Package ‘afmToolkit’

April 3, 2017

Title Functions for Atomic Force Microscope Force-Distance Curves Analysis

Version 0.0.1

Author Rafael Benitez <rabesua@uv.es>, Vicente Jose Bolos, Jose-Luis Toca-Herrera

Maintainer Rafael Benitez <rabesua@uv.es>

Description Set of functions for analyzing Atomic Force Microscope (AFM) force-distance curves. It allows to obtain the contact and unbinding points, perform the baseline correction, estimate the Young's modulus, fit up to two exponential decay function to a stress-relaxation / creep experiment, obtain adhesion energies. These operations can be done either over a single F-d curve or over a set of F-d curves in batch mode.

Depends R (>= 3.2.2), ggplot2

Imports stats, utils, minpack.lm, grid, gridExtra, scales, dplyr, DBI, assertthat, tibble

License GPL

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-04-03 15:30:48 UTC

R topics documented:

afmAdhesionEnergy .......................................................... 2
afmBaselineCorrection .................................................... 3
afmContactPoint ............................................................ 4
afmdata ................................................................. 5
afmDetachPoint ............................................................ 6
afmExpDecay ............................................................... 8
afmexperiment ............................................................. 9
afmExtract ............................................................... 10
afmIndentation ............................................................ 11
afmReadJPK ............................................................... 11
**afmAdhesionEnergy**

Finds the adhesion and the full detach energies from the retract segment of the AFM F-d curve.

**Usage**

```r
afmAdhesionEnergy(afmdata, width = 1, lagdiff = width, mul, mdj = NULL)
```

**Arguments**

- `afmdata`: An afmdata or afmexperiment class variable. Baseline correction should have been done already.
- `width`: Width of the window for the local regression (in vector position units)
- `lagdiff`: Lag for estimating the differences in Delta (or slopes) signal. By default it takes the same value as the window with.
- `mul`: Multiplier for the calculating the threshold in the estimation of jumps and peaks in the Delta signal
- `mdj`: Minimum distance between jumps. If none is given then it will be set equal to `width`

**Value**

An afmdata class variable which will consist on the original input afmdata variable plus a new list named AdhEner with the following fields:

- **Points**: Array containing the indices of the retract segment where the adhesion begins, the unbinding event takes place and the adhesion ends.
- **Energies**: Data frame with three columns: E1adh, E2adh and Etotal, being the first one the energy from the beginning of the adhesion until the unbinding event, then second one the energy from the unbinding event until the full detachment of the tip, and the third one, the sum of them.
Examples

```r
path <- path.package("afmToolkit")
data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path)
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 10)
data <- afmDetachPoint(data, width = 20, mul1 = 2, mul2 = 30)
data <- afmBaselineCorrection(data)
data <- afmAdhesionEnergy(data, width = 20, mul = 10)
str(data$AdhEner)
```

### Description

This function performs the baseline correction to an AFM F-z curve within an `afmdata` structure. It substracts a best fit line to the curve: for the approach and contact segments, it fits a line to the approach curve points where for which \( |z| > ZPointApp \) and for the retract segment, it fits a line to the retract curve where \( |z| > ZpointRet \).

If no `ZPointApp` is given and the contact point has been already estimated (via `afmContactPoint()` function), then it is found as

\[
ZPointApp = 0.7 \times \text{ContactPoint} + 0.3 \times \text{max}(Z)
\]

### Usage

```r
afmBaselineCorrection(afmdata, ZPointApp = NULL, ZPointRet = NULL,
fitpause = c("approach", "retract", "none"), vsTime = FALSE)
```

### Arguments

- **afmdata**: An `afmdata` structure.
- **ZPointApp**: Point in the approach segment of the curve that defines the approach baseline.
- **ZPointRet**: Point in the retract segment of the curves that defines the retract baseline.
- **fitpause**: Behaviour for the baseline correction at the pause segment: if "approach" (default), the pause segment is correted using the best line fit done on the approach segment, if "retract" the best line fit of the retract segment is used, if "none", no baseline correction is done on the pause segment.
- **vsTime**: Logical. If TRUE then the baseline correction is performed following the Force vs time approach described by S. Moreno-Flores (*Moreno Flores (2016)*).

### Value

`afmdata` An `afmdata` structure identical to the one in the input, but with an additional `ForceCorrected` column in the data dataframe of the `afmdata` structure.
References


Examples

```r
AFMcurve <- afmReadJKP("force-save-JPK-2h.txt.gz", path = path.package("afmToolkit"))
ZPointApp <- 6.43e-6
ZPointRet <- 6.45e-6
AFMcurve <- afmBaselineCorrection(AFMcurve,ZPointApp = ZPointApp,ZPointRet = ZPointRet)
plot(AFMcurve)

# Without providing ZPointApp
AFMcurve <- afmReadJKP("force-save-JPK-3h.txt.gz", path = path.package("afmToolkit"))
AFMcurve <- afmContactPoint(AFMcurve,width = 10,mul1 = 1,mul2 = 20,
                           loessSmooth = FALSE)
AFMcurve <- afmBaselineCorrection(AFMcurve)
plot(AFMcurve)
```

---

`afmContactPoint`  

**Contact point**

Find the contact point in for the Force-Distance curve following the local regression and two thresholds methods described in Microscopy Research and Technique 2013 (see reference).

**Usage**

`afmContactPoint(afmdata, width = 1, mul1, mul2, lagdiff = width, Delta = TRUE, loessSmooth = FALSE)`

**Arguments**

- `afmdata` A Force-Distance curve with the afmdata structure. It should be a list with at least the 'data' field with a data frame of at least 4 columns.
- `width` Width of the window for the local regression (in vector position units)
- `mul1` First multiplier for the first alarm threshold
- `mul2` Second multiplier for the second alarm threshold
- `lagdiff` Lag for estimating the differences in Delta (or slopes) signal. By default it takes the same value as the window with.
- `Delta` Logical. If TRUE, then the statistic for determining the contact point is the differences between two consecutive values of the slope of the local regression line. If FALSE then the slope itself is used.
- `loessSmooth` Logical If TRUE, a loess smoothing (via loess.smooth()) is done prior to the determination of the contact point. The span of the smoothing is 0.05 (5 approach segment).
Value

An `afmdata` class variable which will consist on the original input `afmdata` variable plus a new list named CP with the following fields:

- CP The contact point value.
- icp The position in the array for the contact point value.
- delta The delta signal.
- noise The noise of the delta signal.

References


See Also

`afmDetachPoint`

Examples

```r
path <- path.package("afmToolkit")
data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path)
width <- 20
mul1 <- 1
mul2 <- 10
data <- afmContactPoint(data, width = width, mul1 = mul1, mul2 = mul2)
## Not run:
plot(data, segment = "approach") + geom_vline(xintercept = data$CP$CP, lty = 2)
## End(Not run)
```

afmdata  

AFM data

Description

This function creates an `afmdata` structure, which is a list with at least one field called `data` which is a data frame with valid AFM data, that is, at least 3 variables called "Z", "Force", and "Segment".

Usage

```r
afmdata(data, dstr = "Z", Fstr = "Force", Segstr = "Segment", tstr = "Time",
params = list(SpringConstant = numeric(), curvname = NULL))
```
Arguments

data  A data frame consisting in 3 or 4 columns. A minimum of "Z" (or "distance"), "Force" and "Segment". Optionally a fourth column with "Time" could be added.
dstr  Character string with the possible names for the distance variable.
fstr  Character string with the possible names for the force variable.
segstr  Character string with the possible names for the Segment variable.
tstr  Character string with the possible names for the time variable.
params  A list that may contain parameters describing the F-d curve. At least will contain the \texttt{springconstant} and the \texttt{curvename}, being the former the cantilever spring constant and the latter a F-d curve ID. Function \texttt{afmReadJPK} will try to obtain the spring constant from the file header and the curvename from the data file name.

Value

An object of class \texttt{afmdata}

See Also

\texttt{afmexperiment}

Examples

```r
#Making some artificial data following a L-J 12-6 potential
n <- 1000
z <- seq(from = 9e-3, to = 1e-1, length.out = n)
u0 <- 1e-5
z0 <- 1e-2
Force <- -u0*(12*z0^6/z^7-12*z0^12/z^13)
Segment <- rep("approach",n)
AFMcurve <- afmdata(data.frame(Z = z, Force = Force, Segment = Segment))
plot(AFMcurve)
```

afmDetachPoint  \hspace{1cm} \textit{Detach point}

Description

Find the detach point (or unbinding point) for the Force-Distance curve following the local regression and two thresholds methods described in Microscopy Research and Technique 2013 (see reference).

The procedure is similar to the one used by the \texttt{afmContactPoint()} function for obtaining the contact point.
Usage

afmDetachPoint(afmdata, width=1, mul1, mul2, lagdiff = width, Delta=TRUE, loessSmooth = FALSE)

Arguments

- afmdata: A Force-Distance curve with the afmdata structure. It should be a list with at least the 'data' field with a data frame of at least 4 columns.
- width: Width of the window for the local regression (in vector position units)
- mul1: First multiplier for the first alarm threshold
- mul2: Second multiplier for the second alarm threshold
- lagdiff: Lag for estimating the differences in Delta (or slopes) signal. By default it takes the same value as the window with.
- Delta: Logical. If TRUE, then the statistic for determining the contact point is the differences between two consecutive values of the slope of the local regression line. If FALSE then the slope itself is used.
- loessSmooth: Logical If TRUE, a loess smoothing (via loess.smooth()) is done prior to the determination of the contact point. The span of the smoothing is 0.05 (5 approach segment).

Value

An afmdata class variable which will consist on the original input afmdata variable plus a new list named DP with the following fields:

- DP: The detach point value.
- idp: The position in the array for the detach point value.
- delta: The delta signal.
- noise: The noise of the delta signal.

References


See Also

afmContactPoint

Examples

data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path.package("afmToolkit"))
width <- 10
mul1 <- 2
mul2 <- 40
data <- afmDetachPoint(data, width = width, mul1 = mul1, mul2 = mul2)
Description

Fits a viscoelastic exponential decay in a Force-Relaxation or Creep experiments as described in Nanotechnology 2010 (see references).

Usage

afmExpDecay(afmdata, nexp = 2, tmax = NULL, type = c("CH","CF"), plt = TRUE, ...)

Arguments

afmdata An object of afmdata class with a pause segment and a Time column in the data dataframe.
nexp Number of exponentials in the Prony series to be fitted. Currently only one or two exponentials are supported. Default is 2.
tmax Maximum time considered in the relaxation curve. It defaults to Inf, meaning that the whole pause segment is considered.
type Type of the experiment. Can be either "CH" (Constant Height) for a force-relaxation experiment or "CF" (Constant Force) for a creep experiment. Default is type = "CH".
plt Logical. If TRUE (default) then a plot of the pause segment with the overlay of the fit is shown.
...
Options passed to the nlsM() function from the minpack.nlm package. At least should contain the starting values (start = list(...)) for the Levenberg-Mardquart nonlinear least square method.

Value

An afmdata class variable which will consist on the original input afmdata variable plus a new list named ExpFit with the following fields:
expdecayModel: A nls object returned from nlsM() function.
expdecayFit: The values predicted by the fit, returned from the predict() function.

References

Susana Moreno-Flores, Rafael Benitez, Maria dM Vivanco and Jose Luis Toca-Herrera (2010). "Stress relaxation and creep on living cells with the atomic force microscope: a means to calculate elastic moduli and viscosities of cell components”. Nanotechnology, 21 (44), pp. 445101.
Examples

data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path.package("afmToolkit"))
width <- 20
mul1 <- 1
mul2 <- 10
data <- afmContactPoint(data, width = width, mul1 = mul1, mul2 = mul2)
data <- afmDetachPoint(data, width = width, mul1 = mul1, mul2 = mul2)
data <- afmBaselineCorrection(data)
data <- afmExpDecay(data, nexp = 2, type = "CH")

Description

This function creates an afmexperiment structure, which is as list (or an array) of elements of afmdata class.

Usage

afmexperiment(data, ID=NULL)

Arguments

data A variable of afmdata class, or a list of elements of afmdata class.
ID Character string with the identifier of the data variable or a string array in case data is a list of afmdata variables.

Value

An object of class afmexp.

See Also

afmdata

Examples

dataFolder <- paste(path.package("afmToolkit"), "afmexperiment", sep = "/")
dataFiles <- list.files(dataFolder, pattern = "force", full.names = FALSE)
data <- lapply(dataFiles, afmReadJPK, path = dataFolder)
names(data) <- dataFiles
data <- afmexperiment(data)
plot(data[[1]])
afmExtract

Extract computed parameters from an afmexperiment

Description

Extracts some parameters from an afmexperiment for an easy further analysis.

Usage

afmExtract(afmexperiment, params = list("YM", "AE", "ED"), opt.param = NULL)

Arguments

afmexperiment  Data of afmexperiment class.
params  List of parameters to extract from the data.
opt.param  Optional parameter or factor in the params field of the afmdata list to add to the data extraction.

Value

A data frame with the name of the curve and the corresponding values of the parameters extracted.

Examples

## Not run:
require(dplyr)  # Not really necessary

# Load the data
data(batchExperiment)

# Process the afmexperiment
data <- afmContactPoint(batchExperiment, width = 50, mul1 = 1, mul2 = 10)
data <- afmDetachPoint(data, width = 50, mul1 = 1, mul2 = 10)
data <- afmBaselineCorrection(data)
data <- afmZeroPointsSlope(data)
data <- afmIndentation(data)
data <- afmYoungModulus(data, thickness = 2e-7, params = list(alpha = 22))
data <- afmExpDecay(data, plt = FALSE)
data <- afmAdhesionEnergy(data, mul = 7)

# Extract the values of the parameters obtained in the analysis
afmExpParams <- afmExtract(data, opt.param = "type")

# Plotting the Young's Modulus
afmExpParams[[1]] %>% ggplot(aes(x = type, y = YM)) + geom_boxplot() + ylab("Young's Modulus (Pa)")

## End(Not run)
Description

This function computes the deformation of the sample from the calibrated Force-Distance curve, by substracting $Z$ to the Zero Force Point calculated with afmZeroPointSlope function.

Usage

afmIndentation(afmdata)

Arguments

afmdata   An afmdata object. It should be a valid afmdata object upon which the Contact Point, the baseline correction and the Zero Force Point must have been calculated first (using functions afmContactPoint(), afmBaselineCorrection() and afmZeroPointSlope())

Value

Returns a list with one field:

afmdata: An afmdata class in which a Indentation column is added in the data field.

Examples

data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path.package("afmToolkit"))
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 20)
data <- afmAttachPoint(data, width = 40, mul1 = 3, mul2 = 40)
data <- afmBaselineCorrection(data)
data <- afmZeroPointSlope(data, segment = "approach")
data <- afmIndentation(data)
head(data$data)

afmReadJPK  Read Nanowizard JPK ascii file

Description

Read an ascii JPK file.

Reads an ascii JPK file with one to three headers.

Usage

afmReadJPK(filename, path = "/", FColStr = "Vertical", ZColStr = "Height (measured & smoothed)", tColStr = "Segment Time")
**afmReadJPKFolder**

**Arguments**
- `filename`  String with the name of the jpk file.
- `path`  Path to the folder where the file is.
- `FColStr`  String with a pattern identifying the Force column.
- `ZColStr`  String with a pattern identifying the Z column.
- `tColStr`  String with a pattern identifying the Time column.

**Value**
A list containing a field 'data' which is a data frame

**Examples**
```r
data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path.package("afmToolkit"))
str(data)
```

---

**afmReadJPKFolder**  Read all Nanowizard JPK ascii files in a folder

**Description**
Read all JPK ascii files in a given folder. It searches for all files containing a given pattern (".txt" by default) and uses the `afmReadJPJ` function.

**Usage**
`afmReadJPKFolder(folder, pattern = ".txt", ...)`

**Arguments**
- `folder`  Name of the folder containing the jpk files.
- `pattern`  Pattern that will identify the jpk files (".txt" by default).
- `...`  Other parameters passed to `afmReadJPJ` function.

**Value**
An `afmexperiment` class data structure with all F-d curves.

**Examples**
```r
folder <- paste(path.package("afmToolkit"), "afmexperiment", sep = "/")
data <- afmReadJPKFolder(folder = folder)
str(data)
```
**afmReadVeeco**

**Description**

Read an ascii Veeco file.
Reads an ascii Veeco file with one or two segments.

**Usage**

afmReadVeeco(filename, path ="")

**Arguments**

- **filename** String with the name of the jpk file.
- **path** Path to the folder where the file is.

**Value**

A list containing a field 'data' which is a data frame

**Examples**

data <- afmReadVeeco("veeco_file.txt.gz",path = path.package("afmToolkit"))
str(data)

**afmReadVeecoFolder**

**Description**

Read all Veeco ascii files in a given folder. It searches for all files containing a given pattern (".txt" by default) and uses the afmReadVeeco function.

**Usage**

afmReadVeecoFolder(folder, pattern = "\.txt")

**Arguments**

- **folder** Name of the folder containing the Veeco files.
- **pattern** Pattern that will identify the Veeco files (".txt" by default).

**Value**

An afmexperiment class data structure with all F-d curves.
Examples

```r
folder <- paste(path.package("afmToolkit"), "veecoFolder", sep = "/")
data <- afmReadVeecoFolder(folder = folder)
str(data)
```

Description

This function computes the Young’s Modulus of the sample from the approach curve using Hertz’s contact model for a pyramidal tip.

Usage

```r
afmYoungModulus(afmdata, thickness = NULL, model = "Hertz", geometry =
c("pyramid","paraboloid"), silent = TRUE, params)
```

Arguments

- `afmdata`: An `afmdata` object. It should be a valid `afmdata` object upon which the Contact Point, the baseline correction and the Zero Force Point and the Indentation must have been calculated first (using functions `afmContactPoint()`, `afmBaselineCorrection()`, `afmZeroPointSlope()`, and `afmIndentation()`).
- `thickness`: Thickness (in m) of the surface. The Force - Indentation fit will be done for values of the Indentation variable smaller than the thickness. If no value is given, it will be done for all values in the curve for which the Indentation is negative.
- `model`: Contact mechanics model to be used. Currently only Hertz’s pure elastic model is available.
- `geometry`: Geometry of the tip. Currently only pyramidal (default) and paraboloid geometries are implemented.
- `silent`: Logical value. If FALSE it prints the fit model summary (via `summary.lm()`). Default value is TRUE
- `params`: A list containing different parameters of the model: e.g. nu (Poisson’s ratio) or alpha (internal angle, in degrees, of the pyramidal tip) or R (tip radius, in the paraboloid geometry)

Value

An `afmdata` class variable which will consist on the original input `afmdata` variable plus a new list named `YoungModulus` with the following fields:
- `YoungModulus`: The Young’s modulus value (in Pa).
- `fitYm`: The Force vs Indentation^2 fit as an `lm` object.
- `fitdata`: The subset of the data used in the fit.
Examples

data <- afmReadJPK("force-save-JPK-2h.txt.gz", path = path.package("afmToolkit"))
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 20)
data <- afmDetachPoint(data, width = 40, mul1 = 3, mul2 = 40)
data <- afmBaselineCorrection(data)
data <- afmZeroPointslope(data, segment = "approach")
data <- afmIndentation(data)
data <- afmYoungModulus(data, thickness = 1e-8, params = list(alpha = 22),
silent = TRUE)
print(data$YoungModulus$YoungModulus)

afmZeroPointslope Zero Force Point and Slope

Description

This function finds the point of zero force (real contact point) and the slope of the contact part of
the Force-Distance curve.

Usage

afmZeroPointslope(afmdata, fstar = 0, segment = c("approach", "retract"))

Arguments

afmdata An afmdata object. It should be a valid afmdata object upon which the Contact
Point and the baseline correction must have been calculated first (using functions
afmContactPoint() and afmBaselineCorrection())

fstar Value such that fstar * sd is to be considered as zero Force, where sd is the
standard deviation of Force at the basline. It takes fstar = 0 as default value,
meaning that zero force is actually zero.

segment The segment on which everything is calculated.

Value

An afmdata class variable which will consist on the original input afmdata variable plus a new list
named Slopes with the following fields: z0Point: Point of zero force. Slope: Slope of the best fit
line in the contact part of the Force-Distance curve.

Examples

data <- afmReadJPK("force-save-JPK-2h.txt.gz", path = path.package("afmToolkit"))
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 20)
data <- afmDetachPoint(data, width = 40, mul1 = 3, mul2 = 40)
data <- afmBaselineCorrection(data)
data <- afmZeroPointslope(data, segment = "approach")
# Not run:
plot(data, segment = "approach") + geom_vline(xintercept = data$Slopes$z0Point, lty = 2)
## Append to an afmdata list

### Description

This function appends a list to an existing afmdata structure. It is used internally by several afm* functions when attaching the results to the input afmdata variable. This function should not be used directly unless by experienced users.

### Usage

```r
append.afmdata(afmdata, x, name = NULL)
```

### Arguments

- **afmdata**: The afmdata to which the new list is going to be joined.
- **x**: A list to be appended.
- **name**: The name of new field of the resulting afmdata object. If none is given, it is the same as `x`.

### Value

The new list of class afmdata

---

## Example of an afmexperiment data class.

### Description

An afmexperiment list containing 14 afmdata Force-distance experiments. Each experiment has three segments ("approach", "pause" and "retract") and they are divided in two groups depending on the covering of the sample ("CHI" for Chitosan, and "PAH" for Polyallylamine hydrochloride).

### Usage

```r
batchExperiment
```
is.afmdata

Description

Checks whether an R object is an afmdata or not.

Usage

is.afmdata(x)

Arguments

x Any R object.

Value

Returns TRUE if its argument is an afmdata (that is, has "afmdata" amongst its classes) and FALSE otherwise.
is.afmexperiment | Afmexperiment check.

Description
Checks whether an R object is an afmexperiment or not.

Usage
is.afmexperiment(x)

Arguments

- x | Any R object.

Value
Returns TRUE if its argument is an afmdata (that is, has "afmexperiment" amongst its classes) and FALSE otherwise.

plot.afmdata | Plot an afmdata object

Description
Plots an afmdata object.

Usage
## S3 method for class 'afmdata'
plot(x, y = NULL, vs = "Z", segment = "all", ...)

Arguments

- x | An object of afmdata class.
- y | Variable added for compatibility with plot.
- vs | The variable for the x-axis. May take the values "Time" or "Z". It defaults to "Z", plotting thus a Force-Distance curve. If vs is set to "Time", then it plots a Force-Time curve.
- segment | The segment of the curve to be plotted. If segment = "all" then all segments of the curve are plotted. Possible values are: "approach", "pause", "retract" and "all".
- ... | Additional parameters to be passed to the ggplot functions.
summary.afmdata

Examples

# Loading the data
path <- path.package("afmToolkit")
data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path)
# Standard plot (out of the box)
plot(data)
# Computing the contact and detach points
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 10)
data <- afmDetachPoint(data, width = 40, mul1 = 3, mul2 = 20)
# Making the baseline correction
data <- afmBaselineCorrection(data)
# Plot once the baseline correction is done
plot(data)
# Plotting only retract segment
plot(data, segment = "retract")
# Plotting the pause segment: Force vs Time
plot(data, segment = "pause", vs = "Time")

summary.afmdata

Summary of an afmdata class object.

Description

This function summarises the main features of an afmdata object and, optionally, plots all segments available with all parameters estimated.

Usage

## S3 method for class 'afmdata'
summary(object, plt = TRUE, ...)

Arguments

object An object of afmdata class.
plt Logical variable. If TRUE plots all available segments with all available data.
... Additional arguments (for compatibility with summary)

Examples

## Not run: path <- path.package("afmToolkit")
data <- afmReadJPK("force-save-JPK-3h.txt.gz", path = path)
data <- afmContactPoint(data, width = 20, mul1 = 1, mul2 = 10)
data <- afmDetachPoint(data, width = 40, mul1 = 3, mul2 = 20)
data <- afmBaselineCorrection(data)
data <- afmAdhesionEnergy(data, width = 20, mul1 = 10)
data <- afmZeroPointSlope(data, segment = "approach")
data <- afmIndentation(data)
data <- afmYoungModulus(data, thickness = 1e-7, params = list(alpha = 22),
Description

This is an internal function used by the \texttt{afmContactPoint} and \texttt{afmDetachPoint} functions. It computes the slopes of a linear fit to the data in a window of a given radius. This function should not be used directly unless by experienced users.

Usage

\begin{verbatim}
windowedFit(X, width)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{X} \hspace{1cm} Least squares matrix on the form [1 \ z \ Force], according to input parameters in function \texttt{lm.fit}
\item \texttt{width} \hspace{1cm} Width of the window for the local regression (in vector position units)
\end{itemize}

Value

\begin{verbatim}
out \hspace{1cm} A vector of length \texttt{nrow(X)-2*width}, containing with the slopes of the fits.
\end{verbatim}

Examples

\begin{verbatim}
n <- 100
x <- seq(0,2*pi,length.out = n)
y <- sin(x)+0.1*rnorm(n)
X <- matrix(c(rep(1,n),x,y),nrow = n,ncol = 3)
width <- 5
b <- windowedFit(X,width)
plot(x[(width+1):(n-width)],b,xlab = "x",ylab = "y",type = "l")
lines(x,y,col = "red")
legend("bottomleft",c("Slopes","Signal"),col = c(1,2),lty = 1)
\end{verbatim}
Index

*Topic **datasets**
- batchExperiment, 16

**afmAdhesionEnergy**, 2
**afmBaselineCorrection**, 3
**afmContactPoint**, 4, 7
**afmdata**, 5, 9
**afmDetachPoint**, 5, 6
**afmExpDecay**, 8
**afmexperiment**, 6, 9
**afmExtract**, 10
**afmIndentation**, 11
**afmReadJPK**, 11
**afmReadJPKFolder**, 12
**afmReadVeeco**, 13
**afmReadVeecoFolder**, 13
**afmYoungModulus**, 14
**afmZeroPointSlope**, 15
**append.afmdata**, 16
**batchExperiment**, 16
**is.afmdata**, 17
**is.afmexperiment**, 18
**plot.afmdata**, 18
**summary.afmdata**, 19
**windowedFit**, 20