Package 'rheologyEvaluation'

March 10, 2021

Type Package

Title rheologyEvaluation

Version 1.1

Date 2018-10-17

Author Thomas Braschler

Maintainer Thomas Braschler <thomas.braschler@gmail.com>

Description This R package contains utility functions to read and analyze Haake Rheowin text files, particularly for particulate suspensions.

License GPL-3 — file LICENSE

Depends R (¿= 3.5.0)

LazyLoad yes

R topics documented:

`rheologyEvaluation-package`
rheologyEvaluation

Description

Some utility functions for reading and analyzing rheology files from Rheowin

Details

Package:	rheologyEvaluation
Type:	Package
Version:	1.0
Date:	2018-10-17
License:	What license is it under?
LazyLoad:	yes

Author(s)

Thomas Braschler

Maintainer: thomas.braschler@gmail.com

`complete_sweeps` *complete_sweeps*

Description

Completes existing sweeps by interpolation so that the sweeps all have a complete set of x-values. This makes graphing and testing at specific x-values easier, but one needs to bear in mind that this function works with interpolation

Usage

```
complete_sweeps(theSweeps,abscissa="tau_Pa",sweep_identifier_column="file",columns_to_complete=c
```

Arguments

`theSweeps`	Rheological sweeps in R data.frame format, that is concatened in a long table with a column as indicated by the argument `sweep_identifier_column` where the individual sweeps are uniquely identified. In addition, this argument needs to contain at least: the `abscissa` column, the columns enumarated in the `columns_to_complete` argument.
`abscissa`	x-values; this is often, but not necessarily, the controlled parameter during the sweep, as for example the shear stress "tau_Pa".
`sweep_identifier_column`	There are many rows per sweep in general, this column identifies which ones belong together to a given sweep.
`columns_to_complete`	List of columns which should be interpolated so that for all individual sweeps, values at the entire collection of x-values are available.
`scale_with_abscissa`	For extrapolation at low x-values (i.e. below the minimum value of the sweep at hand): for columns listed in `scale_with_abscissa`, proportionality with the x-values is assumed, and extrapolation is from the value at the minimum x-value for the sweep concerned by proportionality with the x-values. Use this only if you are sure from a priori knowledge that proportionality to the abscissa value is better than linear extrapolation in estimation of the missing y-values.

Details

This function anticipates typical rheology plots where the G'/G" values either vary slowly on a logarithmic scale or show constant slopes in the extreme regions. This is why the interpolation is linear in the log-log plot. Use this function cautiously when the curves are following the general same trend but values at all available x-coordinates for all curves are required but not available through measurement. An example plots using some measured rather than imposed quantity on the x-axis, making it impossible to predict the actual x-values before measurement.

Value

A data.frame with the same columns as `theSweeps` but potentially more lines if it was necessary to complete the dataframe to have entries for all the x-values for all the sweeps.

Author(s)

Thomas Braschler

Examples

```
theSweeps=data.frame(file=c(rep("A.txt",7),rep("B.txt",4),rep("C.txt",10)),tau_Pa=c(1,2,4,10,20,40,100,1,10
theSweepsCompleted = complete_sweeps(theSweeps,abscissa="tau_Pa",sweep_identifier_column="file",columns_to_
plot(Gprime_Pa ~ tau_Pa, theSweepsCompleted[theSweepsCompleted$file=="A.txt",],log="xy",type="b",main="compl
lines(Gprime_Pa ~ tau_Pa, theSweepsCompleted[theSweepsCompleted$file=="B.txt",],type="b",col="red")
lines(Gprime_Pa ~ tau_Pa, theSweepsCompleted[theSweepsCompleted$file=="C.txt",],type="b",col="green")
lines(Gprime_Pa ~ tau_Pa, theSweeps[theSweeps$file=="A.txt",],type="p",col="black",pch=21,bg="black")
lines(Gprime_Pa ~ tau_Pa, theSweeps[theSweeps$file=="B.txt",],type="p",col="red",pch=21,bg="red")
lines(Gprime_Pa ~ tau_Pa, theSweeps[theSweeps$file=="C.txt",],type="p",col="green",pch=21,bg="green")
legend("bottomleft",legend=c("A.txt","B.txt","C.txt", "Full symbols original data, empty interpolated"),pch=c
```

`evaluateYield` *evaluateYield*

Description

Evaluates the yield point (stress and strain) from oscillatory rheology sweeps (stress or strain sweep). This is done by finding the intersection point between the G' and G" curves, based on the predominantly solid nature when G'¿G" and inversely predominantly liquid nature when G'¡G" (Schramm, G. A, 1994)

Usage

```
evaluateYield(sweep, Gprime_column = "Gprime_Pa", Gprimeprime_column = "Gprimeprime_Pa",
   abscissa = "tau_Pa", additional_columns_to_be_interpolated = "Gamma_in_percent")
```

Arguments

sweep	Sweep information, needs to contain at least the columns given by the Gprime_column, the Gprimeprime_column and the abscissa column
Gprime_column	Name of the column containing the G' values
Gprimeprime_column	Name of the column containing the G" values
abscissa	Name of the column containing the abscissa of the sweep (the variable that was sweeped to create the oscillatory sweep
additional_columns_to_be_interpolated	Name of additional columns that should be interpolated to find its value at the yield point. The interpolation will be done to exactly the place as for the Gprime_column column (respectively Gprimeprime_column column, at the yield point they are by definition equal), that is using the same weights between the last solid and first liquid measurement point.

Value

A vector of two entries describing the strain and stress of the yield point, and additional characteristics when additional_columns_to_be_interpolated is not empty.

Author(s)

Thomas Braschler

References

Schramm, G. A Practical Approach to Rheology and Rheometry. (Gebrueder HAAKE GmbH, 1994).

Examples

```
# Loading and plotting of the data
data(rheology_sampleData)
# Evaluation in linear scale
yieldPointLinear=evaluateYield(rheology_sampleData,Gprime_column="Gprime_Pa",Gprimeprime_column="Gprimepri
yieldPointLinear

#Evaluation in log scale
rheology_sampleData$log_Gprime_Pa = log(rheology_sampleData$Gprime_Pa)
rheology_sampleData$log_Gprimeprime_Pa = log(rheology_sampleData$Gprimeprime_Pa)
rheology_sampleData$log_tau_Pa = log(rheology_sampleData$tau_Pa)

yieldPointLog=evaluateYield(rheology_sampleData,Gprime_column="log_Gprime_Pa",Gprimeprime_column="log_Gprir
yieldPointLog["Gprime_Pa"] = exp(yieldPointLog["log_Gprime_Pa"])
yieldPointLog["tau_Pa"] = exp(yieldPointLog["log_tau_Pa"])

#In log scale, the yield point is the intersection on the log-log G'-shear plot
plot(Gprime_Pa ~ tau_Pa, rheology_sampleData,log="xy",type="b",main="rheology_sampleData")
lines(Gprimeprime_Pa ~ tau_Pa, rheology_sampleData,type="l")
lines(yieldPointLog["tau_Pa"],yieldPointLog["Gprime_Pa"],type="p",pch=21,cex=1.5,col="red",bg="red")
legend("bottomleft",legend=c("Elastic modulus G'","Viscous modulus G''", "Yield point"),lty=c(-1,1,-1),pch=c
```

`fitPolynomialModelGprimeConcentration` *fitPolynomialModelGprimeConcentration*

Description

Fits a polynomial model of the form `Gprime=A*(x-c0)^n` to experimental G' (Gprime) data. Here, x and Gprime are known (measured), while we try to find A, c0 and n.

Usage

```
fitPolynomialModelGprimeConcentration(Gprime,polymer_conc_mgmL,n=2.5, do_full_fit=TRUE)
```

Arguments

`Gprime`	Measured G' values
`polymer_conc_mgmL`	Known concentration of the suspension, in terms of polymer concentration, typically in mg/mL
`n`	Initial guess for the exponent in the polynomial model
`do_full_fit`	At the end of the fitting procedure, do you want to attempt to fit simultaneously the three parameters `A`, `c0` and `n` after initial fitting? This can provide the best fit, but often does not converge.

Details

Carries out least squares fitting by using nls; depending on the data and starting values, this can be finicky. Check the values that you obtain.

Value

List with named entries for `A`, `n` and `c0`

Author(s)

Thomas Braschler

Examples

```
x=c(30,35,40,45,50)
Gprime=c(10,400,1500,5000,11000)
plot(x,Gprime,log="y",xlab="Polymer concentration [mg/mL]",ylab="G'")
coeffs=fitPolynomialModelGprimeConcentration(Gprime=Gprime,polymer_conc_mgmL=x,n=2.5, do_full_fit=FALSE)
x_theory=seq(from=0,to=50, length.out=200)
lines(x_theory,polynomial_model(x=x_theory,A=coeffs[["A"]],c0=coeffs[["c0"]],n=coeffs[["n"]]))
```

logpolynomial_model *logpolynomial_model*

Description

Evaluation of a polynomial model with an offset (polynomial_model) for the G' values; returns the log value for x¿c0 and penalty negative value for x¡=c0. The idea is that if the polymer concentration is smaller than the critical concentration c0, the suspension is liquid and we return G'=0, otherwise it's a polynomial function of x-c0 with exponent n and coefficient A. The logarithmic form is for fitting purposes

Usage

```
logpolynomial_model(x, A, c0, n,penalty_factor_below_offset=10)
```

Arguments

x	The variable for the polynomial model
A	Proportionality constant for the polynomial relation
c0	Offset for the polynomial model
n	Exponent for the polynomial relation
penalty_factor_below_offset	Factor for being able to return a value even when x is below c0

Value

log(A*(x-c0)^n) for x¿c0, penalty_factor_below_offset*(A*(-x-c0)^n) for x¡c0

Author(s)

Thomas Braschler

Examples

```
x=seq(from=0,to=50,length.out=200)
logGprimeModel = logpolynomial_model(x,1,20,3,penalty_factor_below_offset=NA)
plot(x,exp(logGprimeModel),type="b",xlab="Polymer concentration",log="y",ylab="Model G' value")
```

polynomial_model *polynomial_model*

Description

Evaluation of a polynomial model with an offset for the G' values. The idea is that if the polymer concentration is smaller than the critical concentration c0, the suspension is liquid and we return G'=0, otherwise it's a polynomial function of x-c0 with exponent n and coefficient A

Usage

```
polynomial_model(x, A, c0, n)
```

Arguments

x	The variable for the polynomial model
A	Proportionality constant for the polynomial relation
c0	Offset for the polynomial model
n	Exponent for the polynomial relation

Value

A*(x-c0)^n

Author(s)

Thomas Braschler

Examples

```
x=seq(from=0,to=50,length.out=200)
GprimeModel = polynomial_model(x,1,20,3)
plot(x,GprimeModel,type="b",xlab="Polymer concentration",log="y",ylab="Model G' value")
```

---

read_Haake_Rheowin_text_file

read_Haake_Rheowin_text_file

Description

Reads a text file expored from Rheowin into R

Usage

```
read_Haake_Rheowin_text_file(file,use_C_locale=TRUE,colname_info=
list("tau_Pa"=c("Tau.en.Pa","X..en.Pa"),
"Gprime_Pa"=c("G..en.Pa"),
"Gprimeprime_Pa"=c("X.G...en.Pa.","G..en.Pa.1"),
"Phase_angle_degrees"=c("X..en...1","Da.en.."),
"Gamma_in_percent"=c("X..en..","Ga.en.."),
"f_Hz"=c("f.en.Hz"),
"t_s"=c("t.en.s"),
"t_seg_s"="t_seg.en.s"),...)
```

Arguments

file	The path to the file to be read
use_C_locale	Boolean. If TRUE, forces to use a standard C locale to standardize encoding, which otherwise might be set differently by default on different systems. Uses Sys.setlocale internally for this purpose. This function makes an effort at setting the encoding back to your locale, although this may not work correctly in all circumstances.
colname_info	List indicating how to handle the column names read from the Rheowin text file. The names of this list will be the output column names, while the elements indicates candidates to be looked for in the text file. They will be tried in order, if several are matching, the first match will be used.
...	Arguments to be passed to read.table, used internally for reading the text. Particular attention needs to be payed to the lines to be skipped since there is typically a text block in text files exported from Rheowin in addition to the desired data.

Value

A dataframe with names identical to the names of the colname_info argument. If the corresponding column could not be found, NA entries are provided.

Author(s)

Thomas Braschler

Examples

```
# Access path for reading the sample file
path=system.file("sampleData",package="rheologyEvaluation")
file="Restylane_syringe_1.txt"
colname_info=
list("tau_Pa"=c("Tau.en.Pa","X..en.Pa"),
"Gprimeprime_Pa"=c("G..en.Pa"),
"Gprime_Pa"=c("X.G...en.Pa.","G..en.Pa.1"),
"Phase_angle_degrees"=c("X..en...1","Da.en.."),
"Gamma_in_percent"=c("X..en..","Ga.en.."),
"f_Hz"=c("f.en.Hz"),
"t_s"=c("t.en.s"),
"t_seg_s"="t_seg.en.s")
rheology_sampleData=read_Haake_Rheowin_text_file(file=paste(path,file,sep="/"),colname_info=colname_info,he
plot(Gprime_Pa ~tau_Pa, rheology_sampleData,log="xy") # At the beginning, there is a pre-conditioning step, thi
```

---

rheology_sampleData *Sample rheological stress sweep*

Description

Sample rheological stress sweep, obtained on a commercial sample of Restylane Lidocaine, on a Rheostress 100 instrument, plate-plate geometry, 0.2Hz oscillatory sweep. We used a pre-conditioning step (increase of shear stress at low level followed by decrease), which is removed from the actual sweep

Usage

```
data(rheology_sampleData)
```

Format

Constains a single variable `rheology_sampleData`:

`rheology_sampleData` data.frame with 8 columns: `tau_Pa` indicates the shear stress, `Gprimeprime_Pa` the viscous modulus G", `Gprime_Pa` the elastic modulus G', `Phase_angle_degrees` the phase angle determined by G' and G", `Gamma_in_percent` the shear deformation in percent, `f_Hz` the excitation frequency in Hz, `t_s` the time from the beginning of the experiment, and `t_seg_s` the time in the current acquisition segment.

Examples

```
# Generation of the data
path=system.file("sampleData",package="rheologyEvaluation")
file="Restylane_syringe_1.txt"
colname_info=
list("tau_Pa"=c("Tau.en.Pa","X..en.Pa"),
"Gprimeprime_Pa"=c("G..en.Pa"),
"Gprime_Pa"=c("X.G...en.Pa.","G..en.Pa.1"),
"Phase_angle_degrees"=c("X..en...1","Da.en.."),
"Gamma_in_percent"=c("X..en..","Ga.en.."),
"f_Hz"=c("f.en.Hz"),
"t_s"=c("t.en.s"),
"t_seg_s"="t_seg.en.s")
rheology_sampleData=read_Haake_Rheowin_text_file(file=paste(path,file,sep="/"),colname_info=colname_info,he

# There is a preconditioning step consisting in ramping up and down and up the applied shear stress in this sampl
rheology_sampleData=rheology_sampleData[19:length(rheology_sampleData$tau_Pa),]


# Loading and plotting of the data
data(rheology_sampleData)
plot(Gprime_Pa ~ tau_Pa, rheology_sampleData,log="xy",type="b",main="rheology_sampleData")
lines(Gprimeprime_Pa ~ tau_Pa, rheology_sampleData,type="l")
legend("bottomleft",legend=c("Elastic modulus G'","Viscous modulus G''"),lty=c(-1,1),pch=c(21,-1))
```

---

wall_fraction *wall_fraction*

Description

Function to estimate the wall fraction from the polymer concentration. Needs two constants, the wall concentration and the osmotic coefficient gamma

Usage

```
wall_fraction(polymer_concentration=10,wall_concentration=70,gamma=0.15)
```

Arguments

polymer_concentration	Polymer concentration in mg/mL
wall_concentration	Nominal wall concentration
gamma	Compressibility constant of the walls

Value

Vector of the same length as polymer_concentration, indicating the theoretical wall fraction with the coefficients wall_fraction and gamma

Author(s)

Thomas Braschler

References

Pellet, C. and M. Cloitre, The glass and jamming transitions of soft polyelectrolyte microgel suspensions. Soft Matter, 2016. 12(16): p. 3710-20, see also Supplementary 8 of the manuscript "An injectable meta-biomaterial", particularly for values of the wall concentration and gamma fraction

Examples

```
conc=0:200
uncompressed_wall_concentration=70 # uncompressed wall concentration spontaneously adapted when the polymer pa
plot(conc,wall_fraction(conc,wall_concentration=uncompressed_wall_concentration),xlab="Total polymer concen
lines(c(0,200),c(0,200)/uncompressed_wall_concentration,type="l",col="red")
legend("bottomright",legend=c("Pellet & Cloitre","uncompressed wall material"),lty=c(1,1),pch=c(1,-1),col=c(
```

Index