Package 'nanop'

September 25, 2015

Type Package

Title Tools for Nanoparticle Simulation and Calculation of PDF and Total Scattering Structure Function

Version 2.0-6

Author Anton Gagin, Katharine Mullen, Igor Levin

Maintainer Anton Gagin <av.gagin@gmail.com>

Imports distrEx, rgl

Suggests DEoptim, mco

Description This software package implements functions to simulate spherical, ellipsoid and cubic polyatomic nanoparticles with arbitrary crystal structures and to calculate the associated pair-distribution function and X-ray/neutron total-scattering signals. It also provides a target function that can be used for simultaneous fitting of small- and wide-angle total scattering data in real and reciprocal spaces. The target function can be generated either as a sum of weighted residuals for individual datasets or as a vector of residuals suitable for optimization using multi-criteria algorithms (e.g. Pareto methods).

URL http://scripts.iucr.org/cgi-bin/paper?S1600576714001046

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-09-25 13:43:04

R topics documented:

`broadPDF` *Analytically broaden the PDF*

Description

Analytically broaden the PDF using Gaussians

Usage

```
broadPDF(pdfob, sigma=0,  delta=NA, n=NA, nAtomTypes=1)
```

Arguments

`pdfob`	A list with elements `r` and `gr` that represent distances and values of the PDF, respectively. For core/shell particles elements `gr_CCSS` and `gr_CS` should also be specified. A list of this form is returned by function `calcPDF`.
`sigma`	numeric vector which, if not `NA`, determines the variance of the Gaussian displacements from the mean atomic position throughout the nanoparticle. If the particle core and shell contain `Nc` and `Ns` different atom types, respectively, then the first `Nc` elements in vector `sigma` correspond to atoms within the core and the next `Ns` elements describe Gaussian displacements for the shell atoms. See examples section in `simPart` for details.
`delta, n`	numerics describing the correlation parameters n and δ for thermal atomic displacements; see details.
`nAtomTypes`	number of different types of atoms in the particle.

Details

The correlated atomic displacement parameter for the atoms μ and ν is calculated as

$$\sigma^2_{\mu,\nu} = (\sigma^2_\mu + \sigma^2_\nu)[1 - \frac{\delta}{r^n}].$$

Value

A list with elements `r` and `gr` that represent distances and values of the PDF, respectively.

Note

This routine can be a faster way to account for thermal displacements than `displacePart`.

Examples

```
## simulate particle
Cu1 <- createAtom("Cu", sigma=0.012)
Cu2 <- createAtom("Cu", sigma=0.008)
part <- simPart(atoms=list(Cu1), atomsShell=list(Cu2), r=20,
    rcore=16, latticep=4.08, latticepShell=3.89)

## use a stochastic model for displacements
partx <- displacePart(part, sigma=attributes(part)$sigma)
gr1 <- calcPDF(partx, maxR=40)

## use analytical broadening
gr2 <- calcPDF(part, maxR=40)
gr2 <- broadPDF(gr2, sigma=attributes(part)$sigma, nAtomTypes=2)

# plot PDFs calculated using both methods
matplot(gr1$r, cbind(gr1$gr, gr2$gr), type="l", lty=1, lwd=1:2)
```

`calcPDF` *Functions to calculate the PDF*

Description

Functions to calculate the pair distribution function (PDF) and Q-dependent PDF given a matrix of atomic positions. The latter is relatively time-consuming.

Usage

```
calcPDF(nanop, dr=.01, minR=.01, maxR=20,
        scatterLength=NA, scatterFactor=NA,
        type="neutron", Qmin=1e-16)
calcQDepPDF(nanop=NA, dr=.1, minR=1, maxR=20, dQ=.01, minQ=1, maxQ=20,
            verbose=0, subdivisions = 100, order=1000,
            rel.tol=.Machine$double.eps^.7,
            addNoise=FALSE, noiseFun=NA,
            totalScattParams=list(), preTotalScat=NA, ...)
```

Arguments

`nanop`	numeric matrix in which each row gives the coordinates of an atomic position in the nanoparticle. For `calcPDF` calculations if nanop is not an object returned by `simPart` or `displacePart` attributes `nAtomTypes`, `atomType`, `layer_start`, `layer_end`, `layerS_start`, `layerS_end`, `rowcore`, `scatterLength`, and `scatterFactor` should be set manually; see `simPart`.
`dr`	numeric indicating the desired step size in r.
`minR`	numeric indicating the minimum value of r for which function should be evaluated.

maxR	numeric indicating the maximum value of r for which function should be evaluated.
scatterLength	numeric vector describing neutron scattering lengths for all atom types in the particles. If NA the value is sought in nanop attributes.
scatterFactor	list containing X-ray scattering factor parameters, see calcTotalScatt. If NA the value is sought in nanop attributes.
type	character; type of scattering. Either "X-ray" or "neutron".
Qmin	numeric used to approximate Q-dependent X-ray scattering factor via its value at point Q=Qmin.
dQ	numeric indicating the desired step size in Q in total scattering function calculations.
minQ	numeric indicating the minimum value of Q for which the total scattering function should be evaluated.
maxQ	numeric indicating the maximum value of Q for which the total scattering function should be evaluated.
verbose	numeric; if greater than zero a status report is given after computing the function every verbose steps in r.
subdivisions	numeric, the maximum number of subintervals for the Fourier integral calculation.
order	numeric, order of Gauss-Legendre quadrature.
rel.tol	numeric, relative accuracy requested for the Fourier integral calculation.
addNoise	logical indicating whether noise should be added to the total scattering structure function. If addNoise=TRUE then function noiseFun is applied to the approximation of the total scattering structure function at points Q determined by the numerical integration routine.
noiseFun	function used to add noise to the total scattering structure function, applied only if addNoise=TRUE. The first argument should be a numeric vector representing the total scattering structure function. Additional arguments may be passed via ....
totalScattParams	list contacting objects sigma, n, delta, kind, dr, del, eps, type, scatterLength, and scatterFactor that are passed to calcTotalScatt function for total scattering function calculation. See calcTotalScatt for details. Should be specified if preTotalScat=NA.
preTotalScat	list with elements Q (grid points) and gQ (values of the total scattering structure function at the corresponding gridpoints). If NA calcTotalScatt parameter should be specified.
...	additional arguments for noiseFun.

Details

If preTotalScat is not NA calcQDepPDF function calculates Fourier transform of the vector gQ$preTotalScat. Otherwise it uses parameters specified in totalScattParams to generate total scattering function first.

Value

list with elements

r	numeric vector of values at which the function was evaluated,
gr	numeric vector of function values.

Note

To normalize calcPDF in the same way as calcQDepPDF it should be multiplied by $4\pi r$.

calcQDepPDF is currently rather time-consuming.

See Also

simPart, displacePart

Examples

```
## simulate a particle
Cu <- createAtom("Cu")
part <- simPart(atoms=list(Cu), atomsShell=list(Cu), r=8,
    rcore=6, latticep=4.08, latticepShell=3.89)
## uniform displacement of positions
partx1 <- displacePart(part, sigma=c(.005, 0.005))
## different displacement in core than shell
partx2 <-  displacePart(part, sigma=c(.005, .02))

## calculate and plot PDF associated with both particles
gr1 <- calcPDF(partx1, maxR=24, scatterLength=c(4.87, 7.97), dr=.02)
gr2 <- calcPDF(partx2, maxR=24, scatterLength=c(4.87, 7.97), dr=.02)

plot(gr1$r, gr1$gr, type="l")
lines(gr2$r, gr2$gr, col=2)

## calculate scattering fucntion
gQ <- calcTotalScatt(part, type="neutron", dr=.02,
    scatterLength=c(4.87, 7.97), sigma=c(.005, .02))
t1 <- which(gQ$Q > 30)[1]
t2 <- which(gQ$Q > 34.9)[1]
cut <- gQ$Q[t1:t2][which(abs(gQ$gQ[t1:t2])
    ==min(abs(gQ$gQ[t1:t2]) ))[1]]
## calculate Q-dependent PDF
gr3 <- calcQDepPDF(part, minR=0, maxR=24, dr=0.02, minQ=.771, maxQ=cut,
    verbose=100, preTotalScat=list(Q=gQ$Q, gQ=gQ$gQ) )
## compare results with that obtained by calcPDF:
## ...normalization:
gr2 <- 4*pi*gr2$r*gr2$gr
## calculate and subtract gamma term:
gQSAS <- calcTotalScatt(part, type="neutron",  minQ=0.001,
    maxQ=0.771, dQ=0.005,  dr=.02,
    scatterLength=c(4.87, 7.97), sigma=c(.005, .02))
gammaR <- calcQDepPDF(part, minR=0.01, maxR=24,
```

```
    maxQ=.771, minQ=0.001, dr=.02,
    verbose=100, preTotalScat=list(Q=gQSAS$Q, gQ=gQSAS$gQ))
gr2 <- gr2 - gammaR$gr

## plot pair distribution functions associated with both technique:
plot(gr1$r, gr2, type="l")
lines(gr3$r, gr3$gr, col=2, lwd=2)
```

`calcTotalScatt` *Functions to calculate the total scattering structure function*

Description

Functions to calculate the total scattering structure function given a matrix in which each row represents the position of an atom or to simulate it in the SAS region using particle lattice and size parameters.

Usage

```
calcTotalScatt(nanop, dQ=.01, minQ=0.771, maxQ=35, type="neutron",
    scatterFactor=NA, scatterLength=NA, sigma=NA, n=0,
    delta=0, kind="fastHist", dr = 0.001,  del = 0.01, eps=1e-3)
IqSAS(Q, Rcore=NA, Rpart, latticep, latticepShell=NA,
    scatterLength, N1, N2=NA, pDimer=0, sym, symShell=NA)
IqSASP(Q, shell=NA, Rpart, latticep, latticepShell=NA,
    scatterLength, N1, N2=NA, pDimer=0, sym, symShell=NA, rsigma)
```

Arguments

`nanop`	numeric matrix in which each row gives the coordinates of an atomic position in the nanoparticle. If nanop is not an object returned by `simPart` or `displacePart` then attributes `dimer`, `nAtomTypes`, `atomType`, `layer_start`, `layer_end`, `layerS_start`, `layerS_end`, `r` and `rcore` should be set manually; see `simPart`.
`dQ`	numeric indicating the desired step size in Q.
`minQ`	numeric indicating the minimum value of Q for which the function should be evaluated.
`maxQ`	numeric indicating the maximum value of Q for which the function should be evaluated.
`type`	character indicating type of scattering. Either "X-ray" or "neutron".
`scatterFactor`	list containing X-ray scattering factor parameters; see details. If `NA` the value is sought in `nanop` attributes.
`scatterLength`	in `calcTotalScatt` function call, numeric vector describing neutron scattering lengths for all atom types in the particles. If `NA` the value is sought in `nanop` attributes. In `IqSAS` and IqSASP calls, numeric vector describing average neutron scattering length for the particle core and shell; see examples.

sigma	numeric vector which, if not NA, determines the variances of the Gaussian displacements from the mean atomic positions throughout the nanoparticle. If NA the value is sought in nanop attributes. See simPart for details.
n, delta	numerics describing the correlation parameters n and δ for thermal atomic displacements; see details.
kind	character, can be set to "exact", "fast", "fast_av" or "fastHist". If "exact" the Debye sum is calculated as is, that can be relatively time-consuming. If "fast", "fast_av" or "fastHist" then the pseudogrid approach (Cervellino et al., 2006) is used. If "fastHist" the histogram bin approach is used to calculate interatomic distances and their multiplicities. Option "fast_av" should be used if nanop attribute "r" length greater than 1, i.e. for polydisperse particles . Nonzero dr value in that case switches computational scheme to histogram bin approach. In most case "fastHist" is recommended for better performance.
dr	numeric. If kind="fastHist" or "fast_av" describes histogram bin width.
del	numeric. If kind="fast" or "fast_av" describes the pseudolattice parameter Δ.
eps	numeric. If kind="fast" or "fast_av" describes the step size ϵ for interatomic distances.
Q	numeric vector, contains grid points on which the total scattering structure function should be evaluated.
N1, N2	numerics indicating number of atoms within the unit cell in the particle core and shell.
Rcore	numeric which, if not NA, determines the radius of the core.
Rpart	numeric indicating the radius of the nanoparticle
latticep, latticepShell	numeric vectors indicating the lattice parameter(s) for the core (shell); see simPart.
pDimer	numeric indicating probability of atom to form a cluster with its neighbour.
sym, symShell	characters describing the structure to be used; see simPart.
shell	numeric indicating shell thickness for the core/shell particle. For polydisperse particles shell is supposed to be of constant thickness.
rsigma	numeric indicating standard deviation in the log normal distribution of the particle (core) radius.

Details

The X-ray scattering factor is approximated by the function

$$f(s) = a_1 * exp(-b_1 s) + a_2 * exp(-b_2 s) + a_3 * exp(-b_3 s) + a_4 * exp(-b_4 s) + c$$

where $s = (\frac{Q}{4\pi})^2$. The constants in the function are possible to specify as arguments. In that case scatterFactor parameter should contain vectors a1, b1, a2, b2, a3, b3, a4, b4, a5, b5 and c. Their default values correspond to the values for Au and Pd atoms.

The correlated atomic displacement parameter for the atoms μ and ν is calculated as

$$\sigma^2_{\mu,\nu} = (\sigma^2_\mu + \sigma^2_\nu)[1 - \frac{\delta}{r^n}].$$

Value

`calcTotalScatt`: list with elements

`Q`	numeric vector of values at which the function was evaluated,
`gQ`	numeric vector of function values.

`IqSAS` and `IqSASP`: numeric vector of function values.

Note

IqSASP function calculates total scattering function for the polydisperse uniform particles and polydisperse core/shell particles with constant shell thickness.

References

Cervellino A, Giannini C, Guagliardi A. (2006): On the efficient evaluation of Fourier patterns for nanoparticles and clusters. *J. Comput. Chem.* **27**, 995–1008.

See Also

`simPart`, `displacePart`

Examples

```
## simulate particle
Au <- createAtom("Cu")
Au$name <- "Au"
Pd <- createAtom("Cu")
Pd$name <- "Pd"

part <- simPart(list(Au), atomsShell=list(Pd), r=10, rcore=8)
gQ <- calcTotalScatt(part, type="neutron", sigma=c(.08, .012),
    kind="fast", del=5e-4)
plot(gQ$Q, gQ$gQ, type="l")

## "fast_av" option to calculate scattering function
## for the array of polydisperse particles:
Zn <- createAtom("Zn")
S <- createAtom("S")
part <- simPart(atoms=list(Zn, S), r=c(8, 10, 11.2, 13.4, 14),
    sym="hcp", latticep=c(4.3, 7.02))
gQ <- calcTotalScatt(part, type="neutron", sigma=c(.08, .012),
    kind="fast_av")
plot(gQ$Q, gQ$gQ, type="l")

## total scattering function in small-angle region using
##  parametric model:
Q=seq(0.001, 0.771, 0.005)
gQSAS <- IqSAS(Q=Q, Rpart=26, latticep=c(3.21, 5.21),
    scatterLength=7.1, N1=2, sym="hcp")
plot(Q, log(abs(gQSAS)), type="l")
```

```
## total scattering function for polydisperse sample with
## lognormal distribution
gQSAS2 <- IqSASP(Q=Q, shell=2.8, Rpart=11.2, latticep=c(4.08),
    latticepShell=3.89, N1=4, N2=4, scatterLength=c(7.1, 8.3),
    sym="fcc", symShell="fcc", rsigma=1.1)
plot(Q, log(abs(gQSAS2)), type="l")
```

---

createAtom *Describe fractional coordinates and scattering parameters of the basis atoms*

Description

Function to create object of type atoms to be passed to simPart.

Usage

```
createAtom(name, base=NA, sigma=0.01, scatterLength=NA,
    scatterFactor=NA)
```

Arguments

name	character describing the name of chemical element; see details.
base	numeric matrix in which each row represents fractional coordinates of the corresponding basis atom within the primitive cell; see examples. If NA getBase is called with name argument.
sigma	numeric vector which, if not NA, determines the variance of the Gaussian displacements from the mean atomic positions for given atom.
scatterLength	numeric describing neutron scattering lengths for given atom. If NA getScatterLength is called with name argument.
scatterFactor	list containig X-ray scattering factor parameters. If NA getScatterFactor is called with name argument.

Details

Parameter name can be set up to any character value; if other arguments are not specified it is used as an argument in functions getBase, getScatterLength and getScatterFactor calls. Functions getScatterLength and getScatterFactor contatain information for most chemical elements and some of their isotopes and ions. Fractional coordinates, however, does depend not on the type of chemical element but on the structure under considirations. Therefore argument name in function getBase indicates crystal structure (see getBase). For example passing character "Cl" as an argument results in fractional coordinates of Cl atoms in the rock salt crystal structure.

Value

list with elements name, base, sigma, scatterLength, scatterFactor.

See Also

getBase, getScatterLength, getScatterFactor

Examples

```
## create CdSe nanoparticle:
## CdSe particle has wurtzite structure
## Zn and S atoms could be used to describe
## basis atoms fractional coordinates
## (see getBase() )

## get fractional coordinates for Zn
Cd_base <- getBase("Zn")
## get scattering parameters for Cd
Cd_scL <- getScatterLength("Cd")
Cd_scF <- getScatterFactor("Cd")
Cd <- createAtom("Cd", base=Cd_base, scatterLength=Cd_scL,
    scatterFactor=Cd_scF, sigma=0.01)

Se_base <- getBase("S")
Se_scL <- getScatterLength("Se")
Se_scF <- getScatterFactor("Se")

Se <- createAtom("Se", base=Se_base, scatterLength=Se_scL,
    scatterFactor=Se_scF, sigma=0.008)

## atoms Cd and Se now can be used as arguments in simPart
part <- simPart(atoms=list(Cd, Se), latticep=c(4.3, 7.02),
    sym="hcp")
## uncomment to plot particle:
#plotPart(part)

## create rock salt structure
Na <- createAtom("Na")
Cl <- createAtom("Cl")
## name, scatterLength and scatterFactor parameters
## in Na and Cl are now set up to those of Na and Cl.
## if we are interested only in particle structure
## or actually simulating rock salt there is no need
## to change them
part <- simPart(atoms=list(Na, Cl))
#plotPart(part)

## set up fractional coordinates that cannot be
## simulated by getBase()
## (for example we have ZnS wurtzite structure
## with atom S z-coordinate 0.378
## different from ideal value of 0.375 (3/8) ):
```

```
S_base <- matrix(c(
      c(0.5,  sqrt(3)/6, 0.378),  #S
  c(0.5,  -sqrt(3)/6,  (0.378-0.5))),
      ncol=3, byrow=TRUE)
## each row represents fractional atomic coordinates in the primitive cell
S2 <- createAtom("S", base=S_base)
## use standard parameters for Zn atoms
Zn <- createAtom("Zn")
part <- simPart(atoms=list(Zn, S2), latticep=c(4.3,7.02), sym="hcp")
#plotPart(part)
```

---

gaussConvol *Gauss convolution*

Description

Function to calculate the convolution of a given function with the Gaussian function

Usage

```
gaussConvol(SQ, Q, Qdamp=0.0457, err=1e-6)
```

Arguments

SQ	numeric vector containing function to be convoluted.
Q	numeric vector containing grid points corresponding to SQ.
Qdamp	numeric indicating standard deviation parameter for the Gaussian function.
err	numeric, relative accuracy requested.

Value

numeric vector of function values.

See Also

calcTotalScatt

Examples

```
## simulate a particle
Cu <- createAtom("Cu")
part <- simPart(atoms=list(Cu), atomsShell=list(Cu), r=20,
    rcore=14, latticep=4.08, latticepShell=3.89)

## calculate total scattering function
```

```
gQ <- calcTotalScatt(part, type="neutron",  minQ=0.771, maxQ=18,
    dQ=0.01, scatterLength=c(4.87, 7.97), sigma=c(.01, .01))
plot(gQ$Q, gQ$gQ, type="l")

## simulate instrumental resolution effect
Q <- gQ$Q
gQ <- gaussConvol(SQ=gQ$gQ, Q=gQ$Q, Qdamp=0.061, err=1e-5)
lines(Q, gQ, col=2)
```

`getBase` *Fractional coordinates and scattering parameters for a given chemical element*

Description

Get fractional coordinates, neutron scattering length and X-ray scattering factor.

Usage

```
getBase(name)
getScatterLength(name)
getScatterFactor(name)
scatterFactor(scatterFactor, Q)
```

Arguments

`name`	name of chemical element; see details..
`scatterFactor`	list containing scattering factor parameters as returned by `getScatterFactor`..
`Q`	numeric vector containing grid points at which scattering factor should be calculated..

Details

`getScatterLength` contains data for neutron scattering lengths of the elements and their isotopes taken from http://www.ncnr.nist.gov/resources/n-lengths/list.html. Parameter `name` should be given as in the corresponding table.

`getScatterFactor` contains data for X-ray scattering factors of the elements and their isotopes taken from Waasmaier et al., 1995. Parameter `name` should be given as in table 1. The atomic scattering factor is calculated using the method developed by Waasmaier et al. that implies approximation by a function

$$f(s) = a_1 * exp(-b_1 s) + a_2 * exp(-b_2 s) + a_3 * exp(-b_3 s) + a_4 * exp(-b_4 s) + c$$

, with $s = (\frac{Q}{4\pi})^2$.

`getBase()` function contains information about fractional coordinates for certain simple structure. Parameter `name` can be:

"Cu" to specify fractional coordinates of atoms in monoatomic fcc lattice,

"Fe" to specify fractional coordinates of atoms in monoatomic bcc lattice,

"Na" to specify fractional coordinates of Na atoms in NaCl rock salt structure,

"Cl" to specify fractional coordinates of Cl atoms in NaCl rock salt structure,

"Ca" to specify fractional coordinates of Ca atoms in CaTiO3 ideal perovskite structure,

"Ti" to specify fractional coordinates of Ti atoms in CaTiO3 ideal perovskite structure,

"O3" to specify fractional coordinates of O atoms in CaTiO3 ideal perovskite structure,

"Mg" to specify fractional coordinates of atoms in monoatomic hcp lattice,

"Zn" to specify fractional coordinates of Zn atoms in ZnS wurtzite structure,

"S" to specify fractional coordinates of S atoms in ZnS wurtzite structure.

Value

`getBase`: numeric matrix in which each row represents fractional coordinates of the corresponding atom within the primitive cell.

`getScatterLength`: numeric describing neutron scattering length for the given element.

`getScatterFactor`: list containing X-ray scattering factor parameters for the given element.

`scatterFactor`: numeric vector containing scattering factor value(s).

References

Waasmaier D. and Kirfel A. (1995): New analytical scattering-factor functions for free atoms and ions. *Acta Cryst.* A**51**, 416–431.

See Also

`createAtom`

Examples

```
## get fractional coordinates for Zn
Cd_base <- getBase("Zn")
## get scattering parameters for Cd
Cd_scL <- getScatterLength("Cd")
Cd_scF <- getScatterFactor("Cd")

Se_base <- getBase("S")
Se_scL <- getScatterLength("Se")
Se_scF <- getScatterFactor("Se")

Q <- seq(0.1, 15, 0.1)
## plot scattering factor
plot(Q, scatterFactor(Se_scF, Q))
```

GrSAS *Functions to calculate the gamma baseline term for PDF*

Description

Functions to calculate analytically the gamma baseline term given a particle lattice and size parameters.

Usage

```
GrSAS(r, Rcore=NA, Rpart, latticep, latticepShell=NA,
    N1, N2=NA, sym, symShell=NA)
GrSASCS(r, Rcore=NA, Rpart, latticep, latticepShell=NA,
    N1, N2=NA, sym, symShell=NA)
```

Arguments

`r`	numeric vector that contains grid points at which baseline term should be evaluated.
`Rcore`	numeric which, if not `NA`, determines the radius (radii) of the core.
`Rpart`	numeric indicating the radius (radii) of the particle.
`latticep, latticepShell`	numeric vectors indicating the lattice parameter(s) for the core(shell); see `simPart` for details.
`N1, N2`	numeric indicating number of atoms within the unit cell in the particle core (shell).
`sym, symShell`	characters describing the structure to be used in the particle core (shell) simulations; see `simPart`.

Details

Function `GrSAS` can be used for both uniform and core/shell particles. In the second case the uniform model is applied with scattering length density averaged through the nanoparticle. Function `GrSASCS` calculates baseline term for core/shell particles using model described in Glatter, 1979.

Value

numeric vector of function values.

References

Glatter O. (1979): The interpretation of real-space information from small-angle scattering experiments. *J. Appl. Cryst.*, **12**, 166–175.

See Also

calcPDF, calcQDepPDF

Examples

```
## calculate baseline term for uniform particle
gammaR <- GrSAS(r=seq(0.01, 30, 0.01), Rpart=15,
    latticep=c(4.3, 7.02), sym="hcp", N1=4)
plot(seq(0.01, 30, 0.01), gammaR, type="l")

## compare with baseline computed as Fourier transform
## of the total scattering function:
Zn <- createAtom("Zn")
S <- createAtom("S")
part <- simPart(atoms=list(Zn,S), r=15, latticep=c(4.3, 7.02),
    sym="hcp")
gQSAS <- calcTotalScatt(part, type="neutron",  minQ=0.001,
    maxQ=0.9, dQ=0.005)
gammaR2 <- calcQDepPDF(part, minR=0.01, maxR=30, dr=0.01,
    maxQ=.85, minQ=0.001, verbose=20,
    preTotalScat=list(Q=gQSAS$Q, gQ=gQSAS$gQ))
lines(gammaR2$r, gammaR2$gr, col=2)
```

---

plotPart *Draws a three-dimensional scatterplot*

Description

Function to visualize a nanoparticle using rgl package.

Usage

```
plotPart(nanop, radius=0.4, legend=TRUE, col=NA, box=FALSE,
    play=FALSE, atoms=NA, miller=NA, lattice=c(4.08))
```

Arguments

nanop	numeric matrix in which each row gives the coordinates of an atomic position in the nanoparticle. If nanop is not an object returned by simPart or displacePart attributes nAtomTypes, atomType, r, sym, and symShell must be set manually; see simPart.
radius	numeric vector or single value. Each atom on a plot is represented by a sphere. radius defines the sphere radius (radii).
legend	logical indicating whether to print plot legend.
col	numeric vector defining colours to be used for plotted items. If vector col length does not correspond to number of atom types within the particle then standard colouring scheme is used.

box	logical indicating whether to draw box and axes.
play	logical. If TRUE animation with constantly rotating particle is played.
atoms	character. If not NA specifies atoms to be displayed, see details.
miller	numeric vector, specifies Miller indices. If not NA only the plane in a particle core described by given indices is displayed. Should be given in a form c(h, k, l) for the non-hexagonal symmetry and c(h, k, i, l) for the hexagonal symmetry. Should be specified together with lattice parameter.
lattice	numeric vector indicating particle core lattice parameters. Should be given in the same form as in simPart.

Details

If only core (shell) atoms of a specific type to be plotted atoms should be set up to "core X" or "shell X", respectively. Character describing atom type "X" can be taken from attributes(part)$atomsCore or attributes(part)$atomsShell.

Value

a vector of object IDs.

Examples

```
## rgl library demands graphical terminal to be available
## uncoment all plotPart() calls for 3D visualization
## simulate particle
Au <- createAtom("Cu")
Au$name <- "Au"
Pd <- createAtom("Cu")
Pd$name <- "Pd"
part <- simPart(atoms=list(Au), atomsShell=list(Pd), rcore=8)
## 3d scatter plot
#plotPart(part, col=c(2,4))

## increase number of atom types within the particle:
Zn <- createAtom("Zn")
S <- createAtom("S")
part <- simPart(atoms=list(Zn ,S), atomsShell=list(Au), r=14,
    rcore=12, sym="hcp", symShell="fcc", latticep=c(4.3, 7.04),
    latticepShell=4.08)
## 3d scatter plot
#plotPart(part, col=c(2,4,3))
## play animation:
#plotPart(part, col=c(2,4,3), play=TRUE)

## plot only shell particles
#plotPart(part, col=c(2,4,3), atoms="shell Au", play=TRUE)

part <- simPart(atoms=list(Zn ,S),r=20, sym="hcp",
    latticep=c(4.3, 7.04))
## display plane normal to z-axis:
```

```
#plotPart(part, miller=c(0, 0, 0 ,1), lattice=c(4.3, 7.04))
##S atoms:
#plotPart(part, miller=c(0, 0, 0 ,1), lattice=c(4.3, 7.04),
#    atoms = "core S")
## save picture in a file using rgl function:
#rgl.snapshot( filename = "plane0001 S atoms.png")

Na <- createAtom("Na")
Cl <- createAtom("Cl")
part <- simPart(atoms=list(Na,Cl), sym="fcc")
#plotPart(part, miller=c(1,0,1), box=TRUE, lattice=c(4.08))
## plot only Na atoms:
#plotPart(part, miller=c(1,0,1), box=TRUE, lattice=c(4.08),
#    atoms = "core Na")
```

`rss` *Function to calculate the RSS between the simulated PDF or the total scattering structure function and target data*

Description

Given a nanoparticle model and PDF or total scattering structure function data, this function calculates the PDF or total scattering structure function associated with the model. It then determines the residual sum of squares (RSS) between the simulated signals and the data.

Usage

```
rss(par, dataG=NA, dataS=NA, dataSAS=NA, part=NA, type="neutron",
    simPar=NA, PDF.fixed=list(), TotalScatt.fixed=list(),
    parscale=NA, skel=NA, con=TRUE, oneDW=FALSE,
    punish=FALSE, gammaR=NA, rvector=NA, fixed=NA,
    wG=0.03, wSAS=0.03, avRes=1, pareto=FALSE)
```

Arguments

`par`	numeric vector of parameter values. These should be given along with the appropriate argument `skel`. See details and examples.
`dataG`	numeric vector, which, if not `NA`, determines the PDF data.
`dataS`	numeric vector, which, if not `NA`, determines the total scattering structure function data.
`dataSAS`	numeric vector, which, if not `NA`, determines the total scattering structure function data in SAS region.
`part`	either `NA` or the atomic positions in a nanoparticle given as a numeric matrix in which each row gives the (3D) position coordinates. A matrix of this form is the return value of the function `simPart`. Default value of `NA` means that the particle is to be re-simulated in each call to `rss` (necessary for stochastic model functions).

`type` character indicating type of scattering. Either "X-ray" or "neutron".

`simPar` arguments are passed to the function `simPart` to simulate the nanoparticles if these values are not to be optimized; any of the following arguments may be specified: `atoms`, `atomsShell`, `sym`,`symShell`, `latticep`, `latticepShell`, `r`, `rcore`, `box`, `ellipse`, `shell`, `rcenter`, `move`, `rotShell`, `rcenterShell` and `distr`. The latter should be set up to "normal" or "lognormal" to simulate polydisperse particles with normal or log-normal distribution, respectively. See `simPart` for the other parameters notation details.

`PDF.fixed` arguments are passed to the functions `calcPDF` and `broadPDF`. Any of the following arguments may be specified: `dr`, `minR`, `maxR`, `scatterLength`, `scatterFactor`, `delta`, `n`, `Qmax`, `termRip`, `maxRTermRip`, `N1`, `N2`, and `Qdamp`; see `calcPDF` and `GrSAS`. If `termRip` is `TRUE` termination ripples are simulated with parameters `dr`, `Qmax` and `maxRTermRip` (see `termRip`). If `Qdamp` is not `NA` the PDF is multiplied by the function $exp(-Q_{damp}^2 r^2/2)$.

`TotalScatt.fixed` arguments are passed to the functions `calcTotalScatt`; any of the following arguments may be specified: `dQ`, `minQ`, `maxQ`, `minQ_SAS`, `maxQ_SAS`, `dQ_SAS`, `scatterLength`, `scatterFactor`, `delta`, `n`, `dr`, `del`, `eps`, `kind`, `N1`, `N2`, `f2`, `f2`, `convolution`, `Qdamp`, `SASscale`, and `paramSASQ`; see `calcTotalScatt`. If `convolution` is `TRUE` function `gaussConvol` is called with `Qdamp` parameter. If `paramSASQ` is `TRUE` the total scattering function in SAS region is calculated using functions `IqSAS` or `IqSASP` (with `f1` and `f2` parameters indicating average scattering lengths in the particle core and shell). Parameter `SASscale` sets weighting scheme for total scattering function in SAS region residual. Should be "normal", "log" (for logarithmic scale) or "weighted" (to normalize the residual at each grid point with the function value).

`parscale` either `NA` or a numeric vector of the same length as `par` indicating values by which the values in `par` should be divided for the purpose of parameter scaling.

`skel` an object of class `relistable`. First the parameters to be optimized should be written as a named list of form `parameters<-list(a=1,b=2,c=3)`. Then the argument `par` can be given as `par=unlist(parameters)` and `skel` can be given as `skel=as.relistable(parameters)`.

`con` logical indicating whether to reset `r` to be equal to `rcore` if `rcore > r`.

`oneDW` logical indicating whether to use similar `sigma` parameter for all atom types within the particle.

`punish` if the inequalities described above for `con` are violated, return a large RSS value (`10e15`).

`gammaR` either "param" or a numeric vector of gamma baseline term data. If "param" the baseline is simulated with functions `GrSAS` and `GrSASCS`.

`rvector` numeric vector of grid points at which the PDF was evaluated.

`fixed` list; if not `NA` indicates parameter names.

`wG, wSAS` weights for the PDF and total scattering function in SAS region residuals.

`avRes` numeric value; if the model for the data is stochastic, he number of particles to simulate; the model PDF or total scattering structure function is calculated for each particle and then averaged.

pareto — logical; if TRUE the result value is a vector of residuals for the given data sets that can be used for the subsequent Pareto analysis.

Details

The following parameters may be fitted in rss (i.e., specified in par or skel): r, rsigma (sets standard deviation in the normal or log-normal distribution with mean value r), box, ellipse, bkg (background line that should be extracted from the SAS data), shell, rcore, scaleSq (scale factor for the total scattering function or the PDF), scaleSASq (scale factor for the total scattering function in SAS region or the gamma baseline term), pStack, pDimer, delta, sigma, latticep and latticepShell. For wurtzite structure to fit u parameter (second atom z-coordinate) third element in the vectors latticep and latticepShell may be specified.

Value

If pareto=FALSE the R-factor: the square root of the sum of squared distances between the target data and simulated signal divided by a the sum of squared target data values. If pareto=TRUE a vector with R-factors as components.

Note

Functions

```
simPartPar(sym = "fcc", symShell=NA, latticep = 4.08, r=10,
    atoms=list(), atomsShell=list(), distr = "lognormal",
    rcenter=FALSE, latticepShell=NA, rcore=NA, shell=NA, move=TRUE,
    box=NA, ellipse=NA, rotShell=FALSE, rcenterShell=FALSE),
PDFPar(dr=.01, minR=1, maxR=20, termRip=FALSE, Qmax=30, maxRTermRip=20,
    scatterLength=NA, n=2, delta=NA, Qdamp=NA, scatterFactor=NA, N1=4, N2=4),
```

and

```
TotalScattPar(dQ=.01, minQ=0.771, maxQ=35, minQ_SAS=0.001, maxQ_SAS=0.771,
    dQ_SAS=0.005, scatterLength=NA, scatterFactor=NA, dr = 0.001,
    del = 0.01, eps=1e-3,  kind="fastHist",  N1=4, N2=4, f1=5, f2=5,
   SASscale="normal", convolution = FALSE, Qdamp=0.0457, delta=NA, n=2, paramSASQ=FALSE)
```

return values suitable for simPar, PDF.fixed and TotalScatt.fixed parameters, respectively.

Examples

```
part <- res_gQ <- res_gQ_WAS <- res_gQ_SAS <- list()

## prepare polydisperse sample consisting of 30 particles:
size <- sort(rlnorm(20, meanlog = log(10), sdlog = log(1.1)))
base_Cu <- getBase("Cu")
Au <- createAtom("Au", base=base_Cu)

for(i in 1:20) {
```

```
  cat("r = ", size[i], "\n")
  part[[i]] <- simPart(atoms=list(Au), r=size[i], latticep=4.08,
      rcenter=TRUE)
  res_gQ_WAS[[i]] <- calcTotalScatt(part[[i]],minQ=.771, maxQ=20,
      dr=0.02, dQ=0.02, sigma=c(0.02))
  res_gQ_SAS[[i]] <- calcTotalScatt(part[[i]],minQ=.001,
      dr=0.02, maxQ=.771, dQ=0.01, sigma=c(0.02))
  cat(i,"\n")
}

## calculate average values:
gQ_av_WAS <- 0
for(i in 1:length(res_gQ_WAS)) {
  gQ_av_WAS <- res_gQ_WAS[[i]]$gQ + gQ_av_WAS
}
gQ_av_WAS <- gQ_av_WAS/length(res_gQ_WAS)

gQ_av_SAS <- 0
for(i in 1:length(res_gQ_SAS)) {
  gQ_av_SAS <- res_gQ_SAS[[i]]$gQ + gQ_av_SAS
}
gQ_av_SAS <- gQ_av_SAS/length(res_gQ_SAS)

## calculate PDF and gamma baseline term
resSAS  <- calcQDepPDF(minR=0, maxR=30, dr=0.02, minQ=.001,
    maxQ=.771, verbose=100,
    preTotalScat=list(Q=res_gQ_SAS[[1]]$Q,gQ=gQ_av_SAS))

mr <- which(res_gQ_WAS[[1]]$Q > 17)[1]
mxr <- which(res_gQ_WAS[[1]]$Q > 19)[1]
cuto <- res_gQ_WAS[[1]]$Q[mr:mxr][which(abs(gQ_av_WAS[mr:mxr])
    ==min(abs(gQ_av_WAS[mr:mxr]) ))[1] ]
resWAS <- calcQDepPDF(minR=0, maxR=30, dr=0.02, minQ=.771,
    maxQ=cuto, verbose=100,
    preTotalScat=list(Q=res_gQ_WAS[[1]]$Q,gQ=gQ_av_WAS))

## set boundaries for fitting procedure
iter_0 <- as.relistable(list(latticep=0, r=0, sigma=c(0), rsigma=0))
boundsL <- c(latticep=3.5,  r=10.0, sigma=c(.01), rsigma=1.1)
boundsU <- c(latticep=4.5,  r=14.0,  sigma=c(.03), rsigma=1.3)

## in order to estimate the parameters that were used to
## simulate the particles, the DEoptim package may be
## used. Install it, remove the comment symbols '#' below,
## and use a call like:
#library(DEoptim)

#resDE <- DEoptim(rss, lower = boundsL, upper = boundsU,
#    oneDW=FALSE, type="neutron",
#    control=DEoptim.control(CR=0.85, F=0.7, NP=40, itermax=30,
#        parallelType=1, packages = list("PerformanceAnalytics"),
#        parVar=list("rss", "simPart", "calcPDF",
#        "IqSASP", "GrSAS", "GrSASCS", "broadPDF", "termRip",
```

```
#        "getSymEl", "IqSAS", "displacePart", "calcTotalScatt")),
#    dataS=NA, dataSAS=gQ_av_SAS, dataG=resWAS$gr,
#    simPar = simPartPar(atoms=list(Au), rcenter=TRUE,
#        move=TRUE, rot=FALSE),
#    gammaR = resSAS$gr, rvector = resSAS$r, skel=iter_0,
#    PDF.fixed=PDFPar(minR=0, maxR=30, dr=.01,
#        scatterLength=Au$scatterLength),
#    TotalScatt.fixed=TotalScattPar(minQ=0.771, maxQ=20,
#        dQ=0.02, dQ_SAS=0.01, minQ_SAS=.001, maxQ_SAS=.771,
#        scatterLength=Au$scatterLength, kind="fastHist",
#        SASscale="normal", convolution = FALSE),
#    wG=1.0, wSAS=0.05, avRes=10, pareto=FALSE)
#
## now resDE$optim contains estimates for the lattice parameter,
## particle radius, displacement variance sigma, and standard
## deviation rsigma.
## show results:
#resDE$optim$bestmem

## package mco may be used to construct a Pareto front of
## solutions. Note that calculations may take considerable time!

#library(mco)
#resMCO <- nsga2(rss, 4, 2,
#    dataS=NA, dataSAS=gQ_av_SAS, dataG=resWAS$gr,  oneDW=FALSE,
#    simPar = simPartPar(rcenter=TRUE, move=TRUE, rot=FALSE),
#    gammaR = resSAS$gr, rvector = resSAS$r, skel=iter_0,
#    PDF.fixed=PDFPar(minR=0, maxR=30, dr=.01,
#        scatterLength=scatterLength),
#    TotalScatt.fixed=TotalScattPar(minQ=0.771, maxQ=20, dQ_SAS=0.01,
#        dQ=.02, minQ_SAS=.001, maxQ_SAS=.771, scatterLength=scatterLength,
#        kind="fastHist", SASscale="normal", convolution = FALSE),
#    wG=1.0, wSAS=0.2, avRes=10, pareto=TRUE,
#    constraints=NULL, cdim=0, popsize=40, generations=c(40),
#    cprob=0.85, lower.bounds=boundsL, upper.bounds=boundsU)

## show results
#plot(resMCO, xlab="SAS(Q) residual", ylab="G(r) residual")
#paretoSet(resMCO)
```

---

simPart *Functions to simulate the atomic positions in a nanoparticle*

Description

Functions to simulate the deterministic atomic positions in a nanoparticle and displace those positions stochastically to model thermal effects

Usage

```
simPart(atoms, sym = "fcc", latticep = 4.08, r=10,
    atomsShell=NA, symShell = NA, latticepShell = NA, rcore = NA,
    shell=NA, box=NA, ellipse=NA, pDimer=0, pStack=0,
    rcenter=FALSE, center=c(0,0,0),
    move=TRUE, rotShell=FALSE, rcenterShell=FALSE)

displacePart(nanop, sigma=NA, rcenter=FALSE, latticep=4.08)
```

Arguments

atoms	list with elements describing basis atoms in the primitive cell. Elements should be the same type as the return value of createAtom; see examples.
sym	character describing the structure to be used: "fcc" for face-centered cubic structure, "bcc" for body-centered cubic structure, "sc" for simple cubic structure, "hcp" for hexagonal close-packed structure.
latticep	numeric vector indicating the lattice parameter(s). Should be given in the form: c(a) for fcc, bcc and sc structure, c(a, c) for hcp structure.
r	numeric vector indicating the radius (radii) of the nanoparticle. If length is greater than 1 particles for all given radii are simulated. See Value for details.
atomsShell	list with elements describing basis atoms in the primitive cell for the particle shell.
symShell	character describing the structure associated with the shell. See sym for details.
latticepShell	numeric vector indicating the shell lattice parameter(s). See latticep for details.
rcore	numeric vector which, if not NA, determines the radius (radii) of the core. Should be the same length as vector r.
shell	numeric which, if rcore=NA, determines the particle shell thickness.
box	numeric vector. If not NA particle is simulated in the form of parallelepiped with dimensions box[1], box[2] and box[3].
ellipse	numeric vector. If not NA particle is simulated in the form of ellipse; ellipse[1], ellipse[2] and ellipse[3] give the three elliptic radii.
pDimer	numeric indicating probability of atom to form a cluster with its neighbour.
pStack	numeric indicating probability of stacking fault to appear in each atomic plane. Currently is not available for perovskite symmetry. Produces ABCACABC stacking for cubic lattice and ABABABCBC stacking for hexagonal lattice.
rcenter	logical value indicating whether to choose the center of the particle at random within the unit cell.
center	If rcenter=FALSE, the position at the particle center.

`move` logical value indicating whether to shift atoms near the core-shell boundary to avoid unphysically small interatomic distances. Might not work correctly if `r` length is greater than `1`.

`rotShell` logical value indicating whether to rotate the particle shell by a random angle with respect to the core.

`rcenterShell` logical value indicating whether to choose the center of the particle shell at random. If `FALSE` then the shell center coincide with that of the core.

`sigma` numeric vector which determines the variances of the Gaussian displacements from the mean atomic positions throughout the nanoparticle. If `NA` value is taken from `nanop` attributes. If the particle core and shell contain `Nc` and `Ns` different atom types, respectively, then the first `Nc` elements in vector `sigma` correspond to atoms within the core and the next `Ns` elements describe Gaussian displacements for the shell atoms. See `exapmles` for more details.

`nanop` numeric matrix in which each row represents the position of an atom, e.g., as returned by `simPart`.

Value

numeric matrix with three columns in which each row represents an atomic position and list of the following attributes that describe particle properties:

`"nStacks"` number of stacking faults simulated in the particle.

`"rowcore"` number of atoms within the core.

`"rowshell"` number of atoms within the shell.

`"center"` position of the particle center.

`"nAtomTypes"` number of different atom types in the particle, e.g. for a core/shell particle with "Cu"/"ZnS" symmetry `nAtomTypes=3`.

`"atomType"` vector of length `rowcore+rowshell` that flags atoms in the particle with their atom type numbers. Atoms in the core are numbered with notation $1, 2, \ldots$, while atoms in the shell have numbers $-1, -2, \ldots$, e.g. for a core/shell particle with "Cu"/"ZnS" symmetry vector `atomType` consists of elements `{1, -1, -2}`

`"scatterLength"`
numeric vector describing neutron scattering lengths for all atom types in the particle

.

`"scatterFactor"`
list describing X-ray scattering factor parameters. See `calcTotalScatt` for details.

`"sigma"` see `sigma` argument.

`"layer_end", "layer_start"`
numeric vectors that are used if vector `r` length is greater than `1`. In that case for optimization purposes only one particle with the biggest value of radius `r` is generated, together with the information on how to extract atomic positions for the subparticle with given radius. E.g. if `r=c(8, 11.1, 16)`, `nanop`

is an object that contains resulting matrix, "layer_start=c(1,11,72)" and "layer_end=c(1024,1065,2048)" then nanop[11:1065,] gives atomic coordinates for the subparticle with radius r=11.1.

"layerS_end", "layerS_start"
: numeric vectors. Indicates layers for the particle shell.

"dimer"
: logical indicating whether the particle is a cluster of two spherical particles.

"r"
: If shape="sphere" particle radius (radii). If shape="ellipse" elliptic radii. If shape="box" parallelepiped half-dimensions.

"rcore"
: If shape="sphere" core radius (radii). If shape="ellipse" core elliptic radii. If shape="box" core half-dimensions.

"shape"
: particle shape.

"sym"
: particle (core) symmetry.

"symShell"
: particle shell symmetry.

"atomsCore"
: names of the chemical elements in the particle core.

"atomsShell"
: names of the chemical elements in the particle shell.

Examples

```
## Uncomment all plotPart() calls for 3D visualization

## create CdSe nanoparticle:
## CdSe particle has wurtzite structure
## Zn and S atoms could be used to create base matrix
## (see getBase() )
Cd_base <- getBase("Zn")  #get fractional coordinates for Zn
Cd_scL <- getScatterLength("Cd") #get scattering parameters for Cd
Cd_scF <- getScatterFactor("Cd")

Cd <- createAtom("Cd", base=Cd_base, scatterLength=Cd_scL,
    scatterFactor=Cd_scF, sigma=0.01)

Se_base <- getBase("S")
Se_scL <- getScatterLength("Se")
Se_scF <- getScatterFactor("Se")

Se <- createAtom("Se", base=Se_base, scatterLength=Se_scL,
    scatterFactor=Se_scF, sigma=0.008)

## atoms Cd and Se now can be used as arguments in simPart
part <- simPart(atoms=list(Cd, Se), latticep=c(4.3, 7.02),
    sym="hcp")
#plotPart(part)

## Deterministic particle
## Particle with uniform displacements
Cu <- createAtom("Cu")
part <- simPart(atoms=list(Cu), atomsShell=list(Cu), rcore=8,
    latticep=5)
partx <- displacePart(part, sigma=.02)
```

```
#plotPart(partx, radius=.4, box=TRUE)

## Particle with displacements in the core different
## from displacements in the shell
## create rock salt structure
Zn <- createAtom("Zn")
S <- createAtom("S")
part <- simPart(atoms=list(Zn,S), atomsShell=list(Cd, Se), r=14,
    rcore=10, sym="hcp", latticep=c(3.1, 4.1), symShell="hcp",
    latticepShell=c(4.3, 7.02))
partx <- displacePart(part, sigma=c(.01, .005, .012, .012))
## first elements in sigma and correspond to Zn atoms,
## second - to S atoms, third - to Cd atoms, last - to Se atoms.
attributes(part)$atomType
## elements '1' indicate Zn atoms in the total matrix, '2' indicate
## S atoms, '-1' indicate Cd atoms, and '-2' indicate Se atoms.
#plotPart(partx, radius=.4)

## Particles with radii drawn from a log-Normal size-distribution
## and constant thickness 0.8
r <- exp(rnorm(10, log(10), log(1.1)))
part <- simPart(r=r, shell=0.8, atoms=list(Cu),
    atomsShell=list(Cu))
## particle attributes
attributes(part)
## Extract second subparticle
t1 <- attributes(part)$layer_start[2]
t2 <- attributes(part)$layer_end[2]
part2 <- part[t1:t2, ]
## In order to use part2 as an argument for calcTotalScatt() or
## other functions certain attributes should be set up
## (see calcTotalScatt).
## To avoid extraction of every subparticle use option
##  kind="fast_av" in calcTotalScatt() function.
```

`termRip` *Simulate termination ripples*

Description

Function to simulate termination ripples in the PDF calculated as the Fourier transform of the total scattering function that arise due to the finite value of the upper integration limit Qmax.

Usage

```
termRip(pdf, Qmax, dr, maxR, maxRTermRip)
```

Arguments

pdf	numeric vector containing PDF.
Qmax	numeric indicating value of Qmax at which scattering function was truncated before calculating PDF as the Fourier transform.
dr	numeric indicating the step size in r.
maxR	numeric indicating the maximum value of r in G(r).
maxRTermRip	numeric indicating the maximum value of r in G(r) for which the ripples should be simulated.

Value

numeric vector of function values.

See Also

calcPDF, calcQDepPDF

Examples

```
## simulate a particle
Cu <- createAtom("Cu")
part <- simPart(atoms=list(Cu), atomsShell=list(Cu), r=12,
    rcore=10, latticep=4.08, latticepShell=3.89)

## calculate the PDF
gr1 <- calcPDF(part, maxR=24, scatterLength=c(4.87, 7.97))
gr1 <- broadPDF(gr1, sigma=c(.01, .01), nAtomTypes=2)
## normalization
gr1$gr <- 4*pi*gr1$r*gr1$gr

## calculate and substract the baseline term
gQSAS <- calcTotalScatt(part, type="neutron",  minQ=0.001,
    maxQ=0.771, dQ=0.005, scatterLength=c(4.87, 7.97),
    sigma=c(.01, .01))
gammaR <- calcQDepPDF(minR=0.01, maxR=24, dr=0.01, maxQ=.771,
    minQ=0.001, verbose=50,
    preTotalScat=list(Q=gQSAS$Q, gQ=gQSAS$gQ))
gr1$gr <- gr1$gr - gammaR$gr

plot(gr1$r, gr1$gr, type="l", xlim=c(0,10))
## simulate termination ripples
rvector <- gr1$r
gr1 <- termRip(gr1$gr, Qmax=11.881, dr=0.01, maxR=24, maxRTermRip=16)
## plot the PDF with simulated termination ripples:
lines(rvector, gr1, col=2)

#### compare with ripples due to finite value of Qmax:
## calculate scattering function
gQ <- calcTotalScatt(part, type="neutron",
    scatterLength=c(4.87, 7.97),  sigma=c(.01, .01))
```

```
## set Qmax to a value between 11 and 13 at which scattering function reaches
##  minimum:
t1 <- which(gQ$Q > 11)[1]
t2 <- which(gQ$Q > 13)[1]
cut <- gQ$Q[t1:t2][which(abs(gQ$gQ[t1:t2])
    ==min(abs(gQ$gQ[t1:t2]) ))[1]]
## calculate Q-dependent PDF
gr3 <- calcQDepPDF(minR=0.01, maxR=24, dr=0.01, minQ=.771,
    maxQ=cut, verbose=50, preTotalScat=list(Q=gQ$Q, gQ=gQ$gQ))

## add Q-dependent PDF to plot:
lines(gr3$r, gr3$gr, col=4)
legend(5,7,c("PDF as returned by calcPDF", "PDF with simulated
    termination ripples", "PDF as Fourier transform"), lty=1,
    col=c(1,2,4))
```

Index