mechmat Documentation

Release 0.2.4

Jelle Spijker

Apr 21, 2020

CONTENTS:

CHAPTER

ONE

MECHMAT

Python package for the definition of materials used during mechanical engineering calculations

- Free software: MIT license
- Documentation: https://mechmat.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package makes use of pint for unit safe calculations.

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install mechmat, run this command in your terminal:

```
$ pip install mechmat
```

This is the preferred method to install mechmat, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for mechmat can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/jellespijker/mechmat
```

Or download the tarball:

```
$ curl  -OL https://github.com/jellespijker/mechmat/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use mechmat in a project:

```
import mechmat
```

CHAPTER

FOUR

MECHMAT

4.1 mechmat package

4.1.1 Subpackages

mechmat.core package

Submodules

mechmat.core.chainable module

class mechmat.core.chainable.**Chainable**(***kwargs*)
Bases: object

" A linked attribute class

link_attr(*attr*, *transform*, ***kwargs*)
Link a Linked attribute against another Linked attribute.

Args: attr (str): Attribute name transform: The function which provides the transform. **kwargs: the transform function keywords where the value is either a str (if the attribute can be obtained

from the own instance) or a tuple containing the other instance and attribute name.

linked_transforms(*attr*)

set_guard(*attr*, *unit=None*, *rng=None*, *doc=None*)
Set the guard descriptor unit and range, this is usually set in the __init__() function

Args: attr (str): The guard attribute to be set unit (ureg.Unit): The unit in which guarded inputs are to be converted rng (tuple, list, np.array): The range [low, high] against which to test doc (str): dosctring

unlink_attr(*attr*, *transform*)

class mechmat.core.chainable.**Guarded**
Bases: object

Descriptor guarding Linked attributes

static cite_value(*value*)

static in_range(*value*, *rng*)
is value in specified range

Args: value: value to be tested rng: tuple or list to be tested against

Returns: true when in range otherwise false

class mechmat.core.chainable.**Message**
Bases: set

Linked Message

mechmat.core.errors module

exception mechmat.core.errors.**OutOfRangeError**(*value*, *rng*, *property*)
Bases: ValueError

Raised when trying to set an out-of-range value

Module contents

mechmat.principal package

Submodules

mechmat.principal.core module

mechmat.principal.core.**reciprocal**(*value*)

mechmat.principal.core.**sub**(***kwargs*)

mechmat.principal.core.**add**(***kwargs*)

mechmat.principal.core.**mul**(***kwargs*)

mechmat.principal.core.**div**(***kwargs*)

class mechmat.principal.core.**Interp**(*kind='cubic'*, *cite=None*, ***kwargs*)
Bases: object

mechmat.principal.crossarrhenius module

mechmat.principal.crossarrhenius.**arrhenius_shift**(*temperature*, *arrhenius_activation_energy*, *temperature_ref*)

mechmat.principal.crossarrhenius.**relaxation_time**(*relaxation_time_ref*, *arrhenius*)

mechmat.principal.crossarrhenius.**viscosity_dynamic**(*shear_rate*, *zero_shear_viscosity*, *relaxation_time*, *shear_thinning_const*)

mechmat.principal.crossarrhenius.**zero_shear_viscosity**(*arrhenius*, *zero_shear_viscosity_ref*)

mechmat.principal.density module

mechmat.principal.density.**from_specific_weight**(*specific_weight*)

Args: specific_weight:

Returns:

mechmat.principal.geometry module

mechmat.principal.geometry.**distance**(*point_1*, *point_2*)
Returns the distance between two points.

Args: point_1: Scalar or vector of point 1 point_2: Scalar or vector of point 2

Returns: Scalar of the distance between point_2 and point_1

mechmat.principal.geometry.**halfway**(*point_1*, *point_2*)

mechmat.principal.shear_rate module

mechmat.principal.shear_rate.**circle**(*V_dot*, *r*)
" The apparent shear rate for a melt flowing through a cirlce is defined as

$$\dot{\gamma}_a = \frac{4\dot{V}}{\pi R^3}$$

Source: Rao, Natti S. Basic Polymer Engineering Data. Cincinnati, Ohio, USA: Hanser, 2017.

Args: V_dot: Volumetric_flow in $[L^3t^{-1}]$ r: Radius in $[L^1]$

Returns: Apparent shear rate in $[t^{-1}]$

mechmat.principal.shear_rate.**annulus**(*V_dot*, *r_i*, *r_o*)
The apparent shear rate for a melt flowing through a annulus is defined as

$$\frac{6\dot{V}}{\pi\left(r_o + r_i\right)\left(r_o - r_i\right)^2}$$

Source: Rao, Natti S. Basic Polymer Engineering Data. Cincinnati, Ohio, USA: Hanser, 2017.

Args: V_dot: Volumetric_flow in $[L^3t^{-1}]$ r_i: inner radius in $[L^1]$ r_o: inner radius in $[L^1]$

Returns: Apparent shear rate in $[t^{-1}]$

mechmat.principal.specific_weight module

mechmat.principal.specific_weight.**from_density**(*density*)

Args: density:

Returns:

mechmat.principal.thermal module

mechmat.principal.thermal.**specific_heat_capacity**(*thermal_conductivity*, *density*, *thermal_diffusivity*)

mechmat.principal.thermal.**thermal_conductivity**(*thermal_diffusivity*, *specific_heat_capacity*, *density*)

mechmat.principal.thermal.**thermal_diffusivity**(*thermal_conductivity*, *specific_heat_capacity*, *density*)
The rate of transfer of heat of a material from the hot end to the cold end.

Args: thermal_conductivity: k specific_heat_capacity: c_p density: ρ

Returns:

mechmat.principal.twodomaintaitpvt module

mechmat.principal.twodomaintaitpvt.**get_specific_volume**(*p*, *v_0*, *v_t*, *B*)

mechmat.principal.twodomaintaitpvt.**get_B**(*T*, *b_3*, *b_4*, *b_5*)

mechmat.principal.twodomaintaitpvt.**switch_m_s**(*T*, *T_t*, *s*, *m*)

mechmat.principal.twodomaintaitpvt.**get_T_t**(*p*, *b_5*, *b_6*)

mechmat.principal.twodomaintaitpvt.**get_v_0**(*T*, *b_1*, *b_2*, *b_5*)

mechmat.principal.twodomaintaitpvt.**get_v_t**(*p*, *T*, *T_t*, *b_5*, *b_7*, *b_8*, *b_9*)

Module contents

mechmat.properties package

Subpackages

mechmat.properties.elastic_deformation package

Module contents

mechmat.properties.flow package

Submodules

mechmat.properties.flow.flow module

class mechmat.properties.flow.flow.**Flow**(***kwargs*)

Bases: *mechmat.properties.flow.mass_flow.MassFlow*, *mechmat.properties.flow.volume_flow.VolumeFlow*

mechmat.properties.flow.mass_flow module

class mechmat.properties.flow.mass_flow.**MassFlow**(***kwargs*)

Bases: *mechmat.core.chainable.Chainable*

density
Descriptor guarding Linked attributes

massflow
Descriptor guarding Linked attributes

specific_volume
Descriptor guarding Linked attributes

specific_weight
Descriptor guarding Linked attributes

mechmat.properties.flow.volume_flow module

class mechmat.properties.flow.volume_flow.**VolumeFlow**(***kwargs*)
Bases: *mechmat.properties.geometry.surface.Surface*, *mechmat.properties.geometry.vector.Segment*

volumeflow
Descriptor guarding Linked attributes

Module contents

mechmat.properties.geometry package

Submodules

mechmat.properties.geometry.geometry module

class mechmat.properties.geometry.geometry.**Geometry**(***kwargs*)
Bases: *mechmat.properties.geometry.vector.Vector*, *mechmat.properties.geometry.vector.Segment*, *mechmat.properties.geometry.surface.Surface*, *mechmat.properties.geometry.volume.Volume*

mechmat.properties.geometry.surface module

class mechmat.properties.geometry.surface.**Surface**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

cross_section
Descriptor guarding Linked attributes

mechmat.properties.geometry.vector module

class mechmat.properties.geometry.vector.**Segment**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

distance
Descriptor guarding Linked attributes

point_1
Descriptor guarding Linked attributes

point_2
Descriptor guarding Linked attributes

class mechmat.properties.geometry.vector.**Vector**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

coordinate
Descriptor guarding Linked attributes

mechmat.properties.geometry.volume module

class mechmat.properties.geometry.volume.**Volume**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

volume
Descriptor guarding Linked attributes

Module contents

mechmat.properties.mass package

Submodules

mechmat.properties.mass.mass module

class mechmat.properties.mass.mass.**Mass**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

density
Descriptor guarding Linked attributes

mass
Descriptor guarding Linked attributes

specific_volume
Descriptor guarding Linked attributes

specific_weight
Descriptor guarding Linked attributes

Module contents

mechmat.properties.plastic_deformation package

Module contents

mechmat.properties.pressure package

Submodules

mechmat.properties.pressure.pressure module

class mechmat.properties.pressure.pressure.**Pressure**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

pressure
Descriptor guarding Linked attributes

Module contents

mechmat.properties.shearing package

Submodules

mechmat.properties.shearing.shearing module

class mechmat.properties.shearing.shearing.**Shearing**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

shear_rate
Descriptor guarding Linked attributes

Module contents

mechmat.properties.specific_volume package

Submodules

mechmat.properties.specific_volume.twodomaintaitpvt module

class mechmat.properties.specific_volume.twodomaintaitpvt.**TwoDomainTaitpvT**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

The modified 2-domain Tait pvT model is used to determine the density of the material as a function of the temperature and pressure. This variation impacts on many aspects of the flow simulation.

The 2-domain Tait pvT model is given by the following equations:

$$v(T,p) = v_0(T)\left[1 - C\ln\left(1 + \frac{p}{B(T)}\right)\right] + v_t(T,p)$$

where:

- $v(T,p)$ is the specific geometry at temperature and pressure
- v_0 is the specific geometry at zero gauge pressure
- T is the temperature
- p is the pressure
- C is a constant
- B accounts for the pressure sensitivity of the material

The input for fully specified state:

- b_1s
- b_1m
- b_2s
- b_2m
- b_3s

- b_3m
- b_4s
- b_4m
- b_5
- b_6
- b_7
- b_8
- b_9
- temperature
- pressure

b_1m
Descriptor guarding Linked attributes

b_1s
Descriptor guarding Linked attributes

b_2m
Descriptor guarding Linked attributes

b_2s
Descriptor guarding Linked attributes

b_3m
Descriptor guarding Linked attributes

b_3s
Descriptor guarding Linked attributes

b_4m
Descriptor guarding Linked attributes

b_4s
Descriptor guarding Linked attributes

b_5
Descriptor guarding Linked attributes

b_6
Descriptor guarding Linked attributes

b_7
Descriptor guarding Linked attributes

b_8
Descriptor guarding Linked attributes

b_9
Descriptor guarding Linked attributes

specific_volume_transition_temperature
Descriptor guarding Linked attributes

specific_volume_zero_gauge_pressure
Descriptor guarding Linked attributes

temperature_transition
Descriptor guarding Linked attributes

Module contents

mechmat.properties.thermal package

Submodules

mechmat.properties.thermal.conductivity module

class mechmat.properties.thermal.conductivity.**ThermalConductivity**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

heat_transfer_coeff
Descriptor guarding Linked attributes

thermal_conductance
Descriptor guarding Linked attributes

thermal_conductivity
Descriptor guarding Linked attributes

thermal_diffusivity
Descriptor guarding Linked attributes

thermal_insulance
Descriptor guarding Linked attributes

thermal_resistance
Descriptor guarding Linked attributes

thermal_resistivity
Descriptor guarding Linked attributes

thermal_transmittance_convection
Descriptor guarding Linked attributes

thermal_transmittance_radiation
Descriptor guarding Linked attributes

mechmat.properties.thermal.thermal module

class mechmat.properties.thermal.thermal.**Thermal**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

specific_heat_capacity
Descriptor guarding Linked attributes

temperature
Temperature of a material

temperature_melt
Descriptor guarding Linked attributes

temperature_vapor
Descriptor guarding Linked attributes

thermal_conductivity
Descriptor guarding Linked attributes

thermal_diffusivity
Descriptor guarding Linked attributes

thermal_expansion_coeff
Descriptor guarding Linked attributes

Module contents

mechmat.properties.viscosity package

Submodules

mechmat.properties.viscosity.crossarrhenius module

class mechmat.properties.viscosity.crossarrhenius.**CrossArrhenius**(***kwargs*)
Bases: *mechmat.core.chainable.Chainable*

The model is based on the assumption that the fluid flow obeys the Arrhenius equation for molecular kinetics.

$$\eta(T, \dot{\gamma}) = \frac{\eta_0(T)}{1 + (\lambda(T)\dot{\gamma})^a}$$

where:

- $\eta_0\,(T_{\text{ref}})$ zero shear rate viscosity at reference temperature
- $\lambda\,(T_{\text{ref}}\,)$ "relaxation time" at reference temperature
- a "shear-thinning"constant
- E_{a} Arrhenius activation energy
- R gas constant
- T temperature
- T_{ref} reference temperature

The input for fully specified state:

- temperature
- temperature_cross_arrhenius_ref
- arrhenius_activation_energy
- relaxation_time_ref
- shear_rate
- shear_thinning_const
- viscosity_zero_shear_rate_ref

arrhenius_activation_energy
Descriptor guarding Linked attributes

relaxation_time
Descriptor guarding Linked attributes

relaxation_time_ref
Descriptor guarding Linked attributes

shear_thinning_const
Descriptor guarding Linked attributes

temperature_cross_arrhenius_ref
Descriptor guarding Linked attributes

viscosity_zero_shear_rate
Descriptor guarding Linked attributes

viscosity_zero_shear_rate_ref
Descriptor guarding Linked attributes

Module contents

Module contents

4.1.2 Submodules

4.1.3 mechmat.material module

mechmat.material.**material_factory**(**args*, *flow=False*, ***kwargs*)
Material instance facotry

Args: *args: Chainable sub-propperties flow: is the material a continium flowing $\frac{dm}{}$ **kwargs:

Returns:

4.1.4 mechmat.polymer module

class mechmat.polymer.**PolyLacticAcid**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF, *mechmat.properties.specific_volume.twodomaintaitpvt.TwoDomainTaitpvT*

class mechmat.polymer.**PolyLacticAcidThermoplasticPolyUrethane**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF, *mechmat.properties.specific_volume.twodomaintaitpvt.TwoDomainTaitpvT*

class mechmat.polymer.**PolyLacticAcidThermoplasticStarch**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF, *mechmat.properties.specific_volume.twodomaintaitpvt.TwoDomainTaitpvT*

class mechmat.polymer.**Polyamide66CrossWLF**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF

class mechmat.polymer.**Polycarbonate_CrossWLF**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF

class mechmat.polymer.**PolypropyleneCrossWLF**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF

class mechmat.polymer.**PolystyreneCrossWLF**(***kwargs*)
Bases: mechmat.properties.thermal.polymer.ThermalPolymer, mechmat.properties.viscosity.crosswlf.CrossWLF

4.1.5 Module contents

Top-level package for mechmat.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://gitlab.com/pymech/mechmat/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the Gitlab issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the Gitlab issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

mechmat could always use more documentation, whether as part of the official mechmat docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://gitlab.com/pymech/mechmat/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *mechmat* for local development.

1. Fork the *mechmat* repo on Gitlab.
2. Clone your fork locally:

```
$ git clone git@gitlab.com:your_name_here/mechmat.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mechmat
$ cd mechmat/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mechmat tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to Gitlab:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the Gitlab website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/jellespijker/mechmat/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_mechmat
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Jelle Spijker <spijker.jelle@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.0 (2019-03-29)

- First release on PyPI.

7.2 0.1.4 (2019-05-11)

- Multiple bug fixes
- Accepts Numpy arrays
- State factor for easy creation of material states
- State can now be set when initializing
- Expanded the base material properties
- Added support for Jupyter Markdown, LaTeX and HTML representation

7.3 0.2.0 (2019-05-25)

- Removed the need for a metaclass
- Observer pattern implemented as Chainable class
- Guarded descriptor added
- Modular materials, allows for mix and match of different models
- Two-domain-Tait-pvt added
- Cross-Arrhenius model added

7.4 0.2.1 (2019-05-25)

- property models, functions and values are cited by source

7.5 0.2.2 (2019-05-31)

- Couple of bug-fixes
- dir shows only user variables
- Interpolated function added (measurement data, can now be used)
- Serialization using dill is now possible
- repr string simplified
- Cross-WLF model added
- Some example materials added: PLA, PLA-TPU, PLA-TPS

7.6 0.2.3 (2019-05-31)

- Added multiple thermal properties

7.7 0.2.4 (2019-07-09)

- Bug fix citations

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

R

S

T

U

V

Z