Package: terms (via r-universe)

September 4, 2024

Version 1.0.2

License Mozilla Public License Version 2.0

Title T-matrix for Electromagnetic Radiation with Multiple Scatterers

Description A set of Fortran modules/routines for T-matrix-based calculations of light scattering by clusters of individual scatterers.

URL https://github.com/nano-optics/terms, http://nano-optics.ac.nz/terms/

BugReports https://github.com/nano-optics/terms/issues

Type Package

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.5.0), rhdf5, cubs, dielectric, tidyr, purrr, glue, dplyr, tibble, ggplot2

Suggests stringr, patchwork, gridExtra, egg, ggforce, DT, rgl, knitr, rmarkdown

Repository https://nano-optics.r-universe.dev

RemoteUrl https://github.com/nano-optics/terms

RemoteRef HEAD

RemoteSha 0aa3b67836d5fb6c84b27783b484078cf6d66821

Contents

`cluster_positions` *Clusters of particles*

Description

Defines various cluster geometries and exports in a format suitable for TERMS

Helix of particles

Chain of particles

Core-satellite cluster of spheres

Usage

```
cluster_positions(
  N = 5,
  cl_fun = cluster_chain,
  radius = 50,
  label = "Au",
  ...,
  out = "",
  digits = 8
)

cluster_helix(
  N = 5,
  a = 10,
  b = 10,
  c = 50,
  R0 = 100,
  pitch = 200,
  delta = pi/5,
  delta0 = 0,
  hand = 1,
  ...
)

cluster_chain(
  N = 5,
  pitch = 500,
  a = 50,
  b = 30,
  c = b,
  rot = rotation_euler_passive(0, 0, 0)
)

cluster_satellite(N = 30, Rcore = 30, Rsat = 4, gap = 0.1, exclusion = 10, ...)
```

Arguments

N	number of particles
cl_fun	cluster function
radius	particle radius
label	particle material label
...	extra arguments passed to cl_fun
out	filename
digits	accuracy
a	semi-axis
b	semi-axis
c	semi-axis
R0	helix radius
pitch	chain pitch
delta	helix angle step
delta0	helix start angle
hand	helix handedness
rot	rotation matrix applied to each particle
Rcore	core radius
Rsat	satellite radius
gap	gap distance
exclusion	minimum exclusion distance for hc positions

Value

returns scatterers positions and sizes for an input file

Functions

- `cluster_positions()`: write cluster positions to input file
- `cluster_helix()`: helical cluster
- `cluster_chain()`: linear chain cluster
- `cluster_satellite()`: core-satellite cluster

Examples

```
cluster_positions()
cluster_helix()
cluster_chain()
cluster_satellite()
```

`equal_sizes` *Utility functions*

Description

Matrix of equal sizes

Matrix of equal angles

Combined T-matrix index

T-matrix indices

Unpack T-matrix indices

Unpack T-matrix indices

Wrap staged matrices into a list of T-matrix like objects

Read T-matrix into long-format data.frame

Display staged matrix

Display T-matrix

Display prestaged matrix

Generate an incidence file for spherical cubature

Generate a dielectric function in suitable format for TERMS

Euler rotation matrix

Euler rotation matrix

Axis-angle rotation from Euler angles

Extend a range symmetrically about 0

Extract geometry information from input file

Wrap geometry information into a 'cluster' structure

Visualise a 'cluster' structure

Usage

```
equal_sizes(a, b, c, N)

equal_angles(phi, theta, gamma, N)

p_index(in1, in2)

indices(n_max = 3, n_part = 1)

unpack_indices(n_max = 3, j_max = 2)

read_amat(f)
```

```
amat_to_tmatlist(a, n_max = 3, n_part = 2)

read_tmat(f, save = FALSE)

display_amat(l)

display_tmat(s)

display_prestaged(a, n_max = 3, n_part = 2, draw = TRUE)

export_cubature(q = cubs::cubs(N = 10, cubature = "lebedev"), out = "")

export_dielectric(m = dielectric::epsAu(seq(400, 800)), out = "")

rotation_euler_passive(phi, theta, psi)

rotation_euler_active(phi, theta, psi)

euler_to_axisangle(a, b, c)

symmetric_range(range)

get_geometry(input = "input")

cluster_geometry(ge)

visualise_rgl(cl, outfile = NULL, show_core = FALSE, ...)
```

Arguments

a	Euler angle
b	Euler angle
c	Euler angle
N	number of particles
phi	Euler angle
theta	Euler angle
gamma	Euler angle
in1	index
in2	index
n_max	maximum order
n_part	number of particles
j_max	size of collective T-matrix
f	filename
save	store result as Rds file
l	list of T-matrices, from amat_to_tmatlist

s	T-matrices, from read_tmat
draw	logical, draw output
q	data.frame with angles and weights, from cubs::cubs()
out	filename
m	data.frame with wavelength and epsilon, e.g. from dielectric::epsAu()
psi	Euler angle
range	range (2-vector)
input	filename
ge	geometry, from get_geometry
cl	'cluster' object
outfile	optional output snapshot
show_core	display a core sphere if 'R0' field present
...	additional parameters passed to rgl.ellipsoids

Functions

- `equal_sizes()`: equal sizes
- `equal_angles()`: equal angles
- `p_index()`: p-index
- `indices()`: indices
- `unpack_indices()`: unpack indices
- `read_amat()`: unpack indices
- `amat_to_tmatlist()`: wrap staged matrices
- `read_tmat()`: read T-matrix
- `display_amat()`: display staged matrix
- `display_tmat()`: display T-matrix
- `display_prestaged()`: display prestaged matrix
- `export_cubature()`: export a spherical cubature
- `export_dielectric()`: export a dielectric function
- `rotation_euler_passive()`: passive rotation matrix
- `rotation_euler_active()`: active rotation matrix
- `euler_to_axisangle()`: axis-angle rotation
- `symmetric_range()`: symmetric range
- `get_geometry()`: extract geometry information from input file
- `cluster_geometry()`: wrap geometry information obtained from input file
- `visualise_rgl()`: rgl visualisation of a cluster

extract_time *Extract timings from log files*

Description

Parses a log file to extract timing information from subroutines (verbosity-dependent)

Usage

```
extract_time(log = "log")
```

Arguments

log	filename

Value

returns a tibble of timings

Functions

- `extract_time()`: extract timings from log files

sample_fibonacci *sample_fibonacci*

Description

Fibonacci coverage of a sphere

Usage

```
sample_fibonacci(N = 301)
```

Arguments

N	number of points

Details

Produces a set of points that covers rather uniformly the unit sphere with N points with a spiral-like pattern based on a Fibonacci sequence

`store_xsec` *Reshape cross-section results into a convenient format for post-processing and plotting*

Description

Read and store plain text cross-sections

Extracts commonly-used information from a HDF5 file storing far-field cross-sections (Mode=2), and reshapes the data into long-format data.frames suitable for plotting

Extracts partial absorption cross-sections from a HDF5 file storing far-field cross-sections (Mode=2), and reshapes the data into long-format data.frames suitable for plotting

Usage

```
store_xsec(..., out = "xsec.rds")

consolidate_xsec(hdf5, verbose = TRUE, ...)

consolidate_partials(hdf5, verbose = TRUE)
```

Arguments

`...`	extra arguments passed to the final list
`out`	output Rds filename
`hdf5`	filename
`verbose`	logical: print attributes

Value

returns a list containing data.frames in long format

Functions

- `store_xsec()`: store plain text cross-sections
- `consolidate_xsec()`: consolidate cross-sections
- `consolidate_partials()`: consolidate partial absorption cross-sections for multilayered spheres

x3d_scene *Interactive display of cluster geometries*

Description

Displays a cluster in X3D format

Usage

```
x3d_scene(
  cl,
  viewpoint = c(0, 0, 100),
  orientation = NULL,
  width = "300px",
  height = "300px",
  scale = 100,
  ...
)
```

Arguments

cl	cluster
viewpoint	viewpoint position (3-vector)
orientation	optional viewpoint orientation (axis-angle vector: x,y,z,angle)
width	display width
height	display height
scale	size of axes
...	extra arguments passed to cluster_to_x3d

Value

returns X3D object to embed in a html document with suitable X3D support

Index