pyvisco

Martin Springer

Jan 05, 2026

CONTENTS

Pyvisco is a Python library that supports the identification of Prony series parameters in Generalized Maxwell models describing linear viscoelastic materials.

The source code of pyvisco is hosted at GitHub. The API documentation is hosted at ReadtheDocs. The web application can be accessed through either Heroku or Binder.

CHAPTER

ONE

CONTENTS

1.1 pyvisco

Pyvisco is a Python library that supports the identification of Prony series parameters for Generalized Maxwell models describing linear viscoelastic materials.

1.1.1 Overview

The mechanical response of linear viscoelastic materials is often described with Generalized Maxwell models. The necessary material model parameters are typically identified by fitting a Prony series to the experimental measurement data in either the frequency-domain (via Dynamic Mechanical Thermal Analysis) or time-domain (via relaxation measurements). Pyvisco performs the necessary data processing of the experimental measurements, mathematical operations, and curve-fitting routines to identify the Prony series parameters. The experimental data can be provided as raw measurement sets at different temperatures or as pre-processed master curves.

- If raw measurement data are provided, the time-temperature superposition principle is applied to create a master curve and obtain the shift functions prior to the Prony series parameters identification.
- If master curves are provided, the shift procedure can be skipped, and the Prony series parameters identified directly.

An optional minimization routine is provided to reduce the number of Prony elements. This routine is helpful for Finite Element simulations where reducing the computational complexity of the linear viscoelastic material models can shorten the simulation time.

1.1.2 Usage

The easiest way of using pyvisco is through an interactive Jupyter notebook that provides a graphical user interface to upload the experimental data, perform the curve fitting procedure, and download the obtained Prony series parameters. Currently, the Jupyter notebook is rendered with voila and can be accessed through binder. Click the link below to start the web application.

Alternatively, pyvisco can be installed automatically into Python from PyPI using the command line:

```
pip install pyvisco
```

A full API documentation is available at: .. image:: https://readthedocs.org/projects/pyvisco/badge/?version=latest

target
https://pyvisco.readthedocs.io/en/latest/?badge=latest

alt
Documentation Status

Additionally, the verification subfolder contains example Jupyter notebooks on how to use the library.

1.1.3 Verification

The Python implementation was verified by comparing the obtained Prony series parmaters with the curve fitting routine implemented in the commercial software package ANSYS APDL 2020 R1. Jupyter notebooks showcasing the comparision and supplementary files can be found in the verification subfolder.

1.1.4 Citation

If you are using pyvisco in your published work, please use the following along with the version number and the specific DOI coresponding to that version from Zenodo: .. image:: https://zenodo.org/badge/473711699.svg

target
https://zenodo.org/badge/latestdoi/473711699

alt
DOI

.

APA

```
Springer, Martin (2022). PYVISCO: A Python library for identifying Prony series␣
↪parameters of linear viscoelastic materials (Version {insert version}) [Computer␣
↪software]. doi:{insert DOI}
```

BibTeX

```
@software{Springer_pyvisco_2022,
  author = {Springer, Martin},
  doi = {insert DOI},
  title = {{PYVISCO: A Python library for identifying Prony series parameters of linear␣
↪viscoelastic materials}},
  url = {https://github.com/NREL/pyvisco},
  version = {insert version},
  year = {2022}
}
```

1.2 API Reference

Collection of submodules to identify Prony series parameters of linear viscoelastic materials from measurements in either the time (relaxation tests) or frequency domain (DMTA).

load	Collection of functions to load measurement data and prepare pandas dataframes for further processing.
master	Collection of functions to prepare the master curve for the identification of the Prony series parameters.
opt	Collection of functions to minimize the number of Prony series terms used in the Generalized Maxwell model.
out	Collection of functions to write pandas dataframes to files or buffer.
prony	Collection of function to pre-process the master curve and perform the Prony series parameter identification.
shift	Collection of functions to apply the time-temperature superposition principle to create a master curve from measurements performed at different temperatures.
styles	Collection of function to define figure style and html styles for Jupyter Notebooks.
verify	Collection of function to compare and verify the Python implementation within this module with the curve fitting routine of Ansys APDL 2021 R1.
inter	Collection of classes and functions to create a graphical user interface based on ipywidgets and Jupyter notebook.

1.2.1 pyvisco.load

Collection of functions to load measurement data and prepare pandas dataframes for further processing.

Functions

Eplexor_master(data, modul)	Load master curve data from an EPLEXOR excel file.
Eplexor_raw(data, modul)	Load raw measurement data from an EPLEXOR Excel file.
check_units(units[, modul])	Check that the input file units conform with the conventions used in this modul.
conventions(modul)	Create dictionary with unit conventions for expected physical quantities.
file(path)	Read data from file.
get_sets(df_raw[, num])	Group raw data into measurement sets conducted at the same temperature.
get_units(units, modul, domain[, Eplexor])	Get the physical units based on the measurement data loading condition (tensile or shear) and domain (time or frequency).
prep_csv(data)	Load csv files into pandas dataframe and prepare data.
prep_excel(data)	Load Excel files into pandas dataframe and prepare data.
user_master(data, domain, RefT, modul)	Load master curve data from csv file.
user_raw(data, domain, modul)	Load raw measurement data from csv file.
user_shift(data_shift)	Load user provided shift factors from csv file.

pyvisco.load.**conventions**(*modul*)

Create dictionary with unit conventions for expected physical quantities.

Parameters
modul (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Returns
conv – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Return type
dict of {str : str}

Notes

Tensile moduli are denoted as 'E' and shear moduli are denoted as 'G'. Only the tensile moduli are summarized in the table below. For shear modulus data, replace 'E' with 'G', e.g. 'E_relax' -> 'G_relax'.

Physical quantity	Variable	Unit
Relaxation modulus:	E_relax	[Pa, kPa, MPa, GPa]
Storage modulus:	E_stor	[Pa, kPa, MPa, GPa]
Loss modulus:	E_loss	[Pa, kPa, MPa, GPa]
Complex modulus:	E_comp	[Pa, kPa, MPa, GPa]
Loss factor:	tan_del	•
Instantaneous modulus:	E_0	[Pa, kPa, MPa, GPa]
Equilibrium modulus:	E_inf	[Pa, kPa, MPa, GPa]
Angular frequency:	omega	rad/s
Frequency:	f	Hz
Time:	t	s
Temperature:	T	°C
Relaxation times:	tau_i	s
Relaxation moduli:	E_i	[Pa, kPa, MPa, GPa]
Norm. relaxation moduli:	alpha_i	•
Shift factor:	log_aT	•

pyvisco.load.**file**(*path*)

Read data from file.

Parameters
path (*str*) – Filepath to the file that is being read.

Returns
data – File content.

Return type
bytes

Notes

This function is included to simplify the file upload within the interactive module where graphical dashboards are hosted on a webserver.

pyvisco.load.**prep_csv**(*data*)

Load csv files into pandas dataframe and prepare data.

The function removes NANs from the input data, and identifies the names of the pysical quantities and units from the file header. The file header must consit of two rows. The first row provides the name of the physical quantity and the second row provides the corresponding physical unit.

Parameters
data (*bytes*) – CSV file content to be loaded into a pandas.DataFrame.

Returns

- **df** (*pandas.DataFrame*) – Contains the csv file data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

See also:

load.file
Returns bytes object to be used as input parameter *data*.

pyvisco.load.**prep_excel**(*data*)

Load Excel files into pandas dataframe and prepare data.

The function removes NANs from the input data, and identifies the names of the pysical quantities and units from the file header. The file header must consit of two rows. The first row provides the name of the physical quantity and the second row provides the corresponding physical unit.

Parameters
data (*bytes*) – Excel file content to be loaded into a pandas.DataFrame.

Returns

- **df** (*pandas.DataFrame*) – Contains the Excel file data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

See also:

load.file
Returns bytes object to be used as input parameter *data*.

pyvisco.load.**Eplexor_raw**(*data*, *modul*)

Load raw measurement data from an EPLEXOR Excel file.

This function loads raw Dynamic Mechanical Thermal Analysis (DMTA) measurement files created by a Netzsch Gabo DMA EPLEXOR. Use the *Excel Export!* feature of the Eplexor software with the default template to create the Excel file. The column headers are renamed to follow the conventions used in this modul. The names of the physical quantities and units are checked against the conventions used in this module. the individual measurements at different temperatures are grouped into data sets based on the frequency range of the individual measurement sets.

Parameters

- **data** (*bytes*) – Excel file content loaded into pandas.DataFrame.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Returns

- **df_raw** (*pandas.DataFrame*) – Contains the processed raw measurement data.
- **arr_RefT** (*pandas.Series*) – Contains the temperature levels of the measurement sets.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Raises

KeyError – If the file header variable names or units do not follow the conventions used in this module.

See also:

`load.conventions`
Summarizes conventions for variable names and units.

`load.file`
Returns bytes object to be used as parameter *data*.

`load.check_units`
Raises KeyError if units don't comply with conventions.

pyvisco.load.**user_raw**(*data*, *domain*, *modul*)

Load raw measurement data from csv file.

This function loads raw data of viscoelastic material characterizations conducted at one or more temperature levels. Either tensile or shear modulus data from characterizations performed in either the frequency or time domain are accepted. The file header must consit of two rows. The first row provides the name of the physical quantity and the second row provides the corresponding physical unit. The column headers are checked against the conventions used in this modul. Details on the variable names, units, and file header names are provided in the Notes section below.

Parameters

- **data** (*bytes*) – Excel file content.
- **domain** (*{'freq', 'time'}*) – Defines wether frequency domain ('freq') or time domain ('time') input data are provided.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Returns

- **df_raw** (*pandas.DataFrame*) – Contains the preprocessed raw measurement data.
- **arr_RefT** (*pandas.Series*) – Contains the temperature levels of the measurement sets.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Raises

KeyError – If the file header variable names or units do not follow the conventions used in this module.

See also:

`load.conventions`
Summarizes conventions for variable names and units.

load.file
Returns bytes object to be used as parameter *data*.

load.check_units
Raises KeyError if units don't comply with conventions.

Notes

Dependent on the performed material characterization, either tensile or shear modulus values in either the time or frequency domain must be provided.

Tensile moduli are denoted as 'E' and shear moduli are denoted as 'G'. Frequency domain data are provided in Hertz: [f] = Hz Time domain data are provided in seconds: [t] = s Temperature levels are provided in Celsius: [T] = C Measurement set identifiers are provided by the column *Set*. In Set, all measurement points at the same temperature level are marked with the same number, e.g. 0, for the first measurement set. The first measurement Set (0) represents the coldest temperature followed by the second set (1) at the next higher temperature level and so forth.

Example input files can be found here: (TBD)

Various examples for file headers:

Domain	Row	Tensile Modulus	Shear Modulus
Frequency	1	f, E_stor, E_loss, T, Set	f, G_stor, G_loss, T, Set
Frequency	2	Hz, MPa, MPa, C, -	Hz, GPa, GPa, C, -
Time	1	t, E_relax, T, Set	t, G_relax, T, Set
Time	2	s, MPa, C, -	s, GPa, C, -

pyvisco.load.**Eplexor_master**(*data*, *modul*)

Load master curve data from an EPLEXOR excel file.

This function loads master curve data from an Excel file created with the EPLEXOR software. Use the *Excel Export!* feature of the Eplexor software with the default template to create the Excel file. The column headers are renamed to follow the conventions used in this modul. The names of the physical quantities and units are checked against the conventions used in this module.The modulus data, shift factors, and WLF shift function parameters are extracted from the Excel file.

Parameters

- **data** (*bytes*) – Excel file content.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Raises

KeyError – If the file header variable names or units do not follow the conventions used in this module.

Returns

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **df_aT** (*pandas.DataFrame*) – Contains the shift factors.
- **df_WLF** (*pandas.DataFrame*) – Contains the WLF shift function parameters.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

See also:

load.conventions
Summarizes conventions for variable names and units.

load.file
Returns bytes object to be used as parameter *data*.

load.check_units
Raises KeyError if units don't comply with conventions

pyvisco.load.**user_master**(*data*, *domain*, *RefT*, *modul*)

Load master curve data from csv file.

Either tensile or shear modulus data from materials characterizations performed in either the frequency or time domain are accepted. The file header must consit of two rows. The first row provides the name of the physical quantity and the second row provides the corresponding physical unit. The column headers are checked against the conventions used in this modul. Details on the variable names, units, and file header names are provided in the Notes section below.

Parameters

- **data** (*bytes*) – Excel file content.
- **domain** (*{'freq', 'time'}*) – Defines wether frequency domain ('freq') or time domain ('time') input data are provided.
- **RefT** (*int or float*) – Reference tempeature of the master curve in Celsius.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Returns

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Raises

KeyError – If the file header variable names or units do not follow the conventions used in this module.

See also:

load.conventions
Summarizes conventions for variable names and units.

load.file
Returns bytes object to be used as parameter *data*.

load.check_units
Raises KeyError if units don't comply with conventions.

Notes

Dependent on the performed materials characterization, either tensile or shear modulus values in either the time or frequency domain must be provided.

Tensile moduli are denoted as 'E' and shear moduli are denoted as 'G'. Frequency domain data are provided in Hertz: [f] = Hz Time domain data are provided in seconds: [t] = s

Example input files can be found here: (TBD)

Various examples for file headers:

Domain	Row	Tensile Modulus	Shear Modulus
Frequency	1	f, E_stor, E_loss	f, G_stor, G_loss
Frequency	2	Hz, MPa, MPa	Hz, GPa, GPa
Time	1	t, E_relax	t, G_relax
Time	2	s, MPa	s, GPa

pyvisco.load.**user_shift**(*data_shift*)

Load user provided shift factors from csv file.

Two columns need to be provided in the input file. One for the shift factors *log_aT* and one for the corresponding temperatures *T*. The file header must consit of two rows. The first row provides the name of the physical quantity and the second row provides the corresponding physical unit. The column headers are checked against the conventions used in this modul. Details on the variable names, units, and file header names are provided in the Notes section below.

Parameters
data_shift (*bytes*) – CSV file content.

Returns
df_aT – Contains the shift factors and corresponding temperatures.

Return type
pandas.DataFrame

Raises
KeyError – If the file header variable names or units do not follow the conventions used in this module.

See also:

load.conventions
Summarizes conventions for variable names and units.

load.file
Returns bytes object to be used as parameter *data*.

load.check_units
Raises KeyError if units don't comply with conventions.

Notes

Example file header:

Row	Header
1	T, log_aT
2	C, -

Example input files can be found here: (TBD)

pyvisco.load.**get_sets**(*df_raw*, *num=0*)

Group raw data into measurement sets conducted at the same temperature.

Parameters

- **df_raw** (*pandas.DataFrame*) – Raw measurement data.
- **num** (*int, optional*) – Number of measurement points per temperature level. Default is 0, which means that the number of measurement points within a set is evaluated based on the provided frequency range of the measurement points. The first occurance of the maximum frequency is used to identify the number of measurement points per temperature level.

Returns

df_raw – Contains additional column *Set* compared to input data frame.

Return type

pandas.DataFrame

Notes

This function is intended to be used in combination with input files provided by the Eplexor software. Hence, it is limited to measurements in the frequency domain.

pyvisco.load.**get_units**(*units*, *modul*, *domain*, *Eplexor=False*)

Get the physical units based on the measurement data loading condition (tensile or shear) and domain (time or frequency).

Parameters

- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.
- **domain** (*{'freq', 'time'}*) – Defines wether frequency domain ('freq') or time domain ('time') input data are provided.
- **Eplexor** (*bool, default = false*) – If input file is Excel file from Eplexor software -> True If input file is CSV file from user instrument -> False

Returns

conv – Conventions for physical quantities as key and corresponding units as itmes for the provided measurement loading conditions and domain.

Return type

dict of {str : str}

pyvisco.load.**check_units**(*units*, *modul='E'*)

Check that the input file units conform with the conventions used in this modul.

The input units are compared against the conventions used for the measurement loading conditions and domain. For frequency domain data, the storage and loss modulus values must have the same unit.

Parameters

- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Raises

KeyError – If the file header variable names or units do not follow the conventions used in this module.

1.2.2 pyvisco.master

Collection of functions to prepare the master curve for the identification of the Prony series parameters. Methods are provided to shift the raw measurement data into a master curve and remove measurement outliers through smoothing of the master curve.

Functions

fit_at_pwr(df_raw, gb_ref, gb_shift)	Obtain shift factor between two measurement sets at different tempeatures.
fit_pwr(xdata, ydata)	Define bounds for curve fitting routine and fit power law.
get_aT(df_raw, RefT)	Get shift factors for each temperature level in the raw measurement data.
get_curve(df_raw, df_aT, RefT)	Get master curve by shifting the individual measurement sets.
plot(df_master, units)	Plot master curve.
plot_shift(df_raw, df_master, units)	Plot raw measurement data and shifted master curve.
plot_shift_debug(dshift)	Plot shifted measurement sets.
plot_shift_update(df_master, fig, ax)	Upadate plot of raw measurement data and shifted master curve.
plot_smooth(df_master, units)	Plot filtered and unfilterd master curve.
pwr_x(y, a, b, e)	Calculate the inverse Power Law relation with a deviation term.
pwr_y(x, a, b, e)	Calculate the Power Law relation with a deviation term.
smooth(df_master[, win])	Remove outliers in measurement data by smoothing master curve.

pyvisco.master.**pwr_y**(*x*, *a*, *b*, *e*)

Calculate the Power Law relation with a deviation term.

Parameters

- **x** (*numeric*) – Input to Power Law relation.
- **a** (*numeric*) – Constant.
- **b** (*numeric*) – Exponent.

- **e** (*numeric*) – Deviation term.

Returns
Output of Power Law relation.

Return type
numeric

Notes

Power Law relation: $y = ax^b + e$

pyvisco.master.**pwr_x**(*y*, *a*, *b*, *e*)

Calculate the inverse Power Law relation with a deviation term.

Parameters

- **y** (*numeric*) – Output of Power Law relation.
- **a** (*numeric*) – Constant.
- **b** (*numeric*) – Exponent.
- **e** (*numeric*) – Deviation term.

Returns
Input to Power Law relation.

Return type
numeric

Notes

Inverse Power Law relation: $x = \left(\frac{y-e}{a}\right)^{\frac{1}{b}}$

pyvisco.master.**fit_pwr**(*xdata*, *ydata*)

Define bounds for curve fitting routine and fit power law.

Parameters

- **xdata** (*array-like*) – x data to be fitted.
- **ydata** (*array-like*) – y data to be fitted.

Returns

- **popt** (*array-like*) – Optimal values for the parameters.
- **pcov** (*2D array*) – The estimated covariance of popt. The diagonals provide the variance of the parameter estimate.

See also:

scipy.optimize.curve_fit
Non-linear least squares fit to a function.

pyvisco.master.**fit_at_pwr**(*df_raw*, *gb_ref*, *gb_shift*)

Obtain shift factor between two measurement sets at different tempeatures.

The raw measurement data at each temperature level are fitted by a Power Law function. These Power Law functions improve the robustness of the shifting algorithm, because they functions smooth outliers and bridge possible gaps between the data sets. The intersection of the functions is calculated and used to obtain the shift factor.

Parameters

- **df_raw** (*pandas.DataFrame*) – Contains the processed raw measurement data.
- **gb_ref** (*int*) – Dataframe 'Set' number of the reference measurement set.
- **gb_shift** (*int*) – Dataframe 'Set' number of the measurement set that is shifted.

Returns

log_aT – The decadic logarithm of the shift factor between the two measurement sets.

Return type

numeric

Notes

In certain circumstances the equilibration time between measurements at different temperature levels can be too short to reach a steady state leading to errors in the first data point of the measurement set. To account for such situation, tow Power law fits are conducted. The first fit contains all data points and the second fit drops the first data point. If dropping the data point increased the goodness of fit, this Power Law fit is used to calculate the shift factor.

pyvisco.master.**get_aT**(*df_raw*, *RefT*)

Get shift factors for each temperature level in the raw measurement data.

A reference temperature is specified for which the master curve is created. Measurement sets below the desired reference temperatures are shifted to lower frequencies (longer time periods), whereas measurement sets at temperatures higher than the reference temperature are shifted to higher frequencies (shorter time periods).

Parameters

- **df_raw** (*pandas.DataFrame*) – Contains the processed raw measurement data.
- **RefT** (*int or float*) – Reference tempeature of the master curve in Celsius.

Returns

df_aT – Contains the decadic logarithm of the shift factors 'log_aT' and the corresponding temperature values 'T' in degree Celsius.

Return type

pandas.DataFrame

See also:

load.Eplexor_raw
Returns df_raw from Eplexor Excel file.

load.user_raw
Returns df_raw from csv file.

pyvisco.master.**get_curve**(*df_raw*, *df_aT*, *RefT*)

Get master curve by shifting the individual measurement sets.

The master curve is created from the raw measurement data based on the provided shift factors for the specified reference temperature.

Parameters

- **df_raw** (*pandas.DataFrame*) – Contains the processed raw measurement data.
- **df_aT** (*pandas.DataFrame*) – Contains the decadic logarithm of the shift factors 'log_aT' and the corresponding temperature values 'T' in degree Celsius.
- **RefT** (*int or float*) – Reference tempeature of the master curve in Celsius.

Returns

df_master – Contains the master curve data.

Return type

pandas.DataFrame

See also:

load.Eplexor_raw

Returns df_raw from Eplexor Excel file.

load.user_raw

Returns df_raw from csv file.

master.get_aT

Returns df_aT.

pyvisco.master.**plot**(*df_master*, *units*)

Plot master curve.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns

fig – Domain dependent plot of master curve.

Return type

matplotlib.pyplot.figure

pyvisco.master.**plot_shift**(*df_raw*, *df_master*, *units*)

Plot raw measurement data and shifted master curve.

Parameters

- **df_raw** (*pandas.DataFrame*) – Contains the processed raw measurement data.
- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns

- **fig** (*matplotlib.pyplot.figure*) – Plot of raw measurement data and master curve.
- **ax** (*tuple*) –

Frequency domain: (ax1, lax1, ax2, lax2)

ax1
[matplotlib.axes.Axes] Axes object of storage modulus plot

lax1
[list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in storage modulus plot

ax2
[matplotlib.axes.Axes] Axes object of loss modulus plot

lax2
[list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in loss modulus plot

Time domain: (ax1, lax1)

ax1
[matplotlib.axes.Axes] Axes object of relaxation modulus plot

lax1
[list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in relaxation modulus plot

See also:

`master.plot_shift_update`
Updates figure data.

pyvisco.master.**plot_shift_update**(*df_master*, *fig*, *ax*)

Upadate plot of raw measurement data and shifted master curve.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **fig** (*matplotlib.pyplot.figure*) – Matplotlib figure instance.
- **ax** (*tuple*) –

  Frequency domain: (ax1, lax1, ax2, lax2)

  ax1
  [matplotlib.axes.Axes] Axes object of storage modulus plot

  lax1
  [list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in storage modulus plot

  ax2
  [matplotlib.axes.Axes] Axes object of loss modulus plot

  lax2
  [list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in loss modulus plot

  Time domain: (ax1, lax1)

  ax1
  [matplotlib.axes.Axes] Axes object of relaxation modulus plot

lax1
[list of matplotlib.lines.Line2D] Line2D instances for easy update of xdata and ydata in relaxation modulus plot

See also:

`master.plot_shift`
Creates figure that is updated with this function.

pyvisco.master.**smooth**(*df_master*, *win=1*)

Remove outliers in measurement data by smoothing master curve.

A moving median filter with variable window size is applied to remove outliers from the measurement data.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **win** (*int, default = 1*) – Window size of the median filter. A window size of 1 means that no filtering procedure is performed and the input data are returned.

Returns
df_master – Contains the master curve data, including the filtered arrays.

Return type
pandas.DataFrame

pyvisco.master.**plot_smooth**(*df_master*, *units*)

Plot filtered and unfilterd master curve.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the filtered and unfiltered master curve data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Plot displaying the filtered and unfilterd master curve data.

Return type
matplotlib.pyplot.figure

pyvisco.master.**plot_shift_debug**(*dshift*)

Plot shifted measurement sets. Helpful for manual adjustment of master curve.

Parameters
dshift (*dictionary*) – Contains data of the shifted measurement sets.

Returns
fig – Plot displaying the fitted and shifted measurement data sets.

Return type
matplotlib.pyplot.figure

1.2.3 pyvisco.opt

Collection of functions to minimize the number of Prony series terms used in the Generalized Maxwell model.

Functions

nprony(df_master, prony_series[, window, opt])	Minimize number of Prony terms used in Generalized Maxwell model.
plot_fit(df_master, dict_prony, N, units)	Calculate and plot the optimized Prony series fit.
plot_residual(err)	Plot the least squares residual of the Prony series fits.

pyvisco.opt.**nprony**(*df_master*, *prony_series*, *window='min'*, *opt=1.5*)

Minimize number of Prony terms used in Generalized Maxwell model.

The number of Prony terms is gradually decreased and the new Prony series parameters are identified. The goodness of fit is evaluated based on the R^2 measure. An optimal number of Prony terms is suggested.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **prony_series** (*dict*) – Contains the Prony series parameters of the initial fit.
- **window** (*{'min', 'round', 'exact'}*) – Defines the location of the discretization of the relaxation times. - 'exact' : use whole window of the experimental data and logarithmically space the relaxation times between - 'round' : round the minimum and maximum values of the experimental data to the nearest base 10 number and logarithmically space the remaining relaxation times between the rounded numbers - 'min' : Position of relaxation times is optimized during minimization routine to reduce the number of Prony terms.
- **opt** (*numeric*) – Multiplier for the inital least squares residual to suggest an optimal number of Prony terms: (R_opt)^2 = opt * (R_0)^2

Returns

- **dict_prony** (*dict{N : prony_series}*) – Contains all prony_series parameters for each number of calculated Prony terms, N.
- **N_opt** (*int*) – Optimal number of Prony terms.
- **err** (*pandas.DataFrame*) – Contains the least sqare residuals for each calculated Prony series.

pyvisco.opt.**plot_fit**(*df_master*, *dict_prony*, *N*, *units*)

Calculate and plot the optimized Prony series fit.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **dict_prony** (*dict{N : prony_series}*) – Contains all Prony series parameters for each number of calculated Prony terms, N.
- **N** (*int*) – Number of Prony terms for plot.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns

- **df_GMaxw** (*pandas.DataFrame*) – Contains the calculated Generalized Maxwell model data for the Prony series with N terms.
- **fig** (*matplotlib.pyplot.figure*) – Plot of optimized Prony series fit.

pyvisco.opt.**plot_residual**(*err*)

Plot the least squares residual of the Prony series fits.

Parameters
err (*pandas.DataFrame*) – Contains the least sqare residuals for each calculated Prony series.

Returns
fig – Matplotlib figure instance.

Return type
matplotlib.pyplot.figure

See also:

opt.nprony
Return the err dataframe.

1.2.4 pyvisco.out

Collection of functions to write pandas dataframes to files or buffer.

Functions

add_units(df, units[, index_label])	Add second row units header to pandas dataframe containing.
to_csv(df, units[, index_label, filepath])	Write dataframe to csv file or returns bytes object with csv file content.

pyvisco.out.**add_units**(*df*, *units*, *index_label=None*)

Add second row units header to pandas dataframe containing.

A multiindex pandas dataframe is created. The first header row contains the physical quantities and the second row the corresponding physical units.

Parameters

- **df** (*pandas.DataFrame*) – Units will be added to this dataframe.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.
- **index_label** (*str*) – If index_label is specified, a name for the row index will be included in the multiindex column header.

Returns
df_out – Dataframe with a multiindex colum header containing the physical units.

Return type
pandas.DataFrame

See also:

out.to_csv
Parent function of current function.

pyvisco.out.**to_csv**(*df*, *units*, *index_label=None*, *filepath=None*)

Write dataframe to csv file or returns bytes object with csv file content.

This function adds units to the input dataframe and creates a csv file object. If filepath is specified the csv file is written to the specified location. If filepath is not specified a bytes object containing the csv file data is returned.

Parameters

- **df** (*pandas.DataFrame*) – Dataframe to be written to csv file.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.
- **index_label** (*str*) – If index_label is specified, a name for the row index will be included in the multiindex column header.
- **filepath** (*str, default = None*) – Filepath to storage location of the new csv file.

Returns
If filepath is specified the dataframe will be written to a csv file. If filepath is not specified, the function returns a bytes object containing the csv file data.

Return type
pandas.DataFrame

1.2.5 pyvisco.prony

Collection of function to pre-process the master curve and perform the Prony series parameter identification.

Functions

E_freq_norm(omega, alpha_i, tau_i)	Calculate normalized storage and loss modulus values.
E_relax_norm(time, alpha_i, tau_i)	Calculate normalized relaxation modulus values.
GMaxw_temp(shift_func, df_GMaxw, df_coeff, df_aT)	Calculate Gen.
calc_GMaxw(E_0, df_terms, f_min, f_max, ...)	Calculate the Generalized Maxwell model data from the Prony series parameter.
discretize(df_master[, window, nprony])	Discretizes relaxation times over time or frequency axis.
fit(df_dis[, df_master, opt])	Generalized function to call the domain dependent curve fitting routine.
fit_freq(df_dis, df_master[, opt])	Fit Prony series parameter in frequency domain.
fit_time(df_dis, df_master[, opt])	Fit Prony series parameter in time domain.
ls_res(func)	Wrapper function that calculates the least squares residual.
plot_GMaxw(df_GMaxw, units)	Plot Generalized Maxwell model data for the reference temperature.
plot_GMaxw_temp(df_GMaxw_temp, units)	Plot Generalized Maxwell model data for varies temperature and frequencies.
plot_dis(df_master, df_dis, units)	Plot relaxation times on top of master curve.
plot_fit(df_master, df_GMaxw, units)	Plot the master curve and corresponding Prony fit (Gen.
plot_param(prony_list[, labels])	Plot illustrating the Prony series parameters of one or more fits.
split_x0(func)	Wrapper that splits array x0 of the minimization routine into two arrays.

pyvisco.prony.**discretize**(*df_master*, *window='round'*, *nprony=0*)

Discretizes relaxation times over time or frequency axis.

Discrete relaxation times are required for Prony parameter curve fitting routine. This function spaces the relaxation times over the experimental characterization window.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **window** (*{'round', 'exact', 'min'}*) – Defines the location of the discretization of the relaxation times. - 'exact' : Use whole window of the experimental data and logarithmically space the relaxation times inbetween. - 'round' : Round the minimum and maximum values of the experimental data to the nearest base 10 number and logarithmically space the remaining relaxation times inbetween the rounded numbers - 'min' : Position of relaxation times is optimized during minimization routine to reduce the number of Prony terms.
- **nprony** (*numeric, default = 0*) – Number of Prony terms to be used for the discretization. The number of Prony terms and the number of relaxation times is equal. If no number or 0 is specified, the default behavior of one Prony term per decade is used to automatically calculate the number of Prony terms.

Returns

df_dis – Contains discrete point, equal to the relaxation times, of the master curve data (df_master).

Return type

pandas.DataFrame

References

Kraus, M. A., and M. Niederwald. "Generalized collocation method using Stiffness matrices in the context of the Theory of Linear viscoelasticity (GUSTL)." Technische Mechanik-European Journal of Engineering Mechanics 37.1 (2017): 82-106.

pyvisco.prony.**plot_dis**(*df_master*, *df_dis*, *units*)

Plot relaxation times on top of master curve.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **df_dis** (*pandas.DataFrame*) – Contains the discrete relaxation times and corresponding data.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Plot showing the relaxation times on top of the master curve.

Return type
matplotlib.pyplot.figure

pyvisco.prony.**ls_res**(*func*)

Wrapper function that calculates the least squares residual.

Parameters
func (*function*) – Time domain: prony.E_relax_norm Frequency domain: prony.E_freq_norm

Returns
residual – Calculates least squares residual for specified domain.

Return type
function

pyvisco.prony.**split_x0**(*func*)

Wrapper that splits array x0 of the minimization routine into two arrays.

Splits the the first argument x0 into two arrays alpha_i and tau_i and forwards both arrays to the called function. A single array x0 is necessary to optimize both alpha_i and tau_i at the same time. However, typically, only alpha_i is optimized and tau_i is kept constant. This wrapper allows to use the same function in both scenarios.

Parameters
func (*function*) – Function that calculates least squares residual.

Returns
split

Return type
function

See also:

prony.ls_res
Function to be wrapped during minimization of Prony terms.

pyvisco.prony.**E_relax_norm**(*time*, *alpha_i*, *tau_i*)

Calculate normalized relaxation modulus values.

Parameters

- **time** (*array-like*) – Time in s.
- **alpha_i** (*array-like*) – Normalized relaxation moduli (unitless).
- **tau_i** (*array-like*) – relaxation times in s.

Returns
Relaxation modulus values.

Return type
numpy.ndarray

pyvisco.prony.**fit_time**(*df_dis*, *df_master*, *opt=False*)

Fit Prony series parameter in time domain.

A least-squares minimization is performed using the L-BFGS-B method from the scipy package. The implementation is similar to the optimization problem described by [1] for a homogenous distribution of discrete times.

Parameters

- **df_dis** (*pandas.DataFrame*) – Contains the discrete relaxation times and corresponding data.
- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **opt** (*bool, default = False*) – Flag indicates wether the Prony term minimization routine should be executed or not.

Returns
prony – Contains the Prony series parameters of the fit.

Return type
dict

References

[1] Barrientos, E., Pelayo, F., Noriega, Á. et al. Optimal discrete-time Prony series fitting method for viscoelastic materials. Mech Time-Depend Mater 23, 193-206 (2019). https://doi.org/10.1007/s11043-018-9394-z

pyvisco.prony.**E_freq_norm**(*omega*, *alpha_i*, *tau_i*)

Calculate normalized storage and loss modulus values.

Parameters

- **omega** (*array-like*) – Angular frequency in rad/s.
- **alpha_i** (*array-like*) – Normalized relaxation moduli (unitless).
- **tau_i** (*array-like*) – relaxation times in s.

Returns
Concatenated array of normalized storage and loss modulus values.

Return type
numpy.ndarray

pyvisco.prony.**fit_freq**(*df_dis*, *df_master*, *opt=False*)

Fit Prony series parameter in frequency domain.

A least-squares minimization is performed using the L-BFGS-B method from the scipy package. The implementation is similar to the optimization problem described by [1] for a homogenous distribution of discrete times.

Parameters

- **df_dis** (*pandas.DataFrame*) – Contains the discrete relaxation times and corresponding data.
- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **opt** (*bool, default = False*) – Flag indicates wether the Prony term minimization routine should be executed or not.

Returns
prony – Contains the Prony series parameters of the fit.

Return type
dict

References

[1] Barrientos, E., Pelayo, F., Noriega, Á. et al. Optimal discrete-time Prony series fitting method for viscoelastic materials. Mech Time-Depend Mater 23, 193-206 (2019). https://doi.org/10.1007/s11043-018-9394-z

pyvisco.prony.**calc_GMaxw**(*E_0*, *df_terms*, *f_min*, *f_max*, *decades*, *modul*, ***kwargs*)

Calculate the Generalized Maxwell model data from the Prony series parameter.

Parameters

- **E_0** (*numeric*) – Instantaneous storage modulus. Same variable name is used for either tensile (E_0) or shear (G_0) loading.
- **df_terms** (*pandas.DataFrame*) – Contains the Prony series parameters tau_i and alpha_i.
- **f_min** (*numeric*) – Lower bound frequency for calculation of physical quanitities.
- **f_max** (*numeric*) – Upper bound frequency for calculation of physical quanitities.
- **decades** (*integer*) – Number of decades spanning the frequency window. Is used to calculate the necessary number of data points spanning the frequency range for an appropriate resolution.
- **modul** (*{'E', 'G'}*) – Indicates wether tensile ('E') or shear ('G') modulus data are provided.

Returns
df_GMaxw – Contains the calculated Generalized Maxwell model data for the fitted Prony series parameters with the specified boundaries and parameters.

Return type
pandas.DataFrame

pyvisco.prony.**GMaxw_temp**(*shift_func*, *df_GMaxw*, *df_coeff*, *df_aT*, *freq=[1e-08, 0.0001, 1.0, 10000.0]*)

Calculate Gen. Maxwell model for different loading frequencies and temperatures.

This function showcases the temperature and rate-dependence of the visco- elastic material. The specified shift function is used to calculate the material response at different temperatures and different loading rates.

Parameters

- **shift_func** (*{'WLF', 'D4', 'D3', 'D2', 'D1'}*) – Specifies the shift function to be used for calculations.
- **df_GMaxw** (*pandas.DataFrame*) – Contains the Generalized Maxwell model data for the reference temperature at different loading rates.
- **df_coeff** (*pandas.DataFrame*) – Contains the coefficients and parameters for the specified shift function.

- **df_aT** (*pandas.DataFrame*) – Contains the shift factors. The shift factors are used to identify the Temperature range for the calculation.
- **freq** (*array-like, default = [1E-8, 1E-4, 1E0, 1E4]*) – Loading frequencies for which the calculations are performed.

Returns
Contains the Generalized Maxwell model data for a wide range of temperatures at the specified frequencies.

Return type
df_GMaxw_temp

See also:

shift.fit_WLF
Returns WLF shift functions.

shift.fit_poly
Returns polynomial shift functions of degree 1 to 4.

pyvisco.prony.**plot_GMaxw**(*df_GMaxw*, *units*)

Plot Generalized Maxwell model data for the reference temperature.

Parameters

- **df_GMaxw** (*pandas.DataFrame*) – Contains the Generalized Maxwell model data for the reference temperature at different loading rates.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Plot of calculated storage, loss, and relaxation modulus.

Return type
matplotlib.pyplot.figure

pyvisco.prony.**plot_GMaxw_temp**(*df_GMaxw_temp*, *units*)

Plot Generalized Maxwell model data for varies temperature and frequencies.

Parameters

- **df_GMaxw_temp** (*pandas.DataFrame*) – Contains the Generalized Maxwell model data for various temperatures and different loading rates.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Plot of showing the temperature and rate dependence of the storage, loss, and relaxation modulus.

Return type
matplotlib.pyplot.figure

pyvisco.prony.**plot_param**(*prony_list*, *labels=None*)

Plot illustrating the Prony series parameters of one or more fits.

Parameters

- **prony_list** (*list*) – List of *prony* dictionaries containing the Prony series parameters.

- **labels** (*list of str*) – List of strings to be used as legend label names.

Returns
fig – Plot showing the relaxation moduli over the relaxation times.

Return type
matplotlib.pyplot.figure

See also:

prony.fit
Returns the prony dictionary to be used in prony_list.

pyvisco.prony.**fit**(*df_dis*, *df_master=None*, *opt=False*)

Generalized function to call the domain dependent curve fitting routine.

Parameters

- **df_dis** (*pandas.DataFrame*) – Contains the discrete relaxation times and corresponding data.
- **df_master** (*pandas.DataFrame, default = None*) – Contains the master curve data. Not required for the initial fit in the frequency domain (opt = False).
- **opt** (*bool, default = False*) – Flag indicates wether the Prony term minimization routine should be executed or not.

Returns

- **prony** (*dict*) – Contains the Prony series parameters of the fit.
- **df_GMaxw** (*pandas.DataFrame*) – Contains the calculated Generalized Maxwell model data for the fitted Prony series parameters.

pyvisco.prony.**plot_fit**(*df_master*, *df_GMaxw*, *units*)

Plot the master curve and corresponding Prony fit (Gen. Maxwell model).

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **df_GMaxw** (*pandas.DataFrame*) – Contains the calculated Generalized Maxwell model data for the fitted Prony series parameters.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Domain dependent plot of master curve and Prony fit.

Return type
matplotlib.pyplot.figure

1.2.6 pyvisco.shift

Collection of functions to apply the time-temperature superposition principle to create a master curve from measurements performed at different temperatures.

Functions

WLF(Temp, RefT, WLF_C1, WLF_C2)	Calculate the Williams-Landel-Ferry (WLF) equation [1].
fit_WLF(RefT, df_aT)	Fit the Williams-Landel-Ferry (WLF) equation [1] to a set of shift factors.
fit_poly(df_aT)	Fit polynomial functions of degree 1 to 4 to a set of shift factors.
plot(df_aT, df_WLF, df_C)	Plot shift factors and shift functions.
poly1(x, C0, C1)	Calculate a polynomial function of degree 1 with a single variable 'x'.
poly2(x, C0, C1, C2)	Calculate a polynomial function of degree 2 with a single variable 'x'.
poly3(x, C0, C1, C2, C3)	Calculate a polynomial function of degree 3 with a single variable 'x'.
poly4(x, C0, C1, C2, C3, C4)	Calculate a polynomial function of degree 4 with a single variable 'x'.

pyvisco.shift.**WLF**(*Temp*, *RefT*, *WLF_C1*, *WLF_C2*)

Calculate the Williams-Landel-Ferry (WLF) equation [1].

Parameters

- **Temp** (*numeric*) – Evaluation temperature of the shift factor.
- **RefT** (*numeric*) – Reference temperature chosen to construct the master curve.
- **WLF_C1** (*numeric*) – Empirical constants. (Obtained from fitting the shift factor a_T)
- **WLF_C2** (*numeric*) – Empirical constants. (Obtained from fitting the shift factor a_T)

Returns

log_aT – The decadic logarithm of the WLF shift factor.

Return type

numeric

References

[1] Williams, Malcolm L.; Landel, Robert F.; Ferry, John D. (1955). "The Temperature Dependence of Relaxation Mechanisms in Amorphous Polymers and Other Glass-forming Liquids". J. Amer. Chem. Soc. 77 (14): 3701-3707. doi:10.1021/ja01619a008

pyvisco.shift.**poly1**(*x*, *C0*, *C1*)

Calculate a polynomial function of degree 1 with a single variable 'x'.

Parameters

- **x** (*numeric*) – Input variable.
- **C0** (*numeric*) – Polynomial coefficients

- **C1** (*numeric*) – Polynomial coefficients

Returns
Result of the polynomial function.

Return type
numeric

pyvisco.shift.**poly2**(*x*, *C0*, *C1*, *C2*)

Calculate a polynomial function of degree 2 with a single variable 'x'.

Parameters

- **x** (*numeric*) – Input variable.
- **C0** (*numeric*) – Polynomial coefficients
- **C1** (*numeric*) – Polynomial coefficients
- **C2** (*numeric*) – Polynomial coefficients

Returns
Result of the polynomial function.

Return type
numeric

pyvisco.shift.**poly3**(*x*, *C0*, *C1*, *C2*, *C3*)

Calculate a polynomial function of degree 3 with a single variable 'x'.

Parameters

- **x** (*numeric*) – Input variable.
- **C0** (*numeric*) – Polynomial coefficients
- **C1** (*numeric*) – Polynomial coefficients
- **C2** (*numeric*) – Polynomial coefficients
- **C3** (*numeric*) – Polynomial coefficients

Returns
Result of the polynomial function.

Return type
numeric

pyvisco.shift.**poly4**(*x*, *C0*, *C1*, *C2*, *C3*, *C4*)

Calculate a polynomial function of degree 4 with a single variable 'x'.

Parameters

- **x** (*numeric*) – Input variable.
- **C0** (*numeric*) – Polynomial coefficients
- **C1** (*numeric*) – Polynomial coefficients
- **C2** (*numeric*) – Polynomial coefficients
- **C3** (*numeric*) – Polynomial coefficients
- **C4** (*numeric*) – Polynomial coefficients

Returns
Result of the polynomial function.

Return type
numeric

pyvisco.shift.**fit_WLF**(*RefT*, *df_aT*)

Fit the Williams-Landel-Ferry (WLF) equation [1] to a set of shift factors.

Parameters

- **RefT** (*numeric*) – Reference temperature chosen to construct the master curve.
- **df_aT** (*pandas.DataFrame*) – Contains the decadic logarithm of the shift factors 'log_aT' and the corresponding temperature values 'T' in degree Celsius.

Returns
df – Contains the necessary parameters to calculate the WLF equation (RefT, WLF_C1, WLF_C2).

Return type
pandas.DataFrame

See also:

shift.WLF
Calculates the WLF equation.

Note, -----, Too, a, an, empirical, only, is

References

[1] Williams, Malcolm L.; Landel, Robert F.; Ferry, John D. (1955). "The Temperature Dependence of Relaxation Mechanisms in Amorphous Polymers and Other Glass-forming Liquids". J. Amer. Chem. Soc. 77 (14): 3701-3707. doi:10.1021/ja01619a008

pyvisco.shift.**fit_poly**(*df_aT*)

Fit polynomial functions of degree 1 to 4 to a set of shift factors.

Parameters
df_aT (*pandas.DataFrame*) – Contains the decadic logarithm of the shift factors 'log_aT' and the corresponding temperature values 'T' in degree Celsius.

Returns

- **df_C** (*pandas.DataFrame*) – Contains the coefficients to calculate the polynomial shift functions of degree 1 to 4 for temperatures in degree **Celsius**.
- **df_K** (*pandas.DataFrame*) – Contains the coefficients to calculate the polynomial shift functions of degree 1 to 4 for temperatures in **Kelvin**.
- *Note*
- ——
- *The coefficients of the polynomial shift funtions are dependent on the*
- *Temperature unit. Hence, two different dataframes are provided for*
- *temperatures described in Celsius and Kelvin. For temperatures in Kelvin,*
- *at least 5 significant figures should be used for the polynomial*
- *coefficients to obtain accurate results for the polynomial shift functions.*
- *The interconversion from degree Celsius (T_C) to Kelvin (T_K) is performed*

- **as** ($T\_K = T\_C + 273.15$.)

pyvisco.shift.**plot**(*df_aT*, *df_WLF*, *df_C*)

Plot shift factors and shift functions.

Parameters

- **df_aT** (*pandas.DataFrame*) – Contains the decadic logarithm of the shift factors 'log_aT' and the corresponding temperature values 'T' in degree Celsius.
- **df_WLF** (*pandas.DataFrame*) – Contains the necessary parameters to calculate the WLF equation (RefT, WLF_C1, WLF_C2) in degree Celsius.
- **df_C** (*pandas.DataFrame*) – Contains the coefficients to calculate the polynomial shift functions of degree 1 to 4 for temperatures in degree **Celsius**.

Returns

- **fig** (*matplotlib.pyplot.figure*) – Matplotlib figure instance.
- **df_shift** (*pandasDataFrame*) – Contains the data used to create the plot.

1.2.7 pyvisco.styles

Collection of function to define figure style and html styles for Jupyter Notebooks.

Functions

format_fig()	Set matplotlib.pyplot.rcParams for figure style.

pyvisco.styles.**format_fig**()

Set matplotlib.pyplot.rcParams for figure style.

1.2.8 pyvisco.verify

Collection of function to compare and verify the Python implementation within this module with the curve fitting routine of Ansys APDL 2021 R1.

Functions

load_prony_ANSYS(filepath)	Load Prony series parameters from ANSYS material card file.
plot_fit_ANSYS(df_master, df_GMaxw, ...)	Plot master curve, Prony series fit of Python implementation, and Prony series fit of ANSYS curve fitting routine.
prep_prony_ANSYS(df_prony, prony[, E_0])	Prepare ANSYS Prony series parameters for further processing.

pyvisco.verify.**load_prony_ANSYS**(*filepath*)

Load Prony series parameters from ANSYS material card file.

Parameters

filepath (*str*) – Location and file name of ANSYS material card file (.MPL)

Returns
df_prony – Contains the Prony series parameter.

Return type
pandas.DataFrame

pyvisco.verify.**prep_prony_ANSYS**(*df_prony*, *prony*, *E_0=None*)

Prepare ANSYS Prony series parameters for further processing.

The ANSYS curve fitting routine for viscoelastic materials only stores the Prony series parameters ('tau_i', 'alpha_i') in the material card file. To calculate the master curve from the Prony series parameters the instantenous modulus and frequency range are required and added to the dataframe of the ANSYS Prony series parameters.

Parameters

- **df_prony** (*pandas.DataFrame*) – Contains the ANSYS Prony series parameter.
- **prony** (*dict*) – Contains the Python Prony series parameter
- **E_0** (*float, default = None*) – Instantaneous storage modulus; for either tensile (E_0) or shear (G_0) loading. If E_0 is not provided the instantaneous storage modulus identified during the Python curve fitting process will be used to create the master curve with the ANSYS Prony series parameters.

Returns
prony_ANSYS – Contains the ANSYS Prony series parameter in the same format as the Python implementation provides (see 'prony' Parameter above).

Return type
dict

pyvisco.verify.**plot_fit_ANSYS**(*df_master*, *df_GMaxw*, *df_GMaxw_ANSYS*, *units*)

Plot master curve, Prony series fit of Python implementation, and Prony series fit of ANSYS curve fitting routine.

Parameters

- **df_master** (*pandas.DataFrame*) – Contains the master curve data.
- **df_GMaxw** (*pandas.DataFrame*) – Contains the calculated Generalized Maxwell model data for the Prony series obtained with the **Python** implementation.
- **df_GMaxw_ANSYS** (*pandas.DataFrame*) – Contains the calculated Generalized Maxwell model data for the Prony series obtained with the **ANSYS** implementation.
- **units** (*dict of {str : str}*) – Contains the names of the physical quantities as key and the corresponding names of the units as item.

Returns
fig – Plot of master curve and Prony fits.

Return type
matplotlib.pyplot.figure

1.2.9 pyvisco.inter

Collection of classes and functions to create a graphical user interface based on ipywidgets and Jupyter notebook.

Functions

fig_bytes(fig)	Return figure object in bytes.
generate_zip(files)	Generate zip archive from collection of dataframes and figures.

Classes

Control()	Collection of methods to provide interactive functionality for the Jupyter Notebook.
Widgets()	Collection of widgets for Jupyter notebook dashboard.

pyvisco.inter.**generate_zip**(*files*)

Generate zip archive from collection of dataframes and figures.

pyvisco.inter.**fig_bytes**(*fig*)

Return figure object in bytes.

class pyvisco.inter.**Widgets**

Bases: object

Collection of widgets for Jupyter notebook dashboard.

ini_variables()

Default values for widgets.

widgets()

Define GUI widgets.

class pyvisco.inter.**Control**

Bases: *Widgets*

Collection of methods to provide interactive functionality for the Jupyter Notebook.

collect_files()

Create dictionary to collect output files.

create_loading_bar()

Create loading bar object for methods with longer runtime.

reset_notebook()

Reset notebook configuration and clear data.

exceptions()

Wrapper method providing exception handling for methods loading files.

show_theory(*change*)

Show theory section from HTML file.

set_domain(*change*)

Set measurement domain and update widgets.

set_loading(*change*)

Set modulus based on loading direction and update widgets.

set_instrument(*change*)

Set instrument type and update widgets.

set_type(*change*)

Set type of modulus data and update widgets.

set_shift(*change*)

Set wether user shift factors are provided and update widgets.

set_RefT(*change*)

Set reference temperature and update widgets.

inter_aT(*b*)

Execute interactive routine to fit shift factors.

inter_shift(*b*)

Execute interactive routine to obtaine shift functions.

inter_smooth_fig(*b*)

Create interactive figure for smoothing routine.

inter_smooth(*win*)

Execute the interactive smoothing routine of the master curve.

set_dis(*change*)

Set discretization parameters and update widgets.

inter_dis(*b*)

Execute the interactive discretization routine.

inter_fit(*b*)

Execute the interactive Prony parameter fitting routine.

inter_GMaxw(*b*)

Execute interactive routine to calculate and plot the Generalized Maxwell model.

inter_opt_fig(*N*)

Create interactive figure for optimization routine.

inter_opt(*b*)

Execute the interactive optimization routine.

trigger_download(*data*, *filename*, *kind='text/json'*)

Trigger download through HTML output widget.

Works in Jupyter notebook and voila.

References

https://github.com/voila-dashboards/voila/issues/711

down_zip(*b*)

Create zip archive of all dataframes and figures and trigger download.

reload(*b*)

Reload the webpage to clear all data and recreate class objects.

set_inp_step(*change*)

Set the step size for manually modifying the shift factors.

get_aT(*T*)

Get corresponding shift factor to specified temperature.

set_inp_T(*change*)

Set temperature of shift factor to be modified and update widgets.

set_inp_aT(*change*)

Manually modify shift factors and update master curve.

reset_df_aT()

Reset shift factors to initial state after manually modifying them.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- search

CHAPTER

THREE

DEPENDENCIES

pyvisco depends on the following packages:

```
jupyter==1.0.0
matplotlib==3.5.1
numpy==1.22.2
pandas==1.4.1
scipy==1.8.0
ipython==8.0.1
ipywidgets==7.6.5
ipympl==0.8.8
voila==0.3.6
xlrd==2.0.1
Markdown==3.3.6
lxml_html_clean==0.2.0
jinja2==2.11.3
markupsafe==2.0.1
```

PYTHON MODULE INDEX

p

INDEX

R

S

T

U

W