Package 'simLife'

September 27, 2018

Title Simulation of Fatigue Lifetimes

Version 0.5.1

Date 2018-09-27

Author Markus Baaske [aut, cre],
Felix Ballani [aut, ctb]

Maintainer Markus Baaske <markus.baaske@uni-jena.de>

Description Provides methods for simulation and analysis of a very general fatigue lifetime model for (metal matrix) composite materials.

Depends R (>= 3.1.0)

Imports grDevices, graphics, stats, splancs

Suggests parallel, rgl, lattice, GenSA, unfoldr

License GPL (>= 3)

Repository CRAN

LazyData no

RoxygenNote 6.1.0

NeedsCompilation yes

Date/Publication 2018-09-27 16:20:03 UTC

R topics documented:

`simLife-package` *Simulation of Fatigue Lifetimes*

Description

Provides methods for simulation and analysis of a very general fatigue lifetime model for (metal matrix) composite materials.

Details

These materials are usually made up of some reinforcement primary phase and a secondary phase which is generally considered to have negative properties with regard to the overall lifetime. A spatial geometric particle model [3,4] can be simulated according to some predefined distributional assumptions and then taken as an input to the simulation routine of the fatigue lifetime model. The distributional model parameters of the lifetime model are specified by the user and usually have to be estimated based on experimental data in general. The lifetime model roughly consists of two parts. First, a random individual failure time for all particles is generated based on two types of projected defects (square root projection areas) dependent on the considered joint size-shape-orientation distribution of the whole particle system. Second, a deterministic projected defects accumulation procedure is applied to find the most hazardous defect region which could lead to an overall failure of the specimen in a real life situation. Additionally we distinguish between particles which are hosted fully inside the material and those lying near the surface. Further, the lifetime model also allows for the inclusion of predefined densely clustered regions of particles which can be simulated beforehand. The user can also provide particular material constants such as Vickers hardness and others according to the phenomenological properties of specific material structures and for appropriateness of the lifetime model simulations. As an add-on hardcore particle packings can be generated by the well-known Random Sequential Adsorption (RSA) algorithm.

The high level functions `simFracture`, `woehler` and `woehlerDiagram` are intended as a simple introduction to utilizing the package.

References

- Y. Murakami (2002). Metal Fatigue: Effects of Small Defects and Nonmetallic Inclusions. Elsevier, Amsterdam.
- J.W. Evans. Random and cooperative sequential adsorption. Rev. Mod. Phys., 65: 1281-1304, 1993.
- M. Baaske, A. Illgen, A. Weidner, H. Biermann, F. Ballani (2018). Influence of ceramic particle and fibre reinforcement in metal-matrix-composites on the VHCF behaviour. Part I: Stochastic modelling and statistical inference. In: Christ, H.-J. (ed.), Fatigue of Materials at Very High Numbers of Loading Cycles. Experimental Techniques - Mechanisms - Modeling and Fatigue Assessment. Springer, Heidelberg. In press.
- M. Baaske, A. Illgen, A. Weidner, H. Biermann, F. Ballani (2018). Influence of ceramic particle and fibre reinforcement in metal-matrix-composites on the VHCF behaviour. Part II: Stochastic modelling and statistical inference. In: Christ, H.-J. (ed.), Fatigue of Materials at Very High Numbers of Loading Cycles. Experimental Techniques - Mechanisms - Modeling and Fatigue Assessment. Springer, Heidelberg. In press.

`areaCrit` *Critical area*

Description

Calculate critical area

Usage

```
areaCrit(vickers, stress, factor = 1.56, scale = 1e+06)
```

Arguments

`vickers`	specific Vickers hardness
`stress`	stress level to be applied
`factor`	specific material dependant factor, default set to `1.56` for inner defects
`scale`	scale factor, area in square millimeters (default)

Details

The critical area (see reference below) is roughly defined by the projected defect area above which the specific material defect becomes particularly hazardous for the failure of the whole specimen.

Value

critical area in $[mm]^2$

Author(s)

M. Baaske

References

Y. Murakami (2002). Metal Fatigue: Effects of Small Defects and Nonmetallic Inclusions. Elsevier, Amsterdam.

CL *A cluster configuration of spheroids*

Description

A dataset containing cluster regions of spheroids where each list element consists of a spheroids list with additional attributes `class="prolate"`, `interior` (logical), indicating whether the cluster fully lies in the interior of the simulation box

Usage

```
data(AL2MC_20p_k10_F2p_CL)
```

Format

A list of length 9. Each element contains a list of spheroids

Author(s)

M. Baaske

CLF *A cluster configuration of fibers*

Description

A dataset containing clusters of regions of fibers, including a second phase as spheres where each list element consists of a list of fibers with additional attributes `class="cylinder"`, `interior` (logical), indicating whether the cluster fully lies in the interior of the simulation box.

Usage

```
data(AL2MC_15p_k10_F2p_CL)
```

Format

A list of length 11 containing lists of fibers belonging to the corresponding cluster

Author(s)

M. Baaske

densifyCluster *Densify clusters*

Description

Densification of objects within a predefined ball

Usage

```
densifyCluster(F, CL, ctrl, weight = 10, info = FALSE,
  cores = getOption("simLife.mc", 1L), cl = NULL)
```

Arguments

F	non-overlapping particle system
CL	cluster list, defining the different cluster regions
ctrl	control arguments passed to function GenSA
weight	optional, weight factor weight=10 (default)
info	optional, logical, if TRUE return result from GenSA call
cores	optional, number of cores for mulicore parallization with cores=1L (default) by mclapply which also can be set by a global option "simLife.mc"
cl	optional, parallel cluster object

Details

This densification method requires packageGenSA to be installed. Additionally for performance reasons it is recommended to make use of parallization by package 'parallel' installed and available or some 'cluster' object derived from it.

Parts of the (reinforcement) geometry system are densified in order to account for regions where some objects stick together more close than in other regions of the simulation box. These so-called clusters are generated by a densification algorithm minimizing some energy of object positions and their (overlap) distances within an enclosing ball of objects with a predefined cluster radius. In order to prevent overlaps between objects one possibly has to increase the weight factor iteratively and run it again if some objects happen to overlap.

Value

a list of all objects (including newly densified regions) named S and a named list of clustered regions cluster

Author(s)

M. Baaske

Examples

```
## Not run:

## Simulate and densify particle system
## densify and include a secondary phase

library(unfoldr)

## Unless MS-Windows platform
# library(parallel)
# options(simLife.mc=2L)

# simulation box either this kind of list
# or use object spatstat::box3
box <- list("xrange"=c(0,3),"yrange"=c(0,3),"zrange"=c(0,9))

# parameter for spheroids simulation
theta <- list("size"=list(0.25),"shape"=list("s"=0.25), "orientation"=list("kappa"=1))

# for ease of use: constant size particles
S <- simPoissonSystem(theta,lam=15,size="const",type="prolate",
orientation="rbetaiso",box=box,mu=c(0,1,0),pl=1,label="P")

## 2nd. phase (ferrit)
param <- list("size"=list(0.075), "shape"=list("s"=0.75))
F <- simPoissonSystem(param,lam=2,size="const",type="prolate",
box=box,mu=c(0,1,0),pl=1,label="P")

# apply RSA, this may take some
RSA <- rsa(S,F,verbose=TRUE)

## show 3D spheroids (non-overlapping)
# library(rgl)
# cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
# spheroids3d(RSA[1:length(S)], box, col=cols)
# spheroids3d(RSA[length(S):(length(S)+length(F))], box, col="gray")

## construct clusters
CL <- simPoissonSystem(list("size"=list(0.35)), lam=0.1, size="const",
type="spheres", box=box, pl=1, label="F")

CLUST <- simCluster(RSA, CL, verbose=TRUE)
cat("cluster length: ",length(CLUST),"\n")

## show cluster regions
# library(rgl)
# open3d()
# X <- do.call(rbind,lapply(CLUST, function(x) c(x$center,x$r)))
# invisible(lapply(CLUST, function(x) rgl::spheres3d(X[,1:3],radius=X[,4],col="gray",alpha=0.2)))
# cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
# invisible(lapply(CLUST, function(x) spheroids3d(RSA[x$id],box,col=cols)))
```

```
# some controls for 'GenSA'
ctrl <- list(threshold.stop=0.01,max.call=10000)
# densify region to clustered particles
RET <- densifyCluster(RSA, CLUST, ctrl, weight=100, cores = 1L, cl = NULL)

S <- RET$S
CL <- RET$cluster

## show densified clusters
# library(rgl)
# open3d()
# X <- do.call(rbind,lapply(CLUST, function(x) c(x$center,x$r)))
# invisible(lapply(CLUST, function(x)
# rgl::spheres3d(x=X[,1:3],radius=X[,4],col="gray",alpha=0.2)))
# invisible(lapply(CL, function(x) { spheroids3d(x, box, col=cols) }))


## End(Not run)
```

`drawDefectProjections` *Draw defect projections*

Description

Draw defect projections for spheroids, spherocylinders or spheres. The method first constructs the projected objects sampling some points on the borders and calculates the convex hull of such points. The defect object as returned by `simFracture` and used to show the accumulated defects.

Usage

```
drawDefectProjections(F, D, ...)
```

Arguments

`F`	geometry system
`D`	defects object
`...`	further arguments passed to `getSphereProjection`, `getSpheroidProjection`, `getCylinderProjection`

Value

`NULL`

Author(s)

M. Baaske

drawProjection *Draw defect projections of particles*

Description

The function draws the defect projections in a 3d plot after `spheroids3d` has been called.

Usage

```
drawProjection(E, conv = TRUE, np = 25)
```

Arguments

`E`	a list of ellipses (the projected particles)
`conv`	logical, if `conv=TRUE` (default) the convex hull is drawn
`np`	number of points used for sampling the convex hull of projections

Value

draw projections and the convex hull (if enabled) in a 3d plot return area of polygon and points of convex hull

Author(s)

M. Baaske

extractClusters *Extract defects*

Description

Extract all defects with some area value larger than `area`

Usage

```
extractClusters(clust, area)
```

Arguments

`clust`	list of defects, see `simDefect`
`area`	the area value lower bound

Details

The function extracts all defects (convex hulls of projected defect areas) which have a larger area value than the predefined `area` value for some further analysis.

Value

list of clusters

Author(s)

M. Baaske

`getCrackTime` *Generate individual fracture time*

Description

Generate individual defect time for particle fracture

Usage

```
getCrackTime(theta, a, b, stress, vickers, p, const)
```

Arguments

`theta`	polar angle
`a`	axis length (axis orthogonal to rotational axis)
`b`	rotational axis length
`stress`	stress level
`vickers`	Vickers hardness
`p`	simulation parameter set
`const`	constant parameters for crack

Details

The particle fracture (crack) is assumed to happen orthogonal to the major axis direction along the maximum minor axis length. Thus the projection area can be easily computed for the purpose of defect accumulation. The parameter set is made up of six parameters. Here only the second and third parameters are used to simulate the defect `crack` times. The failure times follow a Weibull distribution with scale parameter $p2 * a^2/(b * \sigma * cos\theta * Hv)$ and shape parameter $p3$ where σ denotes the stress, a the minor axis length and Hv the Vickers hardness. The angle θ is measured between the rotational axis and the axis of main load direction. In this way we account for the orientation of particles (spheroids) when generating fracture times dependent on their tendency to be more or less oriented towards the main load direction.

Value

numeric, the individual fracture time

Author(s)

Felix Ballani, M. Baaske

See Also

getDelamTime

---

getCylinderProjection *Fiber defect projections*

Description

The function draws the defect projections after cylinders3d has been called and returns the points for the convex hull with its points of each projected fiber (spherocylinder).

Usage

```
getCylinderProjection(S, B = rep(1, length(S)), draw = TRUE,
  conv = TRUE, np = 20)
```

Arguments

S	fibers to be projected
B	integer vector of length equal to S of defect types, B=0 for crack and B=1 for delamination.
draw	logical, draw=TRUE (default) draw the projected fibers
conv	logical, conv=TRUE (default) draw the convex hull of projected fibers
np	number of points used for sampling the convex hull of projections

Value

draw defect projections and the convex hull (if enabled) in a 3d plot returning its polygonal area and points of the convex hull

Author(s)

M. Baaske

Examples

```
## Not run:

## Simulate Poisson cylinder system,
## generate a non-overlapping system by RSA,
## densify a cluster of cylinders

library(unfoldr)
```

```
## Unless MS-Windows platform
# library(parallel)
# options(simLife.mc=2L)

lam <- 35
box <- list("xrange"=c(0,3),"yrange"=c(0,3),"zrange"=c(0,9))

## Spheroids of constant sizes
theta <- list("size"=list(.5),"shape"=list("radius"=0.1),
"orientation"=list("kappa"=0.1))

S <- simPoissonSystem(theta,lam,size="const",type="cylinders",
orientation="rbetaiso",box=box,pl=1,label="P")

## secondary phase: particles as spheres
F <- simPoissonSystem(list("size"=list(0.075)), lam=5, size="const",
type="spheres",box=box, pl=1, label="F")

## apply RSA
S2 <- rsa(S,F,pl=101,verbose=TRUE)
#####################################################################
## Optional: 3D visualization of cylinder projection areas
#####################################################################

#require(rgl)
#id <- c(1,5,9,32,10)
#cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
#cylinders3d(S2[id], box, col=cols)
#P <- getCylinderProjection(S2[id], B=c(0,1,0,1,1), draw=TRUE, conv = TRUE, np=20)
#P <- getCylinderProjection(S2[id], B=c(0,0,0,0,0), draw=TRUE, conv = TRUE, np=20)
#P <- getCylinderProjection(S2[id], B=c(1,1,1,1,1), draw=TRUE, conv = TRUE, np=20)

## construct clusters
CL <- simPoissonSystem(list("size"=list(0.35)), lam=0.1, size="const",
type="spheres", box=box, pl=1, label="F")

CLUST <- simCluster(S2, CL, cond=list("eps"=1e-7,"minSize"=1L), verbose=TRUE, pl=1)

## densify
ctrl <- list(threshold.stop=0.01,max.call=5000,verbose=FALSE)
RET <- densifyCluster(S2, CLUST, ctrl, weight=10, cores = 1L, cl = NULL)
G <- RET$cluster

#####################################################################
## Optional: 3D visualization of densified sphere clusters
#####################################################################

#open3d()
#lapply(CLUST,function(x) cylinders3d(S2[x$id],box=box,col=cols))
#X <- do.call(rbind,lapply(CLUST, function(x) c(x$center,x$r)))
#invisible(lapply(CLUST, function(x) rgl::spheres3d(X[,1:3],radius=X[,4],col="gray",alpha=0.2)))
#
```

```
### draw densified cluster
#open3d()
#invisible(lapply(G,function(x) { cylinders3d(x,box=box,col=cols) }))
#invisible(lapply(CLUST, function(x) rgl::spheres3d(X[,1:3],radius=X[,4],col="gray",alpha=0.2)))


## End(Not run)
```

`getDelamTime` *Generate interface debonding times*

Description

Generate individual defect times for particle debonding

Usage

```
getDelamTime(E, stress, param, inF = 0.5, outF = 0.65)
```

Arguments

`E`	object of class `"oblate"`, `"prolate"`, `"cylinders"`, `"spheres"`
`stress`	stress level
`param`	simulation parameter vector, see details
`inF`	weightening factor for inner defect projection
`outF`	weightening factor for outer defect projection

Details

This kind of failure time (time of debonding from the metal matrix structure) roughly depends on the projected area of the object, the applied overall stress level and whether the object lies in the interior of the simulation box or hits one of the box boundaries. The parameter vector is made up of six parameters where the second and third parameters are used to simulate the defect type `crack`, see `getCrackTime`. The order is as follows: $p1$ probability of already materialized defects, scale factor $p2$, shape factor $p3$, shift parameter $p4$ of log times, the slope $p5$ and $p6$ denoting the standard deviation of the random errors of the log times. Only the last three parameters $p4, p5, p6$ are used to determine the defect time for debonding of the considered object.

Value

numeric, the individual debonding time

Author(s)

Felix Ballani

getSphereProjection *Sphere projections*

Description

The function draws the defect projections in a 3d plot and returns the points of its the convex hull together with points of each projected particle (here a sphere).

Usage

```
getSphereProjection(S, draw = TRUE, conv = TRUE, np = 20)
```

Arguments

S	particles to be projected
draw	logical, draw=TRUE (default) draw the projected spheres
conv	logical, conv=TRUE (default) draw the convex hull
np	number of points used for sampling the convex hull of projections

Value

draw projections and the convex hull (if enabled) in a 3d plot returning its polygonal area and points of the convex hull

Author(s)

M. Baaske

getSpheroidProjection *Projection of defect types*

Description

In case one wishes to analyze the defect projections directly the function returns the projected objects itself. The two types of defects lead to different projected objects. In case of defect type crack the circle of maximum radius of the particle is orthogonally projected otherwise the whole particle is projected in the main load direction, i.e. the xy plane.

Usage

```
getSpheroidProjection(S, type)
```

Arguments

S	list of (non-overlapping) particles (spheroids)
type	either crack or delam

Details

Get the projected objects (ellipses) of particles dependant on their defect types

Value

returns a list of 2d ellipse objects.

Author(s)

M. Baaske

`plotDefectAcc` *Draw accumulation path damage areas*

Description

Plot accumulation of damage projection areas

Usage

```
plotDefectAcc(CL, last.path = FALSE, log.axis = "x", use.col = TRUE,
  main = "", ...)
```

Arguments

`CL`	result of accumulation simulation, see `simFracture`
`last.path`	logical, `last.path=FALSE` (default), show the critical area accumulation path leading to the overall failure
`log.axis`	character, `log.axis='x'` (default), switch to logarithmic scale
`use.col`	optional, do colering or black and gray values
`main`	optional, main title of the plot
`...`	optional, graphical parameters, see `par`

Details

Simple plotting function to visualize the process of damage accumulation separately for interior objects and objects hitting the surface of the simulation box (specimen)

Value

the accumulation path of damage areas as a list containing the starting and end points of each horizontal line (the current convex hull area value) together with the starting point of the next convex hull of damage projections and whether the damage occured in the interior or at the surface of the simulation box

Author(s)

M. Baaske

`rsa` *Random sequential adsorption (RSA)*

Description

A simple RSA algorithm

Usage

```
rsa(S, F = NULL, pl = 0, verbose = TRUE)
```

Arguments

`S`	overlapping geometry system
`F`	secondary phase objects as list
`pl`	integer: if `pl>0` give some intermediate results
`verbose`	logical: if `verbose=TRUE` (default) show additional information

Details

The function generates a non-overlapping configuration of spheres, spheroids or spherocylinders with respect to periodic boundary conditions by sequentially adding the given objects at new random positions in the simulation box while keeping their sizes and orientations. If there is an overlap detected the position is rejected and some new random position is generated until all particles have put inside or the maximum number of allowed iterations is reached. This function is most suited for volume fractions less than 0.15.

Value

list of non-overlapping particles

Author(s)

Felix Ballani, Markus Baaske

References

J.W. Evans. Random and cooperative sequential adsorption. Rev. Mod. Phys., 65: 1281-1304, 1993.

S — *A packed spheroids configuration (particle system)*

Description

A dataset containing non-overlapping spheroids where each list element consists of the following variables:

Usage

```
data(AL2MC_20p_k10_F2p_S)
```

Format

A list of length 4905 containing lists each of length six

Details

- id the current cluster id
- center the center of ball which defines the cluser region
- u the orientation vector of spheroid's rotational symmetry axis
- acb two shorter semi-axis lengths and major semi-axis length
- angles polar angle and azimuthal angle
- rotM rotational matrix, center of rotation is the center of the spheroid
- interior attribute, whether the spheroid lies in the interior of the simulation box
- label attribute, character, user defined label, here: 'P' (particle) or 'F' (ferrit)
- area attribute, the projected area

Author(s)

M. Baaske

SF — *A packed (sphero)cylinder configuration (fiber system)*

Description

A dataset containing non-overlapping spherocylinders where each list element consists of the following variables:

Usage

```
data(AL2MC_15p_k10_F2p_S)
```

Format

A list of length 3550 containing lists each of length nine

Details

- id the current cluster id
- center the center of ball which defines the cluser region
- origin0 the center of the fist cylinder cap
- origin1 the center of the second cylinder cap
- length length/height of cylinder without radii of caps
- u orientation vector of spheroid's rotational symmetry axis
- r radius of cylinder
- angles two shorter semi-axis lengths and major semi-axis length
- rotM rotational matrix, center of rotation is the center of the spheroid
- interior attribute, whether the spheroid lies in the interior of the simulation box
- label attribute, character, user defined label, here: 'P' (particle) or 'F' (ferrit)
- area attribute, the projected area

Author(s)

M. Baaske

---

showDensity *Plot estimated densities*

Description

Plot the estimated densities which result from the randomly generated individual failure times.

Usage

```
showDensity(dv, main = "Failure time density estimation", ...)
```

Arguments

dv	named list of individual failure times
main	title of the plot (optional)
...	optional, additional graphics parameters

Author(s)

M. Baaske

Examples

```
## Simulation of individual defect times
## of some particle system
data(AL2MC_20p_k10_F2p_S)

## generate individual failure times
opt <- list("vickers"=107,"distTol"=1,"Tmax"=10^11,
"inAreafactor"=1.56, "outAreafactor"=1.43,
"pointsConvHull"=10, "scale"=1e+06,"pl"=0)

par <- list("P"=c(0.01,6,0.5,75,-15,3),
"F"=c(0,0,0,105,-12,1),
"const"=NULL)

## simulate times
CLT <- simTimes(S,par,vickers=opt$vickers,stress=125,cores=1L)

## times
T <- unlist(sapply(CLT,`[[`,"T"))
V <- unlist(sapply(CLT,`[[`,"V"))
U <- unlist(sapply(CLT,`[[`,"U"))

## show estimated densities
showDensity(list("Delamination"=log10(V),"Crack"=log10(U),"Time"=log10(T)),xlim=c(-2,15))
```

---

simCluster *Generate clustered regions*

Description

Construct regions of clustered objects

Usage

```
simCluster(S, CL, cond = list(eps = 1e-06, minSize = 1L),
  do.rsa = FALSE, verbose = FALSE, pl = 0L)
```

Arguments

S	(non-overlapping) geometry objects system
CL	cluster regions, i.e. objects of class 'spheres', possibly overlapping
cond	conditioning object for cluster algorithm, see details
do.rsa	optional, whether to apply rsa algorithm to cluster spheres
verbose	optional, verbose output, verbose=FALSE (default)
pl	integer, pl>0 for printing information

Details

The function takes a non-overlapping system of spheres, cylinders, spheroids of type "prolate" or "oblate and a random ball configuration CL of class "spheres" which defines clustered regions of the objects given in S according to the minimum required distance eps between these objects and a number of objects required in that region by minSize in the list cond. The function rsa is internally called if do.rsa=TRUE in order to make these cluster regions non-overlapping.

Value

a list of the following element:

- id numeric vector of ids of including spheroids
- center the center of enclosing ball
- r the radius
- interior equals to 0 if any of the enclosed ball objects hit the boundaries of the simulation box and equals 1 otherwise

Author(s)

M. Baaske

Examples

```
## Not run:

## Simulate and densify particle system
## densify and include a secondary phase

library(unfoldr)

## Unless MS-Windows platform
# library(parallel)
# options(simLife.mc=2L)

# simulation box either this kind of list
# or use object spatstat::box3
box <- list("xrange"=c(0,3),"yrange"=c(0,3),"zrange"=c(0,9))

# parameter for spheroids simulation
theta <- list("size"=list(0.25),"shape"=list("s"=0.25), "orientation"=list("kappa"=1))

# for ease of use: constant size particles
S <- simPoissonSystem(theta,lam=15,size="const",type="prolate",
orientation="rbetaiso",box=box,mu=c(0,1,0),pl=1,label="P")

## 2nd. phase (ferrit)
param <- list("size"=list(0.075), "shape"=list("s"=0.75))
F <- simPoissonSystem(param,lam=2,size="const",type="prolate",
box=box,mu=c(0,1,0),pl=1,label="P")
```

```
# apply RSA, this may take some
RSA <- rsa(S,F,verbose=TRUE)

## show 3D spheroids (non-overlapping)
# library(rgl)
# cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
# spheroids3d(RSA[1:length(S)], box, col=cols)
# spheroids3d(RSA[length(S):(length(S)+length(F))], box, col="gray")

## construct clusters
CL <- simPoissonSystem(list("size"=list(0.35)), lam=0.1, size="const",
type="spheres", box=box, pl=1, label="F")

CLUST <- simCluster(RSA, CL, verbose=TRUE)
cat("cluster length: ",length(CLUST),"\n")

## show cluster regions
# library(rgl)
# open3d()
# X <- do.call(rbind,lapply(CLUST, function(x) c(x$center,x$r)))
# invisible(lapply(CLUST, function(x) rgl::spheres3d(X[,1:3],radius=X[,4],col="gray",alpha=0.2)))
# cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
# invisible(lapply(CLUST, function(x) spheroids3d(RSA[x$id],box,col=cols)))

# some controls for 'GenSA'
ctrl <- list(threshold.stop=0.01,max.call=10000)
# densify region to clustered particles
RET <- densifyCluster(RSA, CLUST, ctrl, weight=100, cores = 1L, cl = NULL)

S <- RET$S
CL <- RET$cluster

## show densified clusters
# library(rgl)
# open3d()
# X <- do.call(rbind,lapply(CLUST, function(x) c(x$center,x$r)))
# invisible(lapply(CLUST, function(x)
# rgl::spheres3d(x=X[,1:3],radius=X[,4],col="gray",alpha=0.2)))
# invisible(lapply(CL, function(x) { spheroids3d(x, box, col=cols) }))


## End(Not run)
```

---

simCrackTime *Defect failure times*

Description

Simulation of individual defect failure times. For a secondary phase only the defect type "delamination" is considered.

Usage

```
simCrackTime(S, stress, vickers, param, cores = getOption("simLife.mc",
  1L))
```

Arguments

S	geometry objects system
stress	stress level for generation of failure times
vickers	Vickers hardness, see details
param	list of parameter vectors for simulation of failure times for both phases
cores	optional, number of cores for mulicore parallization with cores=1L (default) by mclapply which also can be set by a global option "simLife.mc"

Value

a list with the following elements:

- id id of particle
- U crack failure time
- V delamination failure time
- T the minimum of both failure times
- B failure type, either 0 for particle crack or 1 for particle delamination
- A projection area set to zero for later use
- label either label="P" for primary phase or label="F" for secondary phase

Author(s)

Felix Ballani, Markus Baaske

---

simDefect *Local damage accumulation*

Description

Simulate fatigue lifetime of the geometry system

Usage

```
simDefect(stress, S, CLT, opt)
```

Arguments

stress	stress level
S	non-overlapping geometry system
CLT	the start (initialized) cluster of objects
opt	control list of fixed parameters, see simTimes

Details

The function simulates the fatigue lifetime model for the given geometry system. The overall fatigue lifetime is determined by the first accumulated convex area of projected defects which exceeds the predefined critical area. The process of defect accumulation is based on the general results for VHCF fatigue behaviour of material structures published by Murakami. Here we treat two nearby defects as an enlarged common one if the weighted (single-linkage) Euclidean distance inbetween (in 3D) is less than some portion (possibly including some material specific constant) of the minimum of the corresponding square roots of projected defect areas. The weight is chosen as the reciprocal value of the cosine of the angle between the main load direction and the rotational axis of the chosen object having minimum distance to the compared defect . Further, the argument `opt` defines the list of controls of fixed parameters with the following elements: `vickers` Vickers hardness, `distTol` as a (proportion) factor of the minium the corresponding square roots of projected defect areas (might be less than 0.95 but in general depends on the inverstigated material structure), `inAreafactor` default set to value 1.56, `outAreafactor` default set to value 1.43, `pointsConvHull` number of sampling points of each objects projection border for approximating the defect projection convex area, `scale` scaling factor (should be set to `1e+06` for μm^2), `pl` print level of information output with some verbose messages for any value larger than 100. In this case only the accumulated defect projections after adding the last individual failed object are returned. Here the last element contains the defect with the highest projected area which leads to the overall failure.

Value

list of clusters, the following values are returned for further analysis:

- id the object id numbers in the accumulated defect
- n internal: accumulated defect id, zero means last element of cluster
- B corresponding defect types of projected objects
- interior whether each object is interior or not
- A projected areas, possibly convex area
- inner whether the defect lies in the interior or not
- T random individual times
- label phase labels
- areaMax attribute, the maximum area of defect projections
- aIn attribute, critical area for inner defect projections
- aOut attribute, critical area for outer defect projections
- maxSize attribute, maximum number of projected objects found in a defect

Author(s)

M. Baaske

`simFracture` *Fatigue lifetime simulation*

Description

Simulation of fatigue lifetime model

Usage

```
simFracture(stress, S, opt, param, last.defect = FALSE, CL = NULL)
```

Arguments

`stress`	applied stress level
`S`	non-overlapping geometry system
`opt`	control list for simulation of individual failure times
`param`	parameter list for generating individual failure times
`last.defect`	logical, `last.defect=FALSE` (default), return either all defect accumulations or only the last one
`CL`	optional, NULL (default), predefined clustered regions of objects

Details

We provide a wrapper function for `simDefect` to simulate the proposed fatigue lifetime model including the generation of individual failure times with additional information of the responsible accumulated defect projection area (simply called defect area) which leads to the failure of the system. This information is returned in the following list with elements: `crack` the responsible defect itself, `T` the individual failure times of objects aggregated as a cluster of objects, `ferrit` whether any secondary phase (here called `ferrit`) is involved for overall failure, `interior` whether the defect occurs in the interior or at the boundaries of the simulation box. The optional list argument `CL` defines clustered regions, see `simCluster`, of objects sticking together more close than others. This is useful in case one also wishes to model densely lieing agglomerations of objects (i.e. clusters of particles or fibers) as an obvious phenomenon of some specimen in mind. This list is made up of the following elements: `id`, the id of the region, `center` the center point of the corresponding region, `r` the radius (as the Euclidean distance from the center point) which within all objects belong to this region, `interior` whether any object within radius `r` hits the box boundaries of the simulation box.

Value

a list made up of the following objects if `last.defect=FALSE` (default):

- fracture return value of `simDefect`
- times return value of `simTimes`
- cl_info additional cluster information of responsible defect cluster

otherwise only `cl_info` is returned.

Author(s)

M. Baaske

Examples

```
## Not run:

## Simulate a defect accumulation of a particle system

## primary particles (as prolate spheroids)
## and secondary phase (as spheres)
data(AL2MC_20p_k10_F2p_S)

## which is clustered and densified according to CL
## additional predefined clustered regions
data(AL2MC_20p_k10_F2p_CL)

## the box is stored together with the geometry system
box <- attr(S,"box")

## distTol=0.25, so use 25% of accumulation distance
opt <- list("vickers"=107, "distTol"=0.25, "Tmax"=10^12,
"inAreafactor"=1.56, "outAreafactor"=1.43,
"pointsConvHull"=10, "scale"=1e+06,"pl"=101)

## constants 'const' are set to the following internally
## and can partially be overwritten if needed
# const <- list("Em"=68.9,"Ef"=400,"nc"=28.2,"nu"=0.33,
#     "pf"=0.138,"nE"=NULL,"sigref"=276,"Vref"=5891)

par <- list("P"=c(0.01,5,0.5,50,-11,3),
"F"=c(0,0,0,105,-12,1),"const"=NULL)

# stress amplitude applied
stress <- 110
## generate individual (particles') failure times
CLT <- simTimes(S,par,vickers=opt$vickers,stress=stress,cores=1L)

## generated random failure times
T <- unlist(sapply(CLT,`[[`,"T"))
V <- unlist(sapply(CLT,`[[`,"V"))
U <- unlist(sapply(CLT,`[[`,"U"))

dev.new()
showDensity(list("Debonding"=log10(V),"Fracture"=log10(U),"Failure time"=log10(T)),xlim=c(-2,15))

# Area max
(inA <- areaCrit(opt$vickers, stress=stress, factor=opt$inAreafactor, scale=opt$scale))
(outA <- areaCrit(opt$vickers, stress=stress, factor=opt$outAreafactor, scale=opt$scale))

## run accumulation
RET <- simDefect(stress,S,CLT,opt)
```

```
length(RET)

#### alternatively, includes simulating times by 'simTimes'
# SIM <- simFracture(stress,S,opt,par,last.defect=FALSE,CL=CL)
# SIM$cl_info

## some simple analysis of accumulation paths until failure,
## that is, while the critical area is exceeded
LR <- RET[[length(RET)]]
isInCluster <- any(unlist(lapply(CL,function(x,y)
any(y$id %in% x$id) , y=LR)))
cat("Broken cluster: ", isInCluster,"\t Ferrit: ",
any("F" %in% LR$label),"\t Acc.size",length(LR$id),"\n")

## select only clusters of size larger than 'msize'
msize <- 1
id <- sapply(RET,function(x) ifelse(length(x$id)>msize,TRUE,FALSE))
cat("Number of defect projections in last cluster: ",length(RET[[length(RET)]]$id),"\n")

## draw all accumulation paths until failure
dev.new()
L <- plotDefectAcc(RET,last.path=FALSE)
## draw last accumulation path until failure
dev.new()
L <- plotDefectAcc(RET,last.path=TRUE)

## 3D visualization of final defect projection area
#library(rgl)
#library(unfoldr)
#qid <- LR$id
#open3d()
#spheroids3d(S[qid],box=box, col=c("#0000FF","#00FF00","#FF0000","#FF00FF"))
#
### drawing only last cluster leading to failure
#drawDefectProjections(S,list(LR))

## End(Not run)
```

`simTimes` *Crack time simulation*

Description

Simulate crack times

Usage

```
simTimes(S, param, vickers, stress, cores = getOption("simLife.mc", 1L),
  verbose = FALSE)
```

Arguments

S	(non-overlapping) geometry system
param	parameters for generating failure times
vickers	vickers hardness
stress	stress level to be applied
cores	optional, number of cores for mulicore parallization with cores=1L (default) by mclapply which also can be set by a global option "simLife.mc"
verbose	logical, not used yet

Details

The function randomly generates phase dependant failure times of defect types crack and delam. The accumulation structure of defects is initialized containing the failure times of objects in ascending order. The failure times of the defect type crack follow a Weibull distribution, see simCrackTime. The failure times of the defect type delam roughly depend on the projected area of the object, the applied overall stress amplitude and whether the object lies in the interior of the simulation box or hits one of the box boundaries. The argument param consists of a distribution parameter list which contains numeric vectors for both reinforcement objects (labled by P) and an optional second phase (labled by F). If no second phase is considered the corresponding parameter set is simply ignored. Each parameter vector is made up of six parameters in the following order: $p1$ probability of already materialized defects, scale factor $p2$, shape factor $p3$, shift parameter $p4$ of log times, the slope $p5$ and $p6$ as the standard deviation of the random error of log times.

Value

list of increasing failure times

Author(s)

M. Baaske

See Also

getCrackTime, getDelamTime

Examples

```
## Simulation of individual defect times
## of some particle system
data(AL2MC_20p_k10_F2p_S)

## generate individual failure times
opt <- list("vickers"=107,"distTol"=1,"Tmax"=10^11,
"inAreafactor"=1.56, "outAreafactor"=1.43,
"pointsConvHull"=10, "scale"=1e+06,"pl"=0)

par <- list("P"=c(0.01,6,0.5,75,-15,3),
"F"=c(0,0,0,105,-12,1),
```

```
"const"=NULL)

## simulate times
CLT <- simTimes(S,par,vickers=opt$vickers,stress=125,cores=1L)

## times
T <- unlist(sapply(CLT,`[[`,"T"))
V <- unlist(sapply(CLT,`[[`,"V"))
U <- unlist(sapply(CLT,`[[`,"U"))

## show estimated densities
showDensity(list("Delamination"=log10(V),"Crack"=log10(U),"Time"=log10(T)),xlim=c(-2,15))
```

`woehler` *Woehler experiment*

Description

Simulate a Woehler experiment

Usage

```
woehler(S, CL, param, opt, stress, cores = getOption("simLife.mc", 1L),
  cl = NULL)
```

Arguments

`S`	non-overlapping object system
`CL`	predefined clustered regions of objects, see `simFracture`
`param`	parameter list for random generation of individual failure times
`opt`	control parameters, see `simTimes`
`stress`	list of stress levels
`cores`	optional, number of cores for mulicore parallization with `cores=1L` (default) by `mclapply` which also can be set by a global option "`simLife.mc`"
`cl`	optional, cluster environment object wich has priority of usage compared to multicore processing in case of `cores>1`

Details

As done in a real life scenario the Woehler diagram measures the applied stress amplitude versus number of cycles to failure. For each given stress level the function is intended to be an all-in-one wrapper function for fatigue lifetime model simulations for different stress levels which returns a matrix of failure times where each row corresponds to one stress level. The Woehler experiments can be simulated in parallel based on using the package `snow` possibly together with `Rmpi` for an `MPI` cluster object.

Value

matrix of failure times, first colunm corresponds to the times and the second to the stress level

Author(s)

M. Baaske

Examples

```
## Not run:

## Unless MS-Windows
# library(parallel)
# options(simLife.mc=2L)

# primary particles and secondary phase (ferrit)
# which is already clustered and densified
data(AL2MC_20p_k10_F2p_S)

# simulation parameters
opt <- list("vickers"=107,"distTol"=1.0,"Tmax"=10^11,
"inAreafactor"=1.56, "outAreafactor"=1.43,
"pointsConvHull"=10, "scale"=1e+06,"pl"=1L)

# lifetimes parameters
par <- list("P"=c(0.01,6,0.5,75,-15,1.5),
"F"=c(0,0,0,105,-12,1),
"const"=NULL)

nsim <- 10
stress <- as.list(seq(from=90,to=140,by=10))

cl <- NULL
## MPI/SOCKS/PSOCKS cluster object (even on Windows)
## must initialize RNG stream (rlecuyer) for reproducible results
# RNGkind("L'Ecuyer-CMRG")
# cl <- makeCluster(8)
# clusterSetRNGStream(cl)

# the following code may take some time
W <- woehler(S, CL=NULL, par, opt, stress=rep(stress,each=nsim),cores=1L,cl=cl)
woehlerDiagram(W, yrange=c(70,145))

## do not forget to stop cluster if used
if(!is.null(cl)) stopCluster(cl)

## End(Not run)
```

woehlerDiagram *Plot Woehler diagram*

Description

Plot a Woehler diagram for simulated Woehler experiments

Usage

```
woehlerDiagram(W, W2 = NULL, yrange = NULL, main = "",
  legend.text = NULL, ...)
```

Arguments

W	results from function woehler
W2	optional, W2=NULL (default) woheler results
yrange	range of stress levels, the min/max values of stress levels (default)
main	main title of the plot
legend.text	text for the legend as vector of strings, NULL (default) for no legend
...	graphic parameters passed legend

Details

The simulated results of the Woehler experiments are summarized in the load-cycle diagram as usually done in the Woehler diagram.

Value

NULL

Author(s)

M. Baaske

Index