Usage

```
AT.fluence.weighted.LET.MeV.cm2.g(E.MeV.u, particle.no, fluence.cm2,
  material.no, stopping.power.source.no)
```

## Arguments

E.MeV.u	energy of particles in the mixed particle field (array of size <code>number.of.field.components</code> ) (see also <code>E.MeV.u</code> ).
particle.no	particle index (array of size <code>number.of.field.components</code> ) (see also <code>particle.no</code> ).
fluence.cm2	fluences of particles in the mixed particle field (array of size <code>number.of.field.components</code> ).
material.no	material index (see also <code>material.no</code> ).
stopping.power.source.no	TODO (see also <code>stopping.power.source.no</code> ).

## Value

```
fluence-weighted
  fluence-weighted
```

## See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L638](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L638)

## Examples

```
# fluence- and dose-weighted LET for a simple mixed field
# of high and low (99/1) energy protons
AT.fluence.weighted.LET.MeV.cm2.g( E.MeV.u      = c(100, 5),
  particle.no  = c(1001, 1001),
  fluence.cm2  = c(99e8, 1e8),
  material.no  = 1,                               # water
  stopping.power.source.no = 2)
AT.dose.weighted.LET.MeV.cm2.g( E.MeV.u      = c(100, 5),
  particle.no  = c(1001, 1001),
  fluence.cm2  = c(99e8, 1e8),
  material.no  = 1,                               # water
  stopping.power.source.no = 2)
```



---

```
AT.FLUKA.particle.name.to.libamtrack.particle.name
    AT.FLUKA.particle.name.to.libamtrack.particle.name
```

---

**Description**

Converts FLUKA style particle names to libamtrack convention.

**Usage**

```
AT.FLUKA.particle.name.to.libamtrack.particle.name(
    FLUKA.particle.names)
```

**Arguments**

```
FLUKA.particle.names
    vector of strings containing the FLUKA-style particle names
```

**Details**

FLUKA particle names relating to all particle with a specific Z, e.g. LI\* or H\*, will be translated to the 'even' isotope  $A = 2*Z$ . The only exception is H\* which will be replaced by 1H. For example, LI\* → 6Li, C\* → 12C, BE\* → 8Be. This is a pure convention and does not necessarily represent the most prominent isotope. For most computations which only rely on Z it is, however, valid.

**Value**

Character string vector with libamtrack-style particle names.

**Examples**

```
AT.FLUKA.particle.name.to.libamtrack.particle.name( FLUKA.particle.names =
    c("H*", "B*", "B10", "C12", "BE7", "U238") )
```

---

```
AT.FLUKA.read.USRBIN.mesh
    AT.FLUKA.read.USRBIN.mesh
```

---

**Description**

Reads USRBIN output for Cartesian mesh, also for multiple output files from cluster runs. As USRBIN (mesh) scores energy deposited per unit volume per primary weight (GeV/cm<sup>3</sup>), the density as to be given for dose computation.

**Usage**

```
AT.FLUKA.read.USRBIN.mesh(exp.name, number.of.runs, unit, data.source =
    'local', density.g.cm3 = 1.0)
```

**Arguments**

exp.name	Experiment name, i.e. name of input file (without '.inp' extension)
number.of.runs	Number of output files from parallel (cluster) runs.
unit	FLUKA output unit number
data.source	'local' if output files are from a local machine, 'condor' if from condor cluster, 'condor_cleaned' if from condor cluster with clean option (-c) in rcfluka.py
density.g.cm3	Physical density of material in mesh, needed for computation of dose.

**Value**

Data frame with deposited energy, deposited dose (and in case of multiple runs estimate of their standard deviation) for each mesh cell.

**Examples**

```
# None yet, requires FLUKA output file
```

---

```
AT.FLUKA.read.USRBIN.regs
    AT.FLUKA.read.USRBIN.regs
```

---

**Description**

Reads USRBIN output for a series of regions, also for multiple output files from cluster runs. As USRBIN (regions, option 12.0) scores energy deposited per primary weight (GeV), both the region volume(s) and density(ies) have to be given for dose computation.

**Usage**

```
AT.FLUKA.read.USRBIN.regs(exp.name, number.of.runs, unit, data.source =
    'local', vol.cm3 = NULL, density.g.cm3 = NULL)
```

**Arguments**

exp.name	Experiment name, i.e. name of input file (without '.inp' extension)
number.of.runs	Number of output files from parallel (cluster) runs.
unit	FLUKA output unit number
data.source	'local' if output files are from a local machine, 'condor' if from condor cluster, 'condor_cleaned' if from condor cluster with clean option (-c) in rcfluka.py

vol.cm3	Volume of regions, either single value (will be applied to all regions) or vector of length matching the number of regions (individual volume for each region).
density.g.cm3	Physical density of regions, either single value (will be applied to all regions) or vector of length matching the number of regions (individual density for each region).

**Value**

Data frame with deposited energy, deposited dose (and in case of multiple runs estimate of their standard deviation) for each region.

**Examples**

```
# None yet, requires FLUKA output file
```

---

```
AT.FLUKA.read.USRTRACK
      AT.FLUKA.read.USRTRACK
```

---

**Description**

Reads USRTRACK output (for energy spectra scoring using flusw.SPC.f) for a series of regions, also for multiple output files from cluster runs (using rcfuka.py). USRTRACK scores the fluence per primary weight per bin width(cm-2/GeV) but only if the correct volume of the corresponding regions is given in the FLUKA input card! AT.FLUKA.read.USRTRACK will however out the absolute fluence for each energy bin (in cm-2) as this can be directly used by libamtrack.

**Usage**

```
AT.FLUKA.read.USRTRACK(exp.name, number.of.runs, unit, data.source =
  'local', compress = TRUE)
```

**Arguments**

exp.name	Experiment name, i.e. name of input file (without '.inp' extension)
number.of.runs	Number of output files from parallel (cluster) runs.
unit	FLUKA output unit number
data.source	'local' if output files are from a local machine, 'condor' if from condor cluster, 'condor_cleaned' if from condor cluster with clean option (-c) in rcfuka.py
compress	If TRUE, all entries with zero fluence will be removed from resulting data frame in order to save memory.

**Value**

Data frame with midpoints and widths of energy bins, particle index no `particle.no`, and fluence (absolute [cm-2], i.e. not normalized to bin width!) for each region.

## Examples

```
# None yet, requires FLUKA output file
```

---

AT.gamma.from.E	<i>AT.gamma.from.E</i>
-----------------	------------------------

---

## Description

Returns relativistic gamma

## Usage

```
AT.gamma.from.E(E.MeV.u)
```

## Arguments

E.MeV.u	vector of energies of particle per nucleon [MeV] (array of size n) (see also <a href="#">E.MeV.u</a> ).
---------	---

## Value

gamma	vector of results (array of size n)
status	status

## See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L52](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L52)

## Examples

```
# Compute mass in MeV/c2 for a proton with 1-1000 MeV kinetic energy:  
938.3 * AT.gamma.from.E( E.MeV.u = 10^(0:3))$gamma
```

---

AT.gamma.response      *AT.gamma.response*

---

### Description

Returns a system (detector or cells) response for given doses according to the chosen gamma response model

### Usage

```
AT.gamma.response(d.Gy, gamma.model, gamma.parameter, lethal.event.mode)
```

### Arguments

`d.Gy`                    doses in Gy (array of size `number.of.doses`).

`gamma.model`            gamma response model index.

`gamma.parameter`        vector holding necessary parameters for the chose gamma response model (array of size 9).

`lethal.event.mode`      if true computation is done in lethal event mode.

### Value

`S`                        gamma responses (array of size `number.of.doses`)

### Examples

```
# Show the gamma response of two Al203 detectors (A & B) and two protocols
# ('peak' and 'total')

# parametrized in two components (single hit/single target and two hit/single
# target)
# as measured and published by Edmund et al., NIM B 262 (2007), 261-275
require(lattice)
# Compute 100 points between 0.1 and 25 Gy
# General hit/target model
d.Gy                      <- 10^seq(from = log10(0.1), to = log10(25),
  length.out = 100)
gamma.model <- 2
# Probe A, 'peak'
R <- 1
Smax <- 0.81e6
k1 <- Smax * (R / 100)
k2 <- Smax * (1 - R / 100)
gamma.parameter.peak.A <- c( k1 = k1, D01 = 0.36, c1 = 1, m1 = 1,
  k2 = k2, D02 = 3.06, c2 = 2, m2 = 1,
```

```

    0)
# Probe A, 'total'
R <- 33
Smax <- 6.2e6
k1 <- Smax * (R / 100)
k2 <- Smax * (1 - R / 100)
gamma.parameter.total.A <- c( k1 = k1, D01 = 1.13, c1 = 1, m1 = 1,
                             k2 = k2, D02 = 1.77, c2 = 2, m2 = 1,
                             0)
# Probe B, 'peak'
R <- 13
Smax <- 2.84e6
k1 <- Smax * (R / 100)
k2 <- Smax * (1 - R / 100)
gamma.parameter.peak.B <- c( k1 = k1, D01 = 4.15, c1 = 1, m1 = 1,
                             k2 = k2, D02 = 5.14, c2 = 2, m2 = 1,
                             0)
# Probe B, 'total'
R <- 44
Smax <- 27.6e6
k1 <- Smax * (R / 100)
k2 <- Smax * (1 - R / 100)
gamma.parameter.total.B <- c( k1 = k1, D01 = 2.90, c1 = 1, m1 = 1,
                             k2 = k2, D02 = 4.66, c2 = 2, m2 = 1,
                             0)
vecA <- AT.gamma.response( d.Gy          = d.Gy,
                           gamma.model  = gamma.model,
                           gamma.parameter = gamma.parameter.peak.A,
                           lethal.event.mode = FALSE)$response
vecB <- AT.gamma.response( d.Gy          = d.Gy,
                           gamma.model  = gamma.model,
                           gamma.parameter = gamma.parameter.total.A,
                           lethal.event.mode = FALSE)$response
vecC <- AT.gamma.response( d.Gy          = d.Gy,
                           gamma.model  = gamma.model,
                           gamma.parameter = gamma.parameter.peak.B,
                           lethal.event.mode = FALSE)$response
vecD <- AT.gamma.response( d.Gy          = d.Gy,
                           gamma.model  = gamma.model,
                           gamma.parameter = gamma.parameter.total.B,
                           lethal.event.mode = FALSE)$response
# Compose data frame
df <- data.frame( d.Gy      = rep( d.Gy, 4),
                  S        = c(vecA, vecB, vecC, vecD ),
                  which    = rep( c( rep("peak", length(d.Gy)),
                                     rep("total", length(d.Gy))), 2),
                  probe    = c( rep("probe A", 2 * length(d.Gy)),
                                rep("probe B", 2 * length(d.Gy))) )
# Plot
xyplot( log10(S) ~ log10(d.Gy)|probe,
        df,
        groups = which,
        type = 'l',

```

```

lwd = 2,
ylim = log10(c(1e3, 4e7)),
ylab = list( "OSL response", cex = 1.2),
xlim = log10(c(0.1, 25)),
xlab = list( "dose / Gy", cex = 1.2),
scales = list( x = list( at      = log10(c(1,10,20)),
                      labels = as.character(c(1,10,20))),
               y = list( at      = c(4,5,6,7),
                      labels = 10^(c(4,5,6,7))))),
aspect = 2.5)

```

---

AT.Gauss.IDF

*AT.Gauss.IDF*


---

### Description

Compute Gauss inverse distribution function (for compatibility)

### Usage

AT.Gauss.IDF(rnd)

### Arguments

rnd                    random number from uniform distribution between 0 and 1 (array of size n).

### Value

lambda.gauss        resulting Gauss lambda (array of size n)

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L584](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L584)

---

AT.Gauss.PDF

*AT.Gauss.PDF*


---

### Description

Computes Gauss probability density function (for compatibility)

### Usage

AT.Gauss.PDF(lambda.gauss)

**Arguments**

lambda.gauss      Gauss lambda (array of size n).

**Value**

density              resulting density (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L578](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L578)

AT.get.materials.data    *AT.get.materials.data*

**Description**

Returns material data for list of materials

**Usage**

AT.get.materials.data(material.no)

**Arguments**

material.no        material indices (array of size number.of.materials) (see also [material.no](#)).

**Value**

density.g.cm3	material density in g/cm3 (array of size number.of.materials)
I.eV	mean ionization potential in eV (array of size number.of.materials)
alpha.g.cm2.MeV	fit parameter for power-law representation of stp.power/range/E-dependence (array of size number.of.materials)
p.MeV	fit parameter for power-law representation of stp.power/range/E-dependence (array of size number.of.materials)
m.g.cm2	fit parameter for the linear representation of fluence changes due to nuclear interactions based on data from Janni 1982 (array of size number.of.materials)
average.A	average mass number (array of size number.of.materials)
average.Z	average atomic number (array of size number.of.materials)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L238](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L238)



---

AT.I.eV.from.composition  
*AT.I.eV.from.composition*

---

**Description**

Computes the I value for a given material composition

**Usage**

AT.I.eV.from.composition(Z, A, weight.fraction)

**Arguments**

Z                    atomic numbers of constituents (array of size n).  
A                    mass numbers of constituents (array of size n).  
weight.fraction     relative fractions of weight of constituents (array of size n).

**Value**

I.eV                I value in eV

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L452](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L452)

---

AT.inverse.gamma.response  
*AT.inverse.gamma.response*

---

**Description**

Computes the inverse gamma response, i.e. the dose for a given response. !!THIS WORKS UPTONOW FOR THE LIN-QUAD-MODEL ONLY, BUT SHOULD BE DEVELOPED TO A GENERAL ROUTINE!!

**Usage**

AT.inverse.gamma.response( surv, alpha, beta)

**Arguments**

surv	Response (survival)
alpha	Alpha parameter in (1/Gy)
beta	Beta parameter in (1/Gy <sup>2</sup> )

**Value**

Dose in Gy

**Examples**

```
# Compute dose for 10% survival for HSG cells
AT.inverse.gamma.response( surv = 0.1, alpha = 0.2, beta = 0.05)
```

---

AT.kappa	<i>AT.kappa</i>
----------	-----------------

---

**Description**

Computes the kappa criterium for the energy loss distribution according to Seltzer and Berger, and CERN W5013 No effective projectile charge is considered!

**Usage**

```
AT.kappa(E.MeV.u, particle.no, material.no, slab.thickness.um)
```

**Arguments**

E.MeV.u	energy of particle per amu (array of size n) (see also <a href="#">E.MeV.u</a> ).
particle.no	particle index (array of size n) (see also <a href="#">particle.no</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
slab.thickness.um	slab thickness in um (array of size n).

**Value**

kappa            kappa parameter (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L65](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L65)

---

 AT.lambda.landau.from.energy.loss

*AT.lambda.landau.from.energy.loss*


---

### Description

Computes the lambda parameter for the Landau distribution acc. to CERN W5013 No effective projectile charge is considered!

### Usage

```
AT.lambda.landau.from.energy.loss(energy.loss.keV, E.MeV.u,
  particle.no, material.no, slab.thickness.um)
```

### Arguments

energy.loss.keV	energy loss (array of size n).
E.MeV.u	energy of particle per nucleon (see also <a href="#">E.MeV.u</a> ).
particle.no	particle index (see also <a href="#">particle.no</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
slab.thickness.um	slab thickness in um.

### Value

lambda.landau (array of size n)

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L149](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L149)

---

 AT.lambda.max

*AT.lambda.max*


---

### Description

Computes the mean lambda, introduced to enable average value for Landau distribution. See Geant3 W5013, p.254

### Usage

```
AT.lambda.max(E.MeV.u, particle.no, material.no, slab.thickness.um)
```

**Arguments**

E.MeV.u            energy of particle per amu (array of size n) (see also [E.MeV.u](#)).

particle.no        particle index (array of size n) (see also [particle.no](#)).

material.no        material index (see also [material.no](#)).

slab.thickness.um            slab thickness in um (array of size n).

**Value**

lambda.max        maximum lambda for given particle (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L115](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L115)

---

AT.lambda.mean	<i>AT.lambda.mean</i>
----------------	-----------------------

---

**Description**

Computes the mean lambda, introduced to enable average value for Landau distribution. See Geant3 W5013, p.254

**Usage**

AT.lambda.mean(E.MeV.u, particle.no, material.no, slab.thickness.um)

**Arguments**

E.MeV.u            energy of particle per amu (array of size n) (see also [E.MeV.u](#)).

particle.no        particle index (array of size n) (see also [particle.no](#)).

material.no        material index (see also [material.no](#)).

slab.thickness.um            slab thickness in um (array of size n).

**Value**

lambda.mean        mean lambda for given particle (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L93](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L93)

---

`AT.lambda.vavilov.from.energy.loss`*AT.lambda.vavilov.from.energy.loss*

---

**Description**

Computes the lambda parameter for the Vavilov distribution acc. to CERN W5013 No effective projectile charge is considered!

**Usage**

```
AT.lambda.vavilov.from.energy.loss(energy.loss.keV, E.MeV.u,  
particle.no, material.no, slab.thickness.um)
```

**Arguments**

<code>energy.loss.keV</code>	energy loss (array of size n).
<code>E.MeV.u</code>	energy of particle per nucleon (see also <a href="#">E.MeV.u</a> ).
<code>particle.no</code>	particle index (see also <a href="#">particle.no</a> ).
<code>material.no</code>	material index (see also <a href="#">material.no</a> ).
<code>slab.thickness.um</code>	slab thickness in um.

**Value**

`lambda.vavilov` (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L304](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L304)

---

`AT.Landau.IDF`*AT.Landau.IDF*

---

**Description**

Computes the Landau inverse distribution function using CERNLIB (G115)

**Usage**

```
AT.Landau.IDF(rnd)
```

**Arguments**

rnd                    random number from uniform distribution between 0 and 1 (array of size n).

**Value**

lambda.landau    resulting Landau lambda (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L85](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L85)

---

AT.Landau.PDF

*AT.Landau.PDF*

---

**Description**

Computes the Landau probability density function using CERNLIB (G115)

**Usage**

AT.Landau.PDF(lambda.landau)

**Arguments**

lambda.landau    Landau lambda (array of size n).

**Value**

density            resulting density (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L77](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L77)



```

# Use stopping power data from PSTAR
AT.Mass.Stopping.Power( E.MeV.u          = 270.55,
                        particle.no       = 6012,
                        material.no       =
AT.material.no.from.material.name("Water, Liquid"),
                        stopping.power.source = "PSTAR")
# Use stopping power data from ICRU
AT.Mass.Stopping.Power( E.MeV.u          = 270.55,
                        particle.no       = 6012,
                        material.no       =
AT.material.no.from.material.name("Water, Liquid"),
                        stopping.power.source = "ICRU")
# Use stopping power data from file shipped with libamtrack
# For water:
AT.Mass.Stopping.Power( E.MeV.u          = 270.55,
                        particle.no       = 6012,
                        material.no       =
AT.material.no.from.material.name("Water, Liquid"),
                        stopping.power.source = system.file("extdata",
"FLUKA_DEDX_WATER_76.8eV.txt", package = "libamtrack"))

# And alumina (CAVE: the data file contains stopping power in keV/um, which is
# return, despite the name of the function):
AT.Mass.Stopping.Power( E.MeV.u          = 270.55,
                        particle.no       = 6012,
                        material.no       =
AT.material.no.from.material.name("Aluminum Oxide"),
                        stopping.power.source = system.file("extdata",
"FLUKA_DEDX_ALOX.txt", package = "libamtrack"))

# When using density scaling, the value becomes clearly wrong! Thus, do
# preferably use mass stopping power data in the files
AT.Stopping.Power( E.MeV.u          = 270.55,
                  particle.no       = 6012,
                  material.no       =
AT.material.no.from.material.name("Aluminum Oxide"),
                  stopping.power.source = system.file("extdata",
"FLUKA_DEDX_ALOX.txt", package = "libamtrack"))

```

---

AT.Mass.Stopping.Power.with.no

*AT.Mass.Stopping.Power.with.no*

---

## Description

Retrieves the electronic mass stopping power in MeV\*cm<sup>2</sup>/g for the requested energies and particles for a specified material and data source. The data source is thereby given via its integer id (s. AT\_StoppingPowerData.h for details). Data that should be read from a file cannot be used with this method.



**Usage**

```
AT.Mass.Stopping.Power.with.no(stopping.power.source.no, E.MeV.u,
particle.no, material.no)
```

**Arguments**

```
stopping.power.source.no    id of the data source (see also stopping.power.source.no).
E.MeV.u                    kinetic energies in MeV per amu (array of size n) (see also E.MeV.u).
particle.no                particle numbers (array of size n) (see also particle.no).
material.no                material number (see also material.no).
```

**Value**

```
stopping.power.MeV.cm2.g    array to return stopping powers (array of size n)
status                      status
```

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_StoppingPower.c#L47](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_StoppingPower.c#L47)

---

```
AT.material.name.from.material.no
    AT.material.name.from.material.no
```

---

**Description**

Returns material name(s) for given material index number(s)

**Usage**

```
AT.material.name.from.material.no(material.no)
```

**Arguments**

```
material.no    material index number(s) (see also material.no).
```

**Value**

```
material.name    material name(s) (see also material.name).
```

**Examples**

```
# Get material names for all predefined materials
AT.material.name.from.material.no( material.no = 1:25)
```

---

```
AT.material.no.from.material.name
    AT.material.no.from.material.name
```

---

**Description**

Returns material name(s) for given material index number(s)

**Usage**

```
AT.material.no.from.material.name(material.name)
```

**Arguments**

material.name material name(s) (see also [material.name](#)).

**Value**

material.no material index number(s) (see also [material.no](#)).

**Examples**

```
# Get material index numbers for three materials of which one is not
# implemented
AT.material.no.from.material.name( material.name = c("Water, Liquid",
  "PMMA", "Squirrel liver pudding"))
```

---

```
AT.max.E.transfer.MeV AT.max.E.transfer.MeV
```

---

**Description**

Kinetic energy maximally transferred from an ion to an electron in a collision - relativistic or non-relativistic

**Usage**

```
AT.max.E.transfer.MeV(E.MeV.u)
```

**Arguments**

E.MeV.u energies of particle per nucleon [MeV/u]; if positive, the computation will be relativistic; if negative, the classic formular will be used (array of size n) (see also [E.MeV.u](#)).

**Value**

max.E.transfer.MeV  
 maximal energies transferred (array of size n)

status  
 status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L209](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L209)

**Examples**

```
# Plot maximum energy transferred in a collision in the range from 1 to 1000
# MeV
# (this is independent of the ion type!) and compare the classical with
# the relativistic approach
E.MeV.u <- 10^seq(0, 3, length.out = 50)
df <- data.frame( E.MeV = E.MeV.u,
                 max.E.keV.classical = AT.max.E.transfer.MeV(-1.0
                 * E.MeV.u)$max.E.transfer.MeV * 1000,
                 max.E.keV.relativistic =
                 AT.max.E.transfer.MeV(E.MeV.u)$max.E.transfer.MeV * 1000)
```

---

AT.max.E.transfer.MeV.new

*AT.max.E.transfer.MeV.new*

---

**Description**

Kinetic energy maximally transferred from an ion to an electron in a collision - relativistic or non-relativistic

**Usage**

AT.max.E.transfer.MeV.new(E.MeV.u, A)

**Arguments**

E.MeV.u  
 energies of particle per nucleon [MeV/u]; if positive, the computation will be relativistic; if negative, the classic formular will be used (array of size n) (see also [E.MeV.u](#)).

A  
 atomic mass (array of size n).

**Value**

max.E.transfer.MeV  
 maximal energies transferred (array of size n)

status  
 status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L209](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L209)

---

AT.max.electron.ranges.m

*AT.max.electron.ranges.m*

---

**Description**

Returns the maximum electron range (track radius) in m for a given parametrization

**Usage**

AT.max.electron.ranges.m(E.MeV.u, material.no, er.model)

**Arguments**

E.MeV.u	kinetic energy for particles in the given field (array of size number.of.particles) (see also <a href="#">E.MeV.u</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
er.model	electron-range model index (see also <a href="#">er.model</a> ).

**Value**

max.electron.range.m  
electron range (track radius) in m (array of size number.of.particles)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_ElectronRange.c#L135](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_ElectronRange.c#L135)

**Examples**

```
# Compute the electron range in PMMA for the Tabata parametrization between
# 0.3 keV and 30 MeV
AT.max.electron.ranges.m( E.MeV.u    = 0.3 * 10^(-4:2),
                          material.no = 4,
                          er.model    = 7)
```

---

```
AT.mean.energy.loss.keV
    AT.mean.energy.loss.keV
```

---

**Description**

Computes the mean energy loss in a slab of material using the Bethe formula for many particles according to ICRU49 BUT WITHOUT shell, Bloch or Barkas correction! No effective projectile charge is considered!

**Usage**

```
AT.mean.energy.loss.keV(E.MeV.u, particle.no, material.no,
    slab.thickness.um)
```

**Arguments**

E.MeV.u	energies of particle per nucleon (see also <a href="#">E.MeV.u</a> ).
particle.no	particle indices (see also <a href="#">particle.no</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
slab.thickness.um	slab thickness in um.

**Value**

result	result
--------	--------

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L42](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L42)

---

```
AT.mean.number.of.tracks.contrib
    AT.mean.number.of.tracks.contrib
```

---

**Description**

Computes the number of track contributing to a representative point in a mixed field

**Usage**

```
AT.mean.number.of.tracks.contrib(E.MeV.u, particle.no, fluence.cm2,
    material.no, er.model,
    stopping.power.source.no)
```

**Arguments**

<code>E.MeV.u</code>	energy of particles in the mixed particle field (array of size <code>number.of.field.components</code> ) (see also <code>E.MeV.u</code> ).
<code>particle.no</code>	particle index (array of size <code>number.of.field.components</code> ) (see also <code>particle.no</code> ).
<code>fluence.cm2</code>	fluences of particles in the mixed particle field (array of size <code>number.of.field.components</code> ).
<code>material.no</code>	material index (see also <code>material.no</code> ).
<code>er.model</code>	chosen electron-range-model (see also <code>er.model</code> ).
<code>stopping.power.source.no</code>	TODO (see also <code>stopping.power.source.no</code> ).

**Value**

`resulting`      `resulting`

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L764](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L764)

---

`AT.momentum.MeV.c.u.from.E.MeV.u`

*AT.momentum.MeV.c.u.from.E.MeV.u*

---

**Description**

Returns relativistic momenta per nucleon for particles with given kinetic energy

**Usage**

`AT.momentum.MeV.c.u.from.E.MeV.u(E.MeV.u)`

**Arguments**

`E.MeV.u`      kinetic energy per nucleon (array of size `n`) (see also `E.MeV.u`).

**Value**

`momentum.MeV.c`   momentum per nucleon (array of size `n`)  
`return`              `return`

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L289](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L289)

**Examples**

```
# Relation between kinetic proton energy and momentum
# in relativistic and non-relativistic regime
E.MeV.u    <- 10^seq(-2, 5, length.out = 100)
df         <- data.frame( E.MeV      = E.MeV.u,
                          p.MeV.c    =
AT.momentum.MeV.c.u.from.E.MeV.u(E.MeV.u)$momentum.MeV.c)
```

---

```
AT.nuclear.spin.from.particle.no
      AT.nuclear.spin.from.particle.no
```

---

**Description**

Returns nuclear spin from particle no

**Usage**

```
AT.nuclear.spin.from.particle.no(particle.no)
```

**Arguments**

particle.no      particle index number (array of size n) (see also [particle.no](#)).

**Value**

I	nuclear spin (array of size n)
status	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataParticle.c#L184](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataParticle.c#L184)

---

```
AT.particle.name.from.particle.no
      AT.particle.name.from.particle.no
```

---

**Description**

Returns particle name(s) for given particle index number(s)

**Usage**

```
AT.particle.name.from.particle.no(particle.no)
```

**Arguments**

particle.no     particle name(s) (see also [particle.no](#)).

**Value**

particle.name     particle index number(s) (see also [particle.name](#)).

**Examples**

```
# Get particle index names for some nuclides
AT.particle.name.from.particle.no( particle.no = c(1001, 6012))
```

---

```
AT.particle.no.from.particle.name
      AT.particle.no.from.particle.name
```

---

**Description**

Returns particle index number(s) for given particle names(s)

**Usage**

```
AT.particle.no.from.particle.name(particle.name)
```

**Arguments**

particle.name     particle name(s) (see also [particle.name](#)).

**Value**

particle.no     particle index number(s) (see also [particle.no](#)).

**Examples**

```
# Get particle names for some nuclides
AT.particle.no.from.particle.name( particle.name = c("1H", "2H", "3He",
"12C", "16O", "238U"))
```



---

AT.particle.no.from.Z.and.A  
*AT.particle.no.from.Z.and.A*

---

**Description**

Returns particle index number from given A and Z

**Usage**

AT.particle.no.from.Z.and.A(Z, A)

**Arguments**

Z                    atomic numbers (array of size n).  
A                    mass number (array of size n).

**Value**

particle.no        corresponding particle index numbers (array of size n)  
status             status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataParticle.c#L34](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataParticle.c#L34)

---

AT.r.RDD.m                    *AT.r.RDD.m*

---

**Description**

Returns distance as a function of dose

**Usage**

AT.r.RDD.m(D.RDD.Gy, E.MeV.u, particle.no, material.no, rdd.model,  
rdd.parameter, er.model, stopping.power.source.no)

**Arguments**

D.RDD.Gy	dose [Gy] (array of size n).
E.MeV.u	particle (ion) energy per nucleon [MeV/u] (see also <a href="#">E.MeV.u</a> ).
particle.no	particle code number (see also <a href="#">particle.no</a> ).
material.no	material code number (see also <a href="#">material.no</a> ).
rdd.model	Radial Dose Distribution model code number (see also <a href="#">rdd.model</a> ).
rdd.parameter	Radial Dose Distribution model parameters vector (array of size 4).
er.model	delta electron range model code number (see also <a href="#">er.model</a> ).
stopping.power.source.no	TODO (see also <a href="#">stopping.power.source.no</a> ).

**Value**

r.RDD.m	distance [m] (array of size n)
status	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_RDD.c#L662](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_RDD.c#L662)

---

AT.run.CPPSC.method    *AT.run.CPPSC.method*

---

**Description**

Computes HCP response and relative efficiency/RBE using compound Poisson processes and successive convolutions (CPP\_SC, the SPIFF algorithm)

**Usage**

```
AT.run.CPPSC.method(E.MeV.u, particle.no, fluence.cm2.or.dose.Gy,
  material.no, stopping.power.source.no,
  rdd.model, rdd.parameters, er.model, gamma.model, gamma.parameters,
  N2, fluence.factor, write.output, shrink.tails, shrink.tails.under,
  adjust.N2, lethal.events.mode)
```

**Arguments**

E.MeV.u	particle energy for each component in the mixed particle field [MeV/u] (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">E.MeV.u</a> ).
particle.no	particle type for each component in the mixed particle field (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">particle.no</a> ).

fluence.cm2.or.dose.Gy	if positive, particle fluence for each component in the mixed particle field [1/cm <sup>2</sup> ]; if negative, particle dose for each component in the mixed particle field [Gy] (array of size <code>number.of.field.components</code> ) (see also <code>fluence.cm2.or.dose.Gy</code> ).
material.no	index number for detector material (see also <code>material.no</code> ).
stopping.power.source.no	TODO (see also <code>stopping.power.source.no</code> ).
rdd.model	index number for chosen radial dose distribution (see also <code>rdd.model</code> ).
rdd.parameters	parameters for chosen radial dose distribution (array of size 4).
er.model	index number for chosen electron-range model (see also <code>er.model</code> ).
gamma.model	index number for chosen gamma response.
gamma.parameters	parameters for chosen gamma response (array of size 9).
N2	number of bins per factor of two for the dose scale of local dose histogram.
fluence.factor	factor to scale the fluences / doses given in <code>fluence.cm2.or.dose.Gy</code> with.
write.output	if true, a log-file is written to SuccessiveConvolutions.txt in the working directory.
shrink.tails	if true, tails of the local dose distribution, contributing less than <code>shrink.tails.under</code> are cut.
shrink.tails.under	limit for tail cutting in local dose distribution.
adjust.N2	if true, N2 will be increase if necessary at high fluence to ensure sufficient local dose histogram resolution.
lethal.events.mode	if true, computations are done for dependent subtargets.

**Value**

N2	number of bins per factor of two for the dose scale of local dose histogram
relative.ency	particle response at dose D / gamma response at dose D
d.check	sanity check: total dose (in Gy) as returned by the algorithm
S.HCP	absolute particle response
S.gamma	absolute gamma response
mean.number.of.tracks.contrib	mean number of tracks contributing to representative point
start.number.of.tracks.contrib	low fluence approximation for mean number of tracks contributing to representative point (start value for successive convolutions)
n.convolution	number of convolutions performed to reach requested dose/fluence
lower.Jensen.bound	lower bound for Jensen's inequity
upper.Jensen.bound	upper bound for Jensen's inequity

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_Algorithms\\_CPP.c#L34](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_Algorithms_CPP.c#L34)

**Examples**

```
# Compute the relative efficiency of an Alanine detector in a mixed
# carbon / proton field
AT.run.CPPSC.method( particle.no          = c(6012, 1001,
  1001),          # namely carbon, protons, and protons with
                  E.MeV.u                = c(270, 270, 5),

# 270 MeV/u (primary Carbon, 270 MeV/u and 5 MeV/u (fast and slow proton
# component)
                fluence.cm2.or.dose.Gy    = c(1e8, 1e9, 1e7),
                # and their corresponding fluences
                material.no                = 5,
                # i.e. Alanine
                rdd.model                  = 3,
                # simple 'Geiss' parametrization of radial dose distribution
                rdd.parameter              = 50e-9,
                # with 50 nm core radius
                er.model                    = 4,
                # M. Scholz' parametrization of track radius
                gamma.model                 = 2,
                # General hit/target X ray response, but
                gamma.parameters           = c(1,500,1,1,0),

# as simple single exponential saturation (one hit, one target), saturation
# dose 500 Gy
                N2                          = 10,
                # ten bins per factor 2 for internal local dose histogramming
                fluence.factor              = 1.0,
                # can be used to easily scale total fluence (historical)
                write.output                = TRUE,
                # write a log file
                shrink.tails                = TRUE,
                # cut tails of local dose distribution, if...
                shrink.tails.under          = 1e-30,
                # ... they contribute less then 1e-30 to first moment of histogram
                adjust.N2                   = TRUE,
                # perform rebinning if local dose distribution becomes too narrow
                lethal.events.mode          = FALSE,
                # use independent subtargets
                stopping.power.source.no    = 2)
```

**Description**

Computes HCP response and relative efficiency/RBE using summation of tracks on a Cartesian grid (the GSM algorithm). Be aware that this routine can take considerable time to compute depending on the arguments, esp. for higher energy (>10 MeV/u) particles. It is therefore advantageous to test your settings with a low number of runs first.

**Usage**

```
AT.run.GSM.method(E.MeV.u, particle.no, fluence.cm2.or.dose.Gy,
  material.no, stopping.power.source.no,
  rdd.model, rdd.parameters, er.model, gamma.model, gamma.parameters,
  N.runs, write.output, nX, voxel.size.m, lethal.events.mode)
```

**Arguments**

E.MeV.u	particle energy for each component in the mixed particle field [MeV/u] (array of size <code>number.of.field.components</code> ) (see also <code>E.MeV.u</code> ).
particle.no	particle type for each component in the mixed particle field (array of size <code>number.of.field.components</code> ) (see also <code>particle.no</code> ).
fluence.cm2.or.dose.Gy	if positive, particle fluence for each component in the mixed particle field [1/cm <sup>2</sup> ]; if negative, particle dose for each component in the mixed particle field [Gy] (array of size <code>number.of.field.components</code> ) (see also <code>fluence.cm2.or.dose.Gy</code> ).
material.no	index number for detector material (see also <code>material.no</code> ).
stopping.power.source.no	TODO (see also <code>stopping.power.source.no</code> ).
rdd.model	index number for chosen radial dose distribution (see also <code>rdd.model</code> ).
rdd.parameters	parameters for chosen radial dose distribution (array of size 4).
er.model	index number for chosen electron-range model (see also <code>er.model</code> ).
gamma.model	index number for chosen gamma response.
gamma.parameters	parameters for chosen gamma response (array of size 9).
N.runs	number of runs within which track positions will be resampled.
write.output	if true, a protocol is written to SuccessiveConvolutions.txt in the working directory.
nX	number of voxels of the grid in x (and y as the grid is quadratic).
voxel.size.m	side length of a voxel in m.
lethal.events.mode	if true, allows to do calculations for cell survival.

**Value**

relative.ency	particle response at dose D / gamma response at dose D
d.check	sanity check: total dose (in Gy) as returned by the algorithm

S.HCP	absolute particle response
S.gamma	absolute gamma response
n.particles	average number of particle tracks on the detector grid
sd.relative.efficiency	standard deviation for relative.efficiency
sd.d.check	standard deviation for d.check
sd.S.HCP	standard deviation for S.HCP
sd.S.gamma	standard deviation for S.gamma
sd.n.particles	standard deviation for n.particles

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_Algorithms\\_GSM.c#L277](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_Algorithms_GSM.c#L277)

### Examples

```
# Compute the relative efficiency of an Alanine detector
# in a proton field
AT.run.GSM.method( # protons
    particle.no           = 1001,
    # with 10 MeV/u
    E.MeV.u              = 10,
    # delivering 1 Gy
    fluence.cm2.or.dose.Gy = c(-1.0),
    # i.e. Alanine
    material.no          = 5,

# simple 'Geiss' parametrization of radial dose distribution
    rdd.model            = 3,
    # with 50 nm core radius
    rdd.parameter        = 50e-9,
    # M. Scholz' parametrization of track radius
    er.model             = 4,
    # Use exponential saturation
    gamma.model          = 4,
    # max. response normalized to 1, saturation dose 500 Gy
    gamma.parameters     = c(1,500),
    # resample 1000 times
    N.runs               = 1000,
    # write a log file
    write.output         = TRUE,
    # use a 10x10 grid
    nX                   = 10,
    # with 5 nm voxel size
    voxel.size.m         = 5e-9,
    # use independent subtargets
    lethal.events.mode   = FALSE,
    # and PSTAR stopping powers
    stopping.power.source.no = 2)
```

---

AT.run.IGK.method      *AT.run.IGK.method*

---

## Description

Computes HCP response and relative efficiency/RBE using Katz' Ion-Gamma-Kill approach according to Waligorski, 1988

## Usage

```
AT.run.IGK.method(E.MeV.u, particle.no, fluence.cm2.or.dose.Gy,
  material.no, stopping.power.source.no,
  rdd.model, rdd.parameters, er.model, gamma.model, gamma.parameters,
  saturation.cross.section.factor, write.output)
```

## Arguments

E.MeV.u	particle energy for each component in the mixed particle field [MeV/u] (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">E.MeV.u</a> ).
particle.no	particle type for each component in the mixed particle field (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">particle.no</a> ).
fluence.cm2.or.dose.Gy	if positive, particle fluence for each component in the mixed particle field [1/cm <sup>2</sup> ]; if negative, particle dose for each component in the mixed particle field [Gy] (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">fluence.cm2.or.dose.Gy</a> ).
material.no	index number for detector material (see also <a href="#">material.no</a> ).
stopping.power.source.no	stopping power source number (PSTAR,...) (see also <a href="#">stopping.power.source.no</a> ).
rdd.model	index number for chosen radial dose distribution (see also <a href="#">rdd.model</a> ).
rdd.parameters	parameters for chosen radial dose distribution (array of size 4).
er.model	index number for chosen electron-range model (see also <a href="#">er.model</a> ).
gamma.model	index number for chosen gamma response.
gamma.parameters	parameters for chosen gamma response (array of size 9).
saturation.cross.section.factor	scaling factor for the saturation cross section.
write.output	if true, a protocol is written to a file in the working directory.

**Value**

relative. efficiency	particle response at dose D / gamma response at dose D
S.HCP	absolute particle response
S.gamma	absolute gamma response
sI.cm2	resulting ion saturation cross section in cm2
gamma.dose.Gy	dose contribution from gamma kills
P.I	ion kill probability
P.g	gamma kill probability

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_Algorithms\\_IGK.c#L34](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_Algorithms_IGK.c#L34)

**Examples**

```
# Compute the relative efficiency of an Alanine detector in a mixed
# carbon / proton field
AT.run.IGK.method( particle.no           = 1001,
                  # namely protons with
                    E.MeV.u             = 10,
                  # 10 MeV/u
                    fluence.cm2.or.dose.Gy = c(-1.0),
                  # delivering 1 Gy
                    material.no          = 5,
                  # i.e. Alanine
                    rdd.model            = 4,

# Katz parametrization of radial dose distribution with simplified extended
# targets
                    rdd.parameter        = c(5e-8,1e-10),
                  # with 50 nm target size and 1e-10 dose minimum
                    er.model             = 2,
                  # Butts&Katz parametrization of track radius
                    gamma.model          = 2,
                  # Use general target/hit model but here...
                    gamma.parameters     = c(1,500,1,1,0),
                  # ...as exponential saturation with characteristic dose 500 Gy
                    saturation.cross.section.factor = 1.4,
                  # factor to take 'brush' around track into account
                    write.output         = TRUE,
                  # write a log file
                    stopping.power.source.no = 2)
```



---

AT.Rutherford.SDCS     *AT.Rutherford.SDCS*

---

### Description

Computes the Rutherford single differential cross section for the energy spectrum of secondary electrons produced by an HCP

### Usage

AT.Rutherford.SDCS(E.MeV.u, particle.no, material.no, T.MeV)

### Arguments

E.MeV.u	energy of particle per nucleon (see also <a href="#">E.MeV.u</a> ).
particle.no	particle index (see also <a href="#">particle.no</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
T.MeV	electron energies (array of size n).

### Value

dsdT.m2.MeV	Rutherford SDCS for given electron energies (array of size n)
status	status

### See Also

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L805](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L805)

---

AT.set.user.material     *AT.set.user.material*

---

### Description

Initializes user defined material. The material can then be used with material index number 0. !Be aware! that is definition is only valid during run-time. When the library is reloaded, the default settings are restored and the material is removed.

### Usage

AT.set.user.material(density.g.cm3, I.eV, average.A, average.Z)

**Arguments**

density.g.cm3	physical density in g per cm3.
I.eV	I value in eV.
average.A	average mass number.
average.Z	average atomic number.

**Value**

status	material defined successfully if zero
--------	---------------------------------------

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L479](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L479)

---

AT.set.user.material.from.composition

*AT.set.user.material.from.composition*

---

**Description**

Initializes user defined material from composition data. The material can then be used with material index number 0. !Be aware! that is definition is only valid during run-time. When the library is reloaded, the default settings are restored and the material is removed.

**Usage**

```
AT.set.user.material.from.composition(density.g.cm3, Z, A,
weight.fraction)
```

**Arguments**

density.g.cm3	physical density (in g per cm3) of material.
Z	atomic numbers of constituents (array of size n).
A	mass numbers of constituents (array of size n).
weight.fraction	relative fractions of weight of constituents (array of size n).

**Value**

status	material defined successfully if zero
--------	---------------------------------------

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataMaterial.c#L494](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataMaterial.c#L494)

---

AT.SPC.convert.to.DDD *AT.SPC.convert.to.DDD*

---

### Description

Converts spectral fluence data differential in depth and energy (spc) into a depth-dose. It also allows for the export of TRiP98 compatible DDD files. The routine calls AT.SPC.read so if you experience problems, please check if additional arguments for AT.SPC.read are needed (via '...'). For the DDD format, see M. Kramer's TRiP98 documentation: <http://bio.gsi.de/DOCS/TRiP98BEAM/DOCS/trip98fmtddd.html>.

### Usage

```
AT.SPC.convert.to.DDD(file.name.spc, file.name.ddd = NULL, endian =
  'little', plot = TRUE, write = TRUE, ...)
```

### Arguments

file.name.spc	path and file name for spc data
file.name.ddd	path and file name for ddd data, if not given file.name.spc will be used but with *.ddd extension (only applicable if write = TRUE)
endian	endianess of spc data, see also <a href="#">AT.SPC.read</a>
plot	if true, the resulting depth-dose curve will be plotted, e.g. for error-checking
write	if true, the resulting depth-dose curve will be exported in TRiP98 format
...	additional arguments for the <a href="#">AT.SPC.read</a> routine

### Value

A data frame containing the depth-dose curve

---

AT.SPC.export.DEDX *AT.SPC.export.DEDX*

---

### Description

Creates stopping power data tables from libamtrack and exports them in TRiP98 compatible format. For the DEDX format, see M. Kramer's TRiP98 documentation: <http://bio.gsi.de/DOCS/TRiP98BEAM/DOCS/trip98fmtdedx.html>. N.B. check the resulting data carefully for NAs, as not all stopping power data sources might cover the energy grid requested.

### Usage

```
AT.SPC.export.DEDX(stopping.power.source.no, file.name.DEDX = NULL,
  element.names = NULL, energy.MeV.u = NULL, plot = TRUE, write = TRUE)
```

**Arguments**

stopping.power.source.no	index of stopping power data source, see also <a href="#">stopping.power.source.no</a> .
file.name.DEDX	path and file name for DEDX data, if not given libamtrack.dedx will be used.
element.names	elements to be used (see <a href="#">particle.no</a> ), if not given the ICRU49/73 elements H...Ar will be used.
energy.MeV.u	energy grid, if not given the ICRU49/73 standard (0.025...1000 MeV/u in 51 steps) will be used.
plot	if true, the resulting data will be plotted.
write	if true, the resulting data will be exported in TRiP98 format.

**Value**

A data frame containing the stopping power data

---

AT.SPC.get

*AT.SPC.get*


---

**Description**

Returns interpolated spc data. Needs list of spc files as input which is obtained by [AT.SPC.get.list](#).  
 TODO: use path.to.files here alternative argument and call AT.SPC.get.list internally.

**Usage**

```
AT.SPC.get(spc.list, energy.MeV.u)
```

**Arguments**

spc.list	data frame with spc file information as returned by <a href="#">AT.SPC.get.list</a>
energy.MeV.u	requested beam energy, has to be within energy grid covered by spc.list

**Value**

list with spc data equivalent to those returned by [AT.SPC.read](#).

**Examples**

```
## Download data from libamtrack homepage
##
## To read a simple SPC file, use AT.SPC.read
##

# To get SPC data for arbitrary energy, you first have to make libamtrack aware
## of the appropriate SPC files:
#
```

```

# spc.list <- AT.SPC.get.list(path.to.files,...) where 'path.to.files'
# should point to spc files

#                               OF ONE KIND, i.e. same
# projectile, target, active/passive
#                               # but different energies

# N.B.: This routine can also be used to browser spc files and check their
# integrity
##
## Using the returned list, you can get spc data for any energy in between
#

# spc <- AT.SPC.get(spc.list, energy = ...) This will can AT.SPC.read
# and AT.SPC.interpolate
#
## returns a list in which the actual data are found in spc$spc
## which can then used in spc related routines, e.g.
## AT.SPC.tapply, AT.SPC.convert.to.DDD, etc.
##
## A spectrum at arbitraty can be obtain by using the data in spc$spc
## with AT.SPC.spectrum.at.depth.g.cm2

```

---

AT.SPC.get.list      *AT.SPC.get.list*

---

## Description

Returns filenames and essential information (energy, projectile etc.) of all spc files that are found at a user-proveded location. Needed by [AT.SPC.get](#).

## Usage

```
AT.SPC.get.list(path.to.files, endian = "little")
```

## Arguments

path.to.files    absolute path to spc files, including wildcard, e.g. \*.active\*.spc  
endian            byte-order of the spc file to read, "big" (AIX) or "little" (Linux, VMS, etc., default). Only necessary if flavour = 'vanilla'

## Value

A dataframe with path and file names, number of depth steps, projectile, target, beam energy, peak position.

**Examples**

```

## Download data from libamtrack homepage
##
## To read a simple SPC file, use AT.SPC.read
##

# To get SPC data for arbitrary energy, you first have to make libamtrack aware
## of the appropriate SPC files:
#

# spc.list <- AT.SPC.get.list(path.to.files,...) where 'path.to.files'
# should point to spc files

#                                     OF ONE KIND, i.e. same
# projectile, target, active/passive   # but different energies

# N.B.: This routine can also be used to browser spc files and check their
# integrity
##
## Using the returned list, you can get spc data for any energy in between
#

# spc <- AT.SPC.get(spc.list, energy = ...) This will can AT.SPC.read
# and AT.SPC.interpolate
#
## returns a list in which the actual data are found in spc$spc
## which can then used in spc related routines, e.g.
## AT.SPC.tapply, AT.SPC.convert.to.DDD, etc.
##
## A spectrum at arbitraty can be obtain by using the data in spc$spc
## with AT.SPC.spectrum.at.depth.g.cm2

```

---

AT.SPC.interpolate     *AT.SPC.interpolate*

---

**Description**

Interpolates between two spc files.

**Usage**

```
AT.SPC.interpolate(spc.lower, spc.upper, energy.MeV.u)
```

**Arguments**

spc.lower	spc data (read by AT.SPC.read) with lower bracketing energy.
spc.upper	spc data (read by AT.SPC.read) with upper bracketing energy.
energy.MeV.u	energy of interpolated spc data

**Value**

list equivalent to those returned by AT.SPC.read.

**Examples**

```
## Download data from libamtrack homepage
##
## To read a simple SPC file, use AT.SPC.read
##

# To get SPC data for arbitrary energy, you first have to make libamtrack aware
## of the appropriate SPC files:
#

# spc.list <- AT.SPC.get.list(path.to.files,...) where 'path.to.files'
# should point to spc files

#                               OF ONE KIND, i.e. same
# projectile, target, active/passive
#                               # but different energies

# N.B.: This routine can also be used to browser spc files and check their
# integrity
##
## Using the returned list, you can get spc data for any energy in between
#

# spc <- AT.SPC.get(spc.list, energy = ...) This will can AT.SPC.read
# and AT.SPC.interpolate
#
## returns a list in which the actual data are found in spc$spc
## which can then used in spc related routines, e.g.
## AT.SPC.tapply, AT.SPC.convert.to.DDD, etc.
##
## A spectrum at arbitrary can be obtain by using the data in spc$spc
## with AT.SPC.spectrum.at.depth.g.cm2
```

---

AT.SPC.read

*AT.SPC.read*


---

**Description**

Read a spc-formatted data file with energy-fluence in n depth steps. For original TRiP format definition by M. Kraemer, please see <http://bio.gsi.de/DOCS/TRiP98BEAM/DOCS/trip98fmtspc.html> and Kraemer and Scholz, Treatment planning for heavy-ion radiotherapy: calculation and optimization of biologically effective dose, Phys. Med. Biol. 45 (2000) 3319-3330. Please note that the user has to take care of picking the spc-file for the projectile and target material desired. Presently, two versions of the reader exists: a slower but stable R version ('vanilla') and a faster but still buggy C version. IMPORTANT: SPC files report DIFFERENTIAL fluences dN/dE per

primary particle, i.e. normalized by the energy bin. This reader converts them to ABSOLUTE fluences by multiplying dE. This facilitates e.g. summation to get the total fluence etc. but produces funny results when ABSOLUTE fluences are plotted. Also, SPCs only contain the left limit of the energy bins. This reader also reports the energy bin midpoint which is more reasonable e.g. for the computation of mean dE/dx etc. Please note that the "compress" statement has been depreciated due to preparation for C translation of spc handling.

### Usage

```
AT.SPC.read( file.name, flavour, endian, mean, header.only = FALSE)
```

### Arguments

file.name	name of spc-file to be read (with extension).
flavour	'vanilla' (default) to use stable R version.
endian	byte-order of the spc file to read, "big" (AIX) or "little" (Linux, VMS, etc., default). Only necessary if flavour = 'vanilla'
mean	method for computing bin midpoints, "geometric" or "arithmetic" (default).
header.only	if true, only information on spc file but no data will be read.

### Value

A list with (1) the 'spc' data frame with the following columns

depth.step	Index number of depth step (one-based)
depth.g.cm2	Depth in g/cm2
particle.no	Particle index number (see also( <a href="#">particle.no</a> )).
E.low.MeV.u	Energy bin lower limit in MeV/u (see also( <a href="#">E.MeV.u</a> )).
E.mid.MeV.u	Energy bin mid in MeV/u (see also( <a href="#">E.MeV.u</a> )).
E.high.MeV.u	Energy bin upper limit in MeV/u (see also( <a href="#">E.MeV.u</a> )).
dE.MeV.u	Energy bin width in MeV/u
dN.dE.per.MeV.u.per.primary	Fluence differential in energy (bin width) per primary.
N.per.primary	Fluence per primary.

and (2) variables containing information as beam energy, peak position, projectile and target material.

### See Also

[AT.SPC.tapply](#) for more example to derive quantities from spc-data.





```

FUN =
AT.dose.weighted.LET.MeV.cm2.g,
    additional.arguments =
list(c("material.no", "AT.material.no.from.material.name('Water, Liquid)'),
FALSE),

c("stopping.power.source.no", "0", FALSE)))$returnValue
df.total$fLET.MeV.cm2.g <- AT.SPC.tapply( spc = spc,
INDEX =
"depth.g.cm2",
FUN =
AT.fluence.weighted.LET.MeV.cm2.g,
    additional.arguments =
list(c("material.no", "AT.material.no.from.material.name('Water, Liquid)'),
FALSE),

c("stopping.power.source.no", "0", FALSE)))$returnValue
df.total$Mean.E.C.MeV.u <- AT.SPC.tapply( spc =
spc[spc$particle.no == 6012,],
INDEX =
"depth.g.cm2",
FUN =
AT.dose.weighted.E.MeV.u,
    additional.arguments =
list(c("material.no", "AT.material.no.from.material.name('Water, Liquid)'),
FALSE),

c("stopping.power.source.no", "0", FALSE)))$returnValue
xyplot( D.Gy ~ depth.g.cm2,
df.total,
grid = TRUE,
type = 'o',
ylab = 'total dose / Gy',
xlab = 'depth / (g/cm2)')
xyplot( LET.MeV.cm2.g + fLET.MeV.cm2.g ~ depth.g.cm2,
df.total,
type = 'o',
main = 'Fluence (fLET) and dose (LET) weighted LET',
grid = TRUE,
auto.key = list(space = 'right', lines = TRUE, points = FALSE),
ylab = 'LET / (MeV*cm2/g)',
xlab = 'depth / (g/cm2)')
xyplot( Mean.E.C.MeV.u ~ depth.g.cm2,
df.total,
type = 'o',
grid = TRUE,
main = file.name,
ylab = 'Average kinetic energy of C / (MeV/u)',
xlab = 'depth / (g/cm2)')

## End(Not run)

```

---

```
AT.SPC.spectrum.at.depth.g.cm2
    AT.SPC.spectrum.at.depth.g.cm2
```

---

### Description

Returns spectrum from spc data in given depth step.

### Usage

```
AT.SPC.spectrum.at.depth.g.cm2( spc, depth.g.cm2, interpolate = TRUE)
```

### Arguments

spc	spc data
depth.g.cm2	depth the spectrum should be taken at (up to now, the data from the closest depth step will be taken)
interpolate	if true, the spectra between the closest depth steps will be linearly interpolated, if false, the depth step just in front of the chosen depth will be taken directly

### Value

A data frame with the following columns (ready to use in most libamtrack functions):

E.MeV.u	Energy [MeV/u] - mid point of bin
particle.no	Particle index number
fluence.cm2	Fluence for each bin

### See Also

Uses [AT.SPC.spectrum.at.depth.step](#). spc data should be read in by [AT.SPC.read](#).

### Examples

```
# None yet.
```

---

```
AT.SPC.spectrum.at.depth.step
      AT.SPC.spectrum.at.depth.step
```

---

**Description**

Returns spectrum from spc data in given depth step.

**Usage**

```
AT.SPC.spectrum.at.depth.step( spc, depth.step)
```

**Arguments**

spc	spc data
depth.step	depth step the spectrum should be taken from (if outside data scope an error message will be issued)

**Value**

A data frame with the following columns (ready to use in most libamtrack functions):

E.MeV.u	Energy [MeV/u] - mid point of bin
particle.no	Particle index number
fluence.cm2	Fluence for each bin

**See Also**

Use [AT.SPC.spectrum.at.depth.g.cm2](#) to get spectrum at given depth. spc data should be read in by [AT.SPC.read](#).

**Examples**

```
# None yet.
```

---

```
AT.SPC.tapply      AT.SPC.tapply
```

---

**Description**

Similar to R's tapply this applies a function to cells defined by indices in a spc data object

**Usage**

```
AT.SPC.tapply( spc, INDEX, FUN, mixed.field.arguments = list(E.MeV.u =
  "E.mid.MeV.u", fluence.cm2 = "N.per.primary", particle.no = "particle.no"),
  additional.arguments = NULL, names.results = NULL)
```

**Arguments**

spc	spc data as returned by <code>AT.SPC.read</code>
INDEX	vector of column names in spc data that should be used as indices (indicating the cells).
FUN	Function to be applied - preferably this is a libamtrack function obeying the standard naming of mixed field variables <code>number.of.field.components</code> and <code>material.no</code> . Additional arguments to FUN can be passed using ...)
<code>mixed.field.arguments</code>	Named list containing the variables necessary to describe a mixed field (energy, fluence, particle type). The defaults are set to correspond to the column names used in <code>AT.SPC.read</code> . The user can change the list referring to other columns (of same length), e.g. when having multiplied the fluence per primary in the spc to get a realistic dose.
<code>additional.arguments</code>	Optional additional arguments to FUN. This should be a list with a three-entry vector for each argument. The first entry is the argument name, the second the value, the third indicated if it should be applied cell-wise (if TRUE). For example: the primitive R function to evaluate the mean energy per cell: <code>mean(x)</code> : <code>additional.arguments = list(c("x", "E.MeV.u", TRUE))</code> Or to pass the material.no (same for all cells): <code>additional.arguments = list(c("material.no", "1", FALSE))</code>
<code>names.results</code>	optional vector with names for the returned values of FUN

**Value**

A data frame with the following columns:

<code>index.columns</code>	Columns for indices given in INDEX to be looped over
<code>results[]</code>	Additional columns containing the returned values from FUN

**Examples**

```
## Not run:
# Load required libraries
require(libamtrack)
require(lattice)
# Use example data set
file.name      <- system.file("extdata",
  "libamtrack.12C.H2O.active3.MeV27000.zip", package = "libamtrack")
endian         <- c("big", "little")[2]
# Read in spc data, we use old style R reader, so endianness has to be given
spc           <- AT.SPC.read ( file.name = file.name,
                             endian    = endian,
                             flavour   = "vanilla")$spc
# Translate particle numbers in particle names (looks better)
spc$particle.name <- AT.particle.name.from.particle.no(particle.no =
  spc$particle.no)
spc$particle.name <- factor(spc$particle.name)

# Compute and plot dose per primary with depth from spc data
```

```

df1 <- AT.SPC.tapply( spc                = spc,
                     INDEX              = "depth.g.cm2",
                     FUN                = AT.total.D.Gy,
                     additional.arguments = list( c("material.no", "1",
FALSE), # Water
                     # Water

c("stopping.power.source.no", "0", FALSE)), # PSTAR
                     names.results     = "D.Gy")
xyplot( D.Gy ~ depth.g.cm2,
        df1,
        type = "o",
grid = TRUE,
        ylab = "dose per primary / Gy",
        xlab = "depth / (g/cm2)")
# Compute and plot dose (1 Gy entrance) with depth from spc data
# To do so, first the fluence per primary has to be scaled and then

# AT.SPC.tapply has to be referred to the new non-standard column (default:
# fluence.cm2 = "N.per.primary")
fluence.factor <- 1.0 / df1$D.Gy[1]
# factor between dose per primary (from above) and 2 Gy
spc$fluence.cm2 <- spc$N.per.primary * fluence.factor
df2 <- AT.SPC.tapply( spc                = spc,
                     INDEX              = "depth.g.cm2",
                     FUN                = AT.total.D.Gy,
                     mixed.field.arguments = list( fluence.cm2 = "fluence.cm2",
E.MeV.u      = "E.mid.MeV.u",
particle.no  = "particle.no"),
                     additional.arguments = list( c("material.no",
"1",
FALSE), # Water
c("stopping.power.source.no",
"0", FALSE)), # PSTAR
                     names.results     = "D.Gy")
xyplot( D.Gy ~ depth.g.cm2,
        df2,
        type = "o",
col = "red",
        grid = TRUE,
ylab = "dose / Gy",
        xlab = "depth / (g/cm2)")

# Compute and plot the dose with depth, but differentiate contribution
# from individual particle species
# Also: use nicer (human-readable) code for material
df3 <- AT.SPC.tapply( spc                = spc,
                     INDEX              = c("depth.g.cm2", "particle.name"),
                     FUN                = AT.total.D.Gy,
                     additional.arguments = list(c("material.no",
"AT.material.no.from.material.name('Water,
Liquid')",
FALSE),

```

```

                                c("stopping.power.source.no",
                                  "0",
                                  FALSE)),
                                names.results = "D.Gy")
xyplot( log10(D.Gy) ~ depth.g.cm2,
        df3,
        groups = particle.name,
        type = 'o',
        grid = TRUE,
        auto.key = list(space = 'right'),
        ylab = "log10( dose per primary / Gy )",
        xlab = 'depth / (g/cm2)')

## End(Not run)

```

---

AT.Stopping.Power      *AT.Stopping.Power*

---

## Description

Retrieves the electronic stopping power in keV/um for the requested energies and particles for a specified material and data source. The data source is thereby given via its name (s. `AT_StoppingPowerData.h` for details), except for data that should be read from a file, in this case the (path and) filename has to be provided. In this case, the user has to make sure that energy and stopping power units are correct and that the data match the given material (use `material.no = 0` for custom-defined material) for density scaling. The file has to be plain ASCII with three columns (separated by space) charge, energy, and stopping power and sorted in ascending order by first charge than energy any alphanumeric comment can be inserted (in separate lines)

## Usage

```
AT.Stopping.Power(stopping.power.source, E.MeV.u, particle.no,
                  material.no)
```

## Arguments

<code>stopping.power.source</code>	name of the data source (see also <a href="#">stopping.power.source</a> ).
<code>E.MeV.u</code>	kinetic energies in MeV per amu (array of size n) (see also <a href="#">E.MeV.u</a> ).
<code>particle.no</code>	particle numbers (array of size n) (see also <a href="#">particle.no</a> ).
<code>material.no</code>	material number (see also <a href="#">material.no</a> ).

## Value

<code>stopping.power.keV.um</code>	array to return stopping powers (array of size n)
<code>status</code>	status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_StoppingPower.c#L86](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_StoppingPower.c#L86)

---

AT.stopping.power.ratio

*AT.stopping.power.ratio*

---

**Description**

Computes the stopping power ratio for a material and a reference material. In case of mixed particle fields, the stopping power ratios of individual components are weighted by their respective fluences. Thus, this routines computes the ration of fluence-weighted stopping powers, NOT of dose-weighted stopping powers.

**Usage**

```
AT.stopping.power.ratio(E.MeV.u, particle.no, fluence.cm2, material.no,
    reference.material.no,
    stopping.power.source.no)
```

**Arguments**

E.MeV.u	energy of particles in the mixed particle field (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">E.MeV.u</a> ).
particle.no	particle index (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">particle.no</a> ).
fluence.cm2	fluences of particles in the mixed particle field (array of size <a href="#">number.of.field.components</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
reference.material.no	material index of reference material.
stopping.power.source.no	TODO (see also <a href="#">stopping.power.source.no</a> ).

**Value**

stopping	stopping
----------	----------

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L721](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L721)



**Examples**

```

df          <- expand.grid( particle.name      = "1H",
                          # Define parameter space:
                          particle.energy.MeV.u = 10^seq(-1, 3,
length.out = 500),        # 1 nuclid, energy between 0.1 and 1000 MeV/u,
                          material.name       = c("Air", "PMMA",
"Aluminum Oxide"),        # and three materials
                          stopping.power.ratio = 0)
df$particle.no <- AT.particle.no.from.particle.name(df$particle.name)
df$material.no <- AT.material.no.from.material.name(df$material.name)
material.no.water <- AT.material.no.from.material.name("Water, Liquid")
for (i in 1:nrow(df)){
  df$stopping.power.ratio[i] <- AT.stopping.power.ratio( E.MeV.u
    = df$particle.energy.MeV.u[i],

  particle.no      = df$particle.no[i],

  fluence.cm2      = 1,
# does not have any meaning here as monoenergetic beams are assumed

  material.no      = df$material.no[i],

  reference.material.no = material.no.water,

  stopping.power.source.no = 2)
}

```

---

AT.Stopping.Power.with.no

*AT.Stopping.Power.with.no*


---

**Description**

Retrieves the electronic stopping power in keV/um for the requested energies and particles for a specified material and data source. The data source is thereby given via its integer id (s. `AT_StoppingPowerData.h` for details). Data that should be read from a file cannot be used with this method.

**Usage**

```

AT.Stopping.Power.with.no(stopping.power.source.no, E.MeV.u,
  particle.no, material.no)

```

**Arguments**

```

stopping.power.source.no
    id of the data source (see also stopping.power.source.no).
E.MeV.u
    kinetic energies in MeV per amu (array of size n) (see also E.MeV.u).
particle.no
    particle numbers (array of size n) (see also particle.no).
material.no
    material number (see also material.no).

```

**Value**

stopping.power.keV.um  
array to return stopping powers (array of size n)

status status

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_StoppingPower.c#L128](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_StoppingPower.c#L128)

---

AT.total.D.Gy	<i>AT.total.D.Gy</i>
---------------	----------------------

---

**Description**

Computes the total dose of a mixed particle field

**Usage**

AT.total.D.Gy(E.MeV.u, particle.no, fluence.cm2, material.no,  
stopping.power.source.no)

**Arguments**

E.MeV.u energy of particles in the mixed particle field (array of size [number.of.field.components](#)) (see also [E.MeV.u](#)).

particle.no particle index (array of size [number.of.field.components](#)) (see also [particle.no](#)).

fluence.cm2 fluences of particles in the mixed particle field (array of size [number.of.field.components](#)).

material.no material index (see also [material.no](#)).

stopping.power.source.no  
TODO (see also [stopping.power.source.no](#)).

**Value**

total.dose.Gy total.dose.Gy

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L521](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L521)

**Examples**

```
# Compute dose of monoenergetic high energy
# and a mixed (99/1) field of high and low
# energy protons in water at same fluence
AT.total.D.Gy( E.MeV.u      = 100,
               particle.no = 1001,
               fluence.cm2 = 100e8,
               material.no  = 1,
               stopping.power.source.no = 2)
AT.total.D.Gy( E.MeV.u      = c(100, 5),
               particle.no = c(1001, 1001),
               fluence.cm2 = c(99e8, 1e8),
               material.no  = 1,
               stopping.power.source.no = 2)
```

---

```
AT.total.fluence.cm2  AT.total.fluence.cm2
```

---

**Description**

Computes the total fluence of a mixed particle field

**Usage**

```
AT.total.fluence.cm2(E.MeV.u, particle.no, D.Gy, material.no,
                    stopping.power.source.no)
```

**Arguments**

E.MeV.u	energy of particles in the mixed particle field (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">E.MeV.u</a> ).
particle.no	particle index (array of size <a href="#">number.of.field.components</a> ) (see also <a href="#">particle.no</a> ).
D.Gy	doses of particles in the mixed particle field (array of size <a href="#">number.of.field.components</a> ).
material.no	material index (see also <a href="#">material.no</a> ).
stopping.power.source.no	TODO (see also <a href="#">stopping.power.source.no</a> ).

**Value**

```
total.fluence.cm
total.fluence.cm
```

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_PhysicsRoutines.c#L549](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_PhysicsRoutines.c#L549)

**Examples**

```
# Compute total fluence in a
# and a mixed field of
# high and low energy protons
# that deliver the same dose to water
AT.total.fluence.cm2( E.MeV.u      = c(100, 5),
                    particle.no = c(1001, 1001),
                    D.Gy        = c(1, 1),
                    material.no = 1,
                    stopping.power.source.no = 2)
```

---

```
AT.translate.dose.into.DSB.distribution
      AT.translate.dose.into.DSB.distribution
```

---

**Description**

Converts a local dose into a DSB distribution assuming Poissonian rule for creation.

**Usage**

```
AT.translate.dose.into.DSB.distribution(f.d.Gy, f.dd.Gy, f,
  enhancement.factor, DSB.per.Gy.per.domain,
  domains.per.nucleus, write.output)
```

**Arguments**

f.d.Gy	bin midpoints for f (array of size n.bins.f).
f.dd.Gy	bin widths for f (array of size n.bins.f).
f	dose frequency (array of size n.bins.f).
enhancement.factor	dose enhancement factor (array of size n.bins.f).
DSB.per.Gy.per.domain	number of DSBs per domain per Gy.
domains.per.nucleus	number of domains in nucleus.
write.output	if true, a log file will be written ("dose.to.DSBs.log") containing the DSB distribution.

**Value**

total.pDSBs	probability sum of DSB probability (quality check, has to be ~1)
total.nDSBs	number of DSBs in nucleus
number.of.iDSBs	number of isolated DSBs in nucleus

number.of.cDSBs  
                  number of complex DSBs in nucleus  
avg.number.of.DSBs.in.cDSBs  
                  average number of DSBs in complex DSBs

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_SuccessiveConvolutions.c#L1130](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_SuccessiveConvolutions.c#L1130)

---

AT.Vavilov.IDF

*AT.Vavilov.IDF*

---

**Description**

Computes the Vavilov probability density function using CERNLIB (G116)

**Usage**

AT.Vavilov.IDF(rnd, kappa, beta)

**Arguments**

rnd               random number from uniform distribution between 0 and 1 (array of size n).  
kappa             straggling parameter (array of size n).  
beta              relativistic speed, between 0 and 1 (array of size n).

**Value**

lambda.vavilov   resulting Vavilov lambda (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L279](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L279)

---

 AT.Vavilov.PDF

*AT.Vavilov.PDF*


---

**Description**

Computes the Vavilov probability density function using CERNLIB (G116)

**Usage**

AT.Vavilov.PDF(lambda.vavilov, kappa, beta)

**Arguments**

lambda.vavilov Vavilov lambda (array of size n).  
 kappa straggling parameter.  
 beta relativistic speed, between 0 and 1.

**Value**

density resulting density (array of size n)

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L269](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L269)

---

 AT.xi.keV

*AT.xi.keV*


---

**Description**

Parameter xi - reduced mean energy loss

**Usage**

AT.xi.keV(E.MeV.u, particle.no, material.no, slab.thickness.um)

**Arguments**

E.MeV.u energies of particle per nucleon (see also [E.MeV.u](#)).  
 particle.no particle indices (see also [particle.no](#)).  
 material.no material index (see also [material.no](#)).  
 slab.thickness.um  
     slab thickness in um.

**Value**

xi	xi
----	----

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_EnergyLoss.c#L26](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_EnergyLoss.c#L26)

---

AT.Z.from.particle.no *AT.Z.from.particle.no*

---

**Description**

Returns atomic number for given particle number

**Usage**

AT.Z.from.particle.no(particle.no)

**Arguments**

particle.no	particle index number (array of size n) (see also <a href="#">particle.no</a> ).
-------------	--

**Value**

Z	atomic number (array of size n)
return	return

**See Also**

View the C source code here: [http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT\\_DataParticle.c#L81](http://sourceforge.net/apps/trac/libamtrack/browser/tags/0.6.3/src/AT_DataParticle.c#L81)

---

E.MeV.u	<i>Ion energy values</i>
---------	--------------------------

---

**Description**

In libamtrack, the kinetic energy for particles is usually given in Megaelectronvolt per nucleon (MeV/u). For therapeutical beam these are usually in range between approx. 1 MeV/u and 500 MeV/u. Please note, that simply the number of nucleons is used, not the actual mass of the nucleon (amu) for a specific nuclid. So, for example for U-238 with 100 MeV/u the total kinetic energy is 23.8 GeV.

---

er.model

*Electron-range models*


---

### Description

In libamtrack, there are currently seven parametrizations of the maximal distance electrons travel from the center of a particle track as a function of the primary particle's energy. They are referred to using sequential positive integer numbers (see details).

### Details

er.model	description
1	A simple test function (constant range)
2	Parametrization proposed by Butts and Katz (Butts & Katz, 1972)
3	Parametrization proposed by M. Waligorski (Waligorski, 1985)
4	Parametrization proposed by O. Geiss (Geiss, 1997)
5	Parametrization proposed by M. Scholz (Scholz, 1997)
6	Parametrization proposed by J. Edmund (Edmund et al., 2007)
7	Parametrization proposed by Tabata (Tabata, 1972)

### See Also

More information on the electron-range models, especially the explicit formulas used, valid energy ranges and references to literature are found in the libamtrack reference manual (<http://libamtrack.dkfz.org/libamtrack/images/3/31/LibamtrackReferenceManual.pdf>).

---

fluence.cm2.or.dose.Gy

*Fluence and dose as equivalent input parameters*


---

### Description

Many functions in libamtrack can take either the fluence or the dose as an argument. For a specific nuclid and energy they can be converted into each other (see also `AT.dose.Gy()` and `AT.fluence.cm2()`). As both can only be positive, the argument `fluence.cm2.or.dose.Gy` will be interpreted as fluence [1/cm2] if positive and as dose [Gy] if given as negative value.

- `fluence.cm2.or.dose.Gy <- 1e7` means fluence = 1e7 / cm2
- `fluence.cm2.or.dose.Gy <- -10` means dose = 10 Gy



---

gamma.model	<i>Gamma (X ray) response models</i>
-------------	--------------------------------------

---

## Description

In libamtrack, there are currently six parametrizations of the gamma / X ray response of a system. They are referred to using sequential positive integer numbers (see details).

## Details

**The implemented gamma models are:**

gamma.model	description
1	A simple test function (constant over entire track width/electron range)
2	Generalized hit/target model
3	Radioluminescence response
4	Simple exponential saturation (special case of 2: one hit / one target)
5	Linear-quadratic model
6	Mejdahl's TLD response for LiF as used by Geiss et al. (1998)

**The corresponding gamma parameters are:**

gamma.model	gamma.parameters[1]	gamma.parameters[2]	gamma.parameters[3]	gamma.parameters[4]	gamma.parameters[5]
1	NA	NA	NA	NA	NA
2	rel. response	characteristic dose [Gy]	hittedness	multiplicity	0 to 1
3	max. response	transition dose [Gy]	dynamic	NA	NA
4	max. response	characteristic dose [Gy]	NA	NA	NA
5	alpha [1/Gy]	beta[1/Gy <sup>2</sup> ]	transition dose [Gy]	NA	NA
6	const	k1	a1	k2	a2

**The corresponding default/example values are:**

gamma.model	gamma.parameters[1]	gamma.parameters[2]	gamma.parameters[3]	gamma.parameters[4]	gamma.parameters[5]
1	NA	NA	NA	NA	NA
2	1.0	10.0	1	1	0
3	1e6	20.0	6.0	NA	NA
4	1e6	10.0	NA	NA	NA
5	0.2	0.018	30.0	NA	NA
6	6e3	0.59	5e-4	0.41	2e-4

## See Also

More information on models, especially the explicit formulas used, valid energy ranges and references to literature are found in the libamtrack reference manual (<http://libamtrack.dkfz.org/libamtrack/images/3/31/LibamtrackReferenceManual.pdf>).

---

material.name                    *Material naming convention*

---

### Description

Currently seven predefined materials exist in libamtrack. They are referred to using sequential positive integer numbers (see also [particle.no](#)) but carry also a name, preferably the NIST name. When referring to a material by name, make sure that the spelling (incl. cases) is correct.

### Details

The predefined materials in libamtrack are:

material.no	material.name
1	"Water, Liquid"
2	"Aluminum Oxide"
3	"Aluminum"
4	"PMMA"
5	"Alanine"
6	"Lithium Fluoride"
7	"Air"

---

material.no                    *Material indexing convention*

---

### Description

Currently seven predefined materials exist in libamtrack. They are referred to using sequential positive integer numbers (see details). The material definition follows largely the pstar and astar definitions of NIST (<http://physics.nist.gov/cgi-bin/Star/compos.pl?ap>). material.no = 0 points in contrast to a material defined by the user during run-time. This feature is still under development and not available yet.

### Details

The predefined materials in libamtrack are:

material.no	<b>material</b>
1	Water, Liquid
2	Aluminum Oxide (Al <sub>2</sub> O <sub>3</sub> )
3	Aluminum
4	PMMA (Polymethylmethacrylate, Perspex, Plexiglass, [C <sub>5</sub> H <sub>8</sub> O <sub>2</sub> ] <sub>n</sub> )
5	Alanine
6	Lithium Fluoride (LiF)
7	Air (dry, at sea level)

- 8 Silicon
- 9 Copper

### See Also

You can use `AT.material.name.from.material.no` and `AT.material.no.from.material.name` to convert material names into material indices and vice versa. `AT.get.materials.data` will return all available data for a predefined material.

---

number.of.field.components

*Handling of mixed particle fields*

---

### Description

Many functions in libamtrack offer the possibility to consider mixed particle fields. These fields are composed from a finite number of specific nuclids defined by `particle.no`. All particles of an individual component are supposed to have exactly the same energy (given by `particle.no`), without spread. Thus an energy distribution has to be describe in discrete energy steps. Below is a compilation of the arguments a mixed field is described by. Depending on the function all or a subset are necessary. If arrays of length  $>$  `number.of.field.components` are entered, the additional data will be ignored. If array hold too few data, the computation will not be executed.

### Arguments

number.of.field.components

A variable describing how many components are used for the field. This also dictates the array lengths of all following variables.

E.MeV.u

Particle energies (array of size `number.of.field.components`, see also `E.MeV.u`)

particle.no

Particle type indices (array of size `number.of.field.components`), see also `particle.no`

fluence.cm2.or.dose.Gy

Fluences (if positive), or doses (if negative, array of size `number.of.field.components`), see also `fluence.cm2.or.dose.Gy`

---

particle.name

*Particle naming convention*

---

### Description

The particle naming system in libamtrack works as follows: a particle is identified by a character string consisting of the mass number  $A$  and the abbreviation of the element (identifying the atomic number  $Z$ ).

**Details**

- For a proton `particle.name` is '1H'.
- For deuterium `particle.name` is '2H'.
- For He-4 `particle.name` is '4He'.
- For C-12 `particle.name` is '12C'.
- For a U-238 `particle.name` is '238U'.

---

<code>particle.no</code>	<i>Particle indexing convention</i>
--------------------------	-------------------------------------

---

**Description**

The particle indexing system in libamtrack works as follows: for a particle with atomic number  $Z$  and mass number  $A$  the index is `particle.no` =  $1000 * Z + A$ . This ensures uniqueness for all particle types used in libamtrack. Also, an extension to anti-protons or ions could be made using negative numbers.

**Details**

- For a proton ( $Z=1, A=1$ ) `particle.no` is 1001.
- For deuterium ( $Z=1, A=2$ ) `particle.no` is 1002.
- For He-4 ( $Z=2, A=4$ ) `particle.no` is 2004.
- For C-12 ( $Z=6, A=12$ ) `particle.no` is 6012.
- For a U-238 ( $Z=92, A=238$ ) `particle.no` is 92238.

**See Also**

You can use `AT.particle.name.from.particle.no` and `AT.particle.no.from.particle.name` to convert particle names into particle indices and vice versa.

---

<code>rdd.model</code>	<i>Radial dose distribution (rdd) models</i>
------------------------	--

---

**Description**

In libamtrack, there are currently seven parametrizations of distribution of local dose around particle track as a function of the primary particle's energy and type. They are referred to using sequential positive integer numbers (see details).

**Details**

**The implemented rdd models are:**

rdd.model	description
1	A simple test function (constant over entire track width/electron range)
2	Point target rdd proposed by Butts and Katz (Butts & Katz, 1969)
3	Simplified extended target rdd proposed by O. Geiss (Geiss, 1997)
4	Simplified extended target rdd proposed by Katz (Katz et al., 1972)
5	Point target rdd proposed by Cucinotta (Cucinotta, XXXX)
6	Extended target rdd proposed by Katz (Katz, XXXX)
7	Extended target rdd proposed by Cucinotta (Cucinotta, XXXX)

**The corresponding rdd parameters are:**

rdd.model	rdd.parameters[1]	rdd.parameters[2]	rdd.parameters[3]
1	NA	NA	NA
2	minimal radius (integration limit) [m]	lower dose cut-off [Gy]	NA
3	core diameter/target size [m]	NA	NA
4	core diameter/target size [m]	lower dose cut-off [Gy]	NA
5	minimal radius (integration limit) [m]	lower dose cut-off [Gy]	NA
6	minimal radius (integration limit) [m]	core diameter/target size [m]	lower dose cut-off [Gy]
7	minimal radius (integration limit) [m]	core diameter/target size [m]	lower dose cut-off [Gy]

**The corresponding default values are:**

rdd.model	rdd.parameters[1]	rdd.parameters[2]	rdd.parameters[3]
1	NA	NA	NA
2	1e-10	1e-10	NA
3	5e-8	NA	NA
4	5e-8	1e-10	NA
5	5e-11	1e-10	NA
6	1e-10	1e-8	1e-10
7	5e-11	1e-8	1e-10

**See Also**

More information on the radial dose distribution models, especially the explicit formulas used, valid energy ranges and references to literature are found in the libamtrack reference manual (<http://libamtrack.dkfz.org/libamtrack/images/3/31/LibamtrackReferenceManual.pdf>).

---

stopping.power.source *Sources of stopping power data*

---

**Description**

Three stopping power data sources are implemented in libamtrack. They are referred to using keywords or integer identifiers (see below). In addition, the user can provide a filename point to custom data in text format. The built-in sources provide electronic mass stopping power as a function of specific energy (details see below).

Scaling with density (or use of material parameters by Bethe formula) is done using the material specified. In case of custom stopping power data, the user has either to make sure that the right data (i.e. electronic mass stopping power) are provided with the correct units (E/u and MeV\*cm<sup>2</sup>/g) OR use any kind of data / units if they know what they are doing... (`material.no = 0`).

Built-in data are given in specific energy per amu for the most frequent isotope (i.e. H-1, He-4 etc.) or directly in specific energy per nucleon (i.e. in case of the Bethe formula). Scaling to other isotopes happens via the number of nucleons. Thus, effects of changes in binding energy per nucleon are neglected. These are usually small (0.7% important for high-accuracy application). The same is true for the material parameters, e.g. the I-value. In that case, it is advised to use external tabulated data.

These tabulated data have to be plain ASCII with three columns (separated by space): charge, energy, and stopping power. The data have to be sorted in ascending order by first charge than energy. In addition, any alphanumeric comment can be inserted (in separate lines). Stopping power data handling underwent a major rework in v0.6.0 and identifiers were changed. To avoid confusion, downward compatibility might be not maintained and integer identifiers deprecated.

## Details

### The stopping power sources are:

<code>stopping.power.source.no</code>	<code>stopping.power.source</code>	<b>description</b>
—	<FILENAME>	data are read from external file (see details)
1	Bethe	Analytical Bethe formula, including density effect, for any material
2	PSTAR	Tabulated data from NIST's PSTAR code, for other ions scales by density
3	ICRU	Tabulated data from ICRU49 (H, He) and ICRU73, for liquid water
—	libEdx_SOURCE	Reserved keyword for later use with libEdx and data source SOURCE

## See Also

More information, especially the explicit formulas used, valid energy ranges and references to literature are found in the libamtrack reference manual (<http://libamtrack.dkfz.org/libamtrack/images/3/31/LibamtrackReferenceManual.pdf>).

# Index

\*Topic **Sources of stopping power data**

stopping.power.source, 93

\*Topic **electron-range models**

er.model, 88

\*Topic **gamma response models**

gamma.model, 89

\*Topic **material indexing**

material.no, 90

\*Topic **material naming**

material.name, 90

\*Topic **package**

libamtrack-package, 4

\*Topic **particle indexing**

particle.no, 92

\*Topic **particle naming**

particle.name, 91

\*Topic **radial dose distribution models**

rdd.model, 92

AT.A.from.particle.no, 6, 8

AT.add.leading.zeros, 6, 9

AT.add.trailing.zeros, 6, 10

AT.atomic.weight.from.Z, 10

AT.average.A.from.composition, 6, 11

AT.average.Z.from.composition, 6, 11

AT.beam.par.physical.to.technical, 5, 12

AT.beam.par.technical.to.physical, 5, 13

AT.beta.from.E, 5, 14

AT.CPPSC.alpha.and.beta, 4, 15

AT.CSDA.range.g.cm2, 5, 16

AT.D.RDD.Gy, 4, 17

AT.dose.Gy.from.fluence.cm2, 5, 18

AT.dose.weighted.E.MeV.u, 5, 19

AT.dose.weighted.LET.MeV.cm2.g, 5, 20

AT.E.from.beta, 5, 21

AT.E.MeV.u.from.momentum.MeV.c.u, 5, 22

AT.effective.charge.from.E.MeV.u, 5, 23

AT.effective.Z.from.composition, 6, 24

AT.electron.density.m3, 6, 24

AT.electron.density.m3.from.composition, 6, 25

AT.electron.density.m3.from.material.no, 6, 26

AT.energy.loss.from.lambda.gauss, 26

AT.energy.loss.from.lambda.landau, 27

AT.energy.loss.from.lambda.vavilov, 28

AT.energy.straggling.after.slab.E.MeV.u, 28

AT.energy.straggling.MeV2.cm2.g, 29

AT.fluence.cm2.from.dose.Gy, 5, 30

AT.fluence.weighted.E.MeV.u, 5, 31

AT.fluence.weighted.LET.MeV.cm2.g, 5, 32

AT.FLUKA.particle.name.to.libamtrack.particle.name, 5, 33

AT.FLUKA.read.USRBIN.mesh, 5, 33

AT.FLUKA.read.USRBIN.regs, 5, 34

AT.FLUKA.read.USRTRACK, 5, 35

AT.gamma.from.E, 5, 36

AT.gamma.response, 4, 37

AT.Gauss.IDF, 39

AT.Gauss.PDF, 39

AT.get.materials.data, 6, 40

AT.I.eV.from.composition, 6, 41

AT.inverse.gamma.response, 4, 41

AT.kappa, 42

AT.lambda.landau.from.energy.loss, 43

AT.lambda.max, 43

AT.lambda.mean, 44

AT.lambda.vavilov.from.energy.loss, 45

AT.Landau.IDF, 45

AT.Landau.PDF, 46

AT.Mass.Stopping.Power, 5, 47

AT.Mass.Stopping.Power.with.no, 5, 48

AT.material.name.from.material.no, 6, 49

- AT.material.no.from.material.name, [6](#), [50](#)
- AT.max.E.transfer.MeV, [5](#), [50](#)
- AT.max.E.transfer.MeV.new, [51](#)
- AT.max.electron.ranges.m, [4](#), [52](#)
- AT.mean.energy.loss.keV, [53](#)
- AT.mean.number.of.tracks.contrib, [5](#), [53](#)
- AT.momentum.MeV.c.u.from.E.MeV.u, [5](#), [54](#)
- AT.nuclear.spin.from.particle.no, [6](#), [55](#)
- AT.particle.name.from.particle.no, [6](#), [55](#)
- AT.particle.no.from.particle.name, [6](#), [56](#)
- AT.particle.no.from.Z.and.A, [6](#), [57](#)
- AT.r.RDD.m, [4](#), [57](#)
- AT.run.CPPSC.method, [4](#), [15](#), [58](#)
- AT.run.GSM.method, [4](#), [60](#)
- AT.run.IGK.method, [4](#), [63](#)
- AT.Rutherford.SDCS, [5](#), [65](#)
- AT.set.user.material, [6](#), [65](#)
- AT.set.user.material.from.composition, [6](#), [66](#)
- AT.SPC.convert.to.DDD, [67](#)
- AT.SPC.export.DEDX, [67](#)
- AT.SPC.get, [68](#), [69](#)
- AT.SPC.get.list, [68](#), [69](#)
- AT.SPC.interpolate, [70](#)
- AT.SPC.read, [4](#), [67](#), [68](#), [71](#), [75–77](#)
- AT.SPC.spectrum.at.depth.g.cm2, [4](#), [75](#), [76](#)
- AT.SPC.spectrum.at.depth.step, [4](#), [75](#), [76](#)
- AT.SPC.tapply, [4](#), [72](#), [76](#)
- AT.Stopping.Power, [5](#), [79](#)
- AT.stopping.power.ratio, [5](#), [80](#)
- AT.Stopping.Power.with.no, [5](#), [81](#)
- AT.total.D.Gy, [5](#), [82](#)
- AT.total.fluence.cm2, [5](#), [83](#)
- AT.translate.dose.into.DSB.distribution, [84](#)
- AT.Vavilov.IDF, [85](#)
- AT.Vavilov.PDF, [86](#)
- AT.xi.keV, [86](#)
- AT.Z.from.particle.no, [6](#), [87](#)
- E.MeV.u, [14](#), [17](#), [19–21](#), [23](#), [26–32](#), [36](#), [42–45](#), [47](#), [49–54](#), [58](#), [61](#), [63](#), [65](#), [72](#), [79–83](#), [86](#), [87](#), [91](#)
- electron-range model index (er.model), [88](#)
- electron-range model numbering (er.model), [88](#)
- electron-range models (er.model), [88](#)
- er.model, [17](#), [52](#), [54](#), [58](#), [59](#), [61](#), [63](#), [88](#)
- Fluence or dose argument (fluence.cm2.or.dose.Gy), [88](#)
- fluence.cm2.or.dose.Gy, [59](#), [61](#), [63](#), [88](#), [91](#)
- gamma response model index (gamma.model), [89](#)
- gamma response model numbering (gamma.model), [89](#)
- gamma response models (gamma.model), [89](#)
- gamma.model, [89](#)
- Handling of mixed particle fields (number.of.field.components), [91](#)
- Ion energies (E.MeV.u), [87](#)
- libamtrack (libamtrack-package), [4](#)
- libamtrack-package, [4](#)
- material index (material.no), [90](#)
- material naming (material.name), [90](#)
- material number (material.no), [90](#)
- material numbering (material.no), [90](#)
- material.name, [49](#), [50](#), [90](#)
- material.no, [16](#), [17](#), [19–21](#), [26–30](#), [32](#), [40](#), [42–45](#), [47](#), [49](#), [50](#), [52–54](#), [58](#), [59](#), [61](#), [63](#), [65](#), [77](#), [79–83](#), [86](#), [90](#), [94](#)
- number.of.field.components, [20](#), [21](#), [31](#), [32](#), [54](#), [58](#), [59](#), [61](#), [63](#), [77](#), [80](#), [82](#), [83](#), [91](#)
- particle index (particle.no), [92](#)
- particle names (particle.name), [91](#)
- particle naming (particle.name), [91](#)
- particle number (particle.no), [92](#)
- particle numbering (particle.no), [92](#)
- particle.name, [56](#), [91](#)
- particle.no, [8](#), [16](#), [17](#), [19–21](#), [23](#), [26–30](#), [32](#), [35](#), [42–45](#), [47](#), [49](#), [53–56](#), [58](#), [61](#), [63](#), [65](#), [68](#), [72](#), [79–83](#), [86](#), [87](#), [90](#), [91](#), [92](#)
- radial dose distribution model index (rdd.model), [92](#)



- radial dose distribution model
  - numbering (rdd.model), [92](#)
- radial dose distribution models
  - (rdd.model), [92](#)
- rdd.model, [17](#), [58](#), [59](#), [61](#), [63](#), [92](#)
  
- Source index of stopping power data
  - (stopping.power.source), [93](#)
- Sources of stopping power data
  - (stopping.power.source), [93](#)
- stopping.power.source, [47](#), [79](#), [93](#)
- stopping.power.source.no, [17](#), [19–21](#), [30](#),  
[32](#), [49](#), [54](#), [58](#), [59](#), [61](#), [63](#), [68](#), [80–83](#)
  
- X ray response models (gamma.model), [89](#)