
Livestock CPython Package Documentation

Release 2018.3

Christian Kongsgaard

Mar 11, 2018

CONTENTS

1 Documentation for the PyPI Package: 3

1.1 Livestock Air 3

1.2 Livestock Geometry 8

1.3 Livestock Hydrology 9

1.4 Livestock Misc 10

1.5 Livestock SSH 10

2 Documentation for the Grasshopper Components: 11

3 Indices and tables 13

Python Module Index 15

Livestock is the name of the library of components that has been developed for this thesis. Livestock consists of a series of Grasshopper Python Script components and a underlying collection of Python scripts and a PyPI – Python Package Index - package. This is the documentation for the PyPI package.

DOCUMENTATION FOR THE PYPI PACKAGE:

1.1 Livestock Air

`livestock.air.celsius_to_kelvin(celsius: float) → float`
Converts a temperature in Celsius to Kelvin.

Source: <https://en.wikipedia.org/wiki/Celsius>

Parameters `celsius` (*float*) – Temperature in Celsius

Returns Temperature in Kelvin

Return type `float`

`livestock.air.compute_temperature_relative_humidity(temperature_in_k: <built-in function array>, relative_humidity: <built-in function array>, vapour_mass_flux: <built-in function array>, volume: <built-in function array>) → tuple`

Computes the coupled relative humidity and temperature of an air volume given a vapour flux. The vapour pressure is capped off so it can not exceed the saturated vapour pressure. This potential means that not the whole amount

of vapour flux will be used.

Parameters

- **temperature_in_k** (*numpy.array*) – Current temperature of the air volume in K
- **relative_humidity** (*numpy.array*) – Current relative humidity of the air volume as unit less.
- **vapour_mass_flux** (*numpy.array*) – Vapour mass flux to be added to the air volume in kg/h.
- **volume** – Air volume in m³ :type volume: *numpy.array*

Returns Tuple containing the new temperature in K, new relative humidity as unit less, the latent heat used and the vapour flux used.

Return type `tuple`

`livestock.air.convert_relative_humidity_to_percentage` (*rh*: <built-in function array>) → <built-in function array>

Converts relative humidity from percentage to an unit less number.

Parameters *rh* (*numpy.array*) – Relative humidity as unitless

Returns Relative humidity in percentage

Return type *numpy.array*

`livestock.air.convert_relative_humidity_to_unitless` (*rh*: <built-in function array>) → <built-in function array>

Converts relative humidity from percentage to an unit less number.

Parameters *rh* (*numpy.array*) – Relative humidity in %

Returns Relative humidity as unitless

Return type *numpy.array*

`livestock.air.convert_vapour_flux_to_kgh` (*vapour_flux*: <built-in function array>) → <built-in function array>

Converts a vapour flux from m³/day to kg/h Density of water: 1000kg/m³ Hours per day: 24h/day Conversion: 1000kg/m³/ 24h/day

Parameters *vapour_flux* (*numpy.array*) – Vapour flux in m³/day

Returns Vapour flux in kg/h

Return type *numpy.array*

`livestock.air.diameter_from_area` (*area*: <built-in function array>) → <built-in function array>

Computes the diameter from a given area of a circle. $A = \pi * (d/2)^2 \Rightarrow d = \sqrt{4 * A / \pi}$

Parameters *area* (*numpy.array*) – Area of a circle in m

Returns Diameter in m

Return type *numpy.array*

`livestock.air.kelvin_to_celsius` (*kelvin*: *float*) → *float*

Converts a temperature in Kelvin to Celsius.

Source: <https://en.wikipedia.org/wiki/Celsius>

Parameters *kelvin* (*float*) – Temperature in Kelvin

Returns Temperature in Celsius

Return type *float*

`livestock.air.latent_heat_flux` (*vapour_mass_flux*: <built-in function array>) → <built-in function array>

Computes the latent heat flux related to a certain evapotranspiration flux. The latent heat flux is negative if the vapour flux is positive.

Source: Manickathan, L. et al., 2018. Parametric study of the influence of environmental factors and tree properties on the transpirative cooling effect of trees. Agricultural and Forest Meteorology.

Parameters *vapour_mass_flux* (*numpy.array*) – Vapour volume flux in kg/h

Returns Latent heat flux in J/h.

Return type *numpy.array*


```
livestock.air.max_possible_vapour_flux(vapour_mass_flux: float, volume: float, temperature_in_kelvin: float, vapour_pressure: float) → float
```

Computes the difference between the saturated vapour pressure of an air volume after adding the vapour and latent heat flux to an air volume and the actual vapour pressure of an air volume.

Parameters

- **vapour_mass_flux** (*float*) – Vapour mass flux in kg/h
- **volume** – Air volume in m³ :type volume: float
- **temperature_in_kelvin** (*float*) – Current temperature in K
- **vapour_pressure** (*float*) – Current vapour pressure in Pa

Returns Difference between the saturated vapour pressure after adding the vapour and latent heat flux to

the air volume and the actual vapour pressure of the air volume. :rtype: float

```
livestock.air.new_mean_relative_humidity(volume: <built-in function array>, temperature_internal: <built-in function array>, vapour_pressure_external: <built-in function array>, vapour_production: <built-in function array>) → <built-in function array>
```

Computes a new mean vapour pressure and converts it in to a relative humidity.

Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters

- **volume** (*numpy.array*) – Air volume in m³
- **temperature_internal** (*numpy.array*) – External temperature in K
- **vapour_pressure_external** (*numpy.array*) – External vapour pressure in Pa
- **vapour_production** (*numpy.array*) – Vapour production in kg/h

Returns Relative humidity - unitless

Return type numpy.array

```
livestock.air.new_mean_temperature(volume: <built-in function array>, temperature: <built-in function array>, heat: <built-in function array>) → <built-in function array>
```

Calculates a new mean temperature for the volume.

Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters

- **volume** (*numpy.array*) – Volume in m³
- **temperature** (*numpy.array*) – Temperature at the top of the air volume in K
- **heat** (*numpy.array*) – Added heat to the air volume in J/h

Returns Temperature in K

Return type numpy.array

`livestock.air.new_mean_vapour_pressure` (*volume*: <built-in function array>, *temperature*: <built-in function array>, *vapour_pressure_external*: <built-in function array>, *vapour_production*: <built-in function array>) → <built-in function array>

Calculates a new vapour pressure for the volume.

Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters

- **volume** (*numpy.array*) – Volume in m³
- **temperature** (*numpy.array*) – Temperature in K
- **vapour_pressure_external** (*numpy.array*) – External vapour pressure in Pa
- **vapour_production** (*numpy.array*) – Vapour production in kg/h

Returns New vapour pressure in Pa

Return type *numpy.array*

`livestock.air.new_temperature_and_relative_humidity` (*folder*: *str*) → bool

Calculates a new temperatures and relative humidity for air volumes.

Parameters **folder** (*str*) – Path to folder containing case files.

Returns True

Return type bool

`livestock.air.relative_humidity_to_vapour_pressure` (*relative_humidity*: *float*, *temperature*: *float*) → float

Convert relative humidity to vapour pressure given a air temperature.

Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters

- **relative_humidity** (*float*) – Relative humidity - unitless
- **temperature** (*float*) – Air temperature in K

Returns Vapour pressure in Pa

Return type float

`livestock.air.run_row` (*input_package*: *list*) → tuple

Calculates a new temperatures and relative humidity for a row. A row represent all cells to a given time.

Parameters **input_package** (*list*) – Input package with need inputs.

Returns The row on which the calculation was performed.

Return type tuple

`livestock.air.saturated_vapour_pressure` (*temperature*: *float*) → float

Computes the saturated vapour pressure for a given temperature. Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters **temperature** (*float*) – Temperature in Kelvin

Returns Vapour pressure in Pa

Return type float

`livestock.air.stratification` (*height: float, value_mean: float, height_top: float, value_top: float*) → float

Calculates the stratification of the temperature or relative humidity of the air volume.

Parameters

- **height** (*float*) – Height at which the stratification value is wanted in m.
- **value_mean** (*float*) – Mean value of the air volume. Assumed equal to the value at half of the height of the air volume.
- **height_top** (*float*) – Height at the top of the boundary in m.
- **value_top** (*float*) – Value at the top of the air volume

Returns Value at desired height.

Return type float

`livestock.air.vapour_pressure_to_relative_humidity` (*vapour_pressure: float, temperature: float*) → float

Convert vapour pressure to relative humidity given a air temperature

Source: Peuhkuri, Ruut, and Carsten Rode. 2016. “Heat and Mass Transfer in Buildings.”

Parameters

- **vapour_pressure** (*float*) – Vapour pressure in Pa
- **temperature** (*float*) – Air temperature in K

Returns Relative humidity as unitless

Return type float

`livestock.air.wind_speed_to_flux` (*wind_speed: <built-in function array>, height: <built-in function array>, cross_section: <built-in function array>*) → <built-in function array>

Converts a wind speed through an area to a wind flux.

Parameters

- **wind_speed** (*numpy.array*) – Wind speed in m/s
- **height** (*numpy.array*) – Height of the area in m
- **cross_section** (*numpy.array*) – Width of the area in m

Returns Wind flux in m³/h

Return type numpy.array

`livestock.air.wind_speed_to_hour_flux` (*wind_speed: float*) → float

Converts wind speed into a hourly flux. m/s to m³/h m/s to m³/s = 1:sup:2 m³/s to m³/h = 3600s/h

Parameters **wind_speed** (*float*) – Wind speed in m/s

Returns Wind flux in m³/h

Return type float

Go Back to:

[Livestock Frontpage](#)

[Livestock PyPi](#)

[Livestock Grasshopper](#)

1.2 Livestock Geometry

`livestock.geometry.centroid_z (polygon: shapely.geometry.polygon.Polygon) → float`

Calculates the mean z-value from a Shapely polygon.

Parameters `polygon` (*shapely.geometry.Polygon*) – Shapely Polygon with z-values.

Returns Mean z-value of the polygon

Return type float

`livestock.geometry.obj_to_lists (obj_file: str) → tuple`

Converts an .obj file into lists.

Parameters `obj_file` (*str*) – .obj file path

Returns tuple with vertices, normals, faces

Return type tuple

`livestock.geometry.obj_to_polygons (obj_file: str) → list`

Converts an .obj file into a list of shapely polygons.

Parameters `obj_file` (*str*) – .obj file path

Returns Shapely polygons in a list

Return type list

`livestock.geometry.obj_to_shp (obj_file: str, shp_file: str) → bool`

Convert an .obj file into a shape file.

Parameters

- `obj_file` (*str*) – Path to .obj file
- `shp_file` (*str*) – File path for shapefile

Returns True

Return type bool

`livestock.geometry.shapely_to_pyshp (shapely_geometry: shapely.geometry.polygon.Polygon) → shapefile._Shape`

This function converts a shapely geometry into a geojson and then into a pyshp object.

Copied from Karim Bahgat's answer at:

<https://gis.stackexchange.com/questions/52705/how-to-write-shapely-geometries-to-shapefiles>

Parameters `shapely_geometry` (*shapely.geometry*) – Shapely geometry to convert.

Returns pyshp record object

Return type shapefile._Shape

Go Back to:

[Livestock Frontpage](#)

[Livestock PyPi](#)

[Livestock Grasshopper](#)

1.3 Livestock Hydrology

```
class livestock.hydrology.CMFModel (folder)
    Bases: object

    add_tree (cmf_project, cell_index, property_dict)
        Adds a tree to the model

    config_outputs (cmf_project)
        Function to set up result gathering dictionary

    configure_cells (cmf_project: cmf.cmf_core.project, cell_properties_dict: dict)
        Configure the cells

    create_boundary_conditions (cmf_project)

    create_stream (shape, shape_param, outlet)
        Create a stream

    create_weather (cmf_project)
        Creates weather for the project

    gather_results (cmf_project, time)

    load_cmf_files (delete_after_load=False)

    mesh_to_cells (cmf_project, mesh_path, delete_after_load=True)
        Takes a mesh and converts it into CMF cells :param mesh_path: Path to mesh .obj file :param cmf_project:
        CMF project object. :param delete_after_load: If True, it deletes the input files after they have been loaded.
        :return: True

    print_solver_time (solver_time, start_time, last_time, step)

    run_model ()
        Runs the model with everything

    save_results ()
        Saves the computed results to a xml file

    set_vegetation_properties (cell_: cmf.cmf_core.Cell, property_dict: dict)

    solve (cmf_project, tolerance)
        Solves the model

livestock.hydrology.cell_results (looking_for, result_file, folder)
    Processes cell results

livestock.hydrology.cmf_results (path)

livestock.hydrology.convert_cmf_points (points)

livestock.hydrology.layer_results (looking_for, result_file, folder)
    Processes layer results

livestock.hydrology.surface_flux_results (path)
```

Go Back to:

[Livestock Frontpage](#)

[Livestock PyPi](#)

[Livestock Grasshopper](#)

1.4 Livestock Misc

`livestock.misc.run_cfd(files_path)`

Runs a OpenFoam case

Go Back to:

[Livestock Frontpage](#)

[Livestock PyPi](#)

[Livestock Grasshopper](#)

1.5 Livestock SSH

`livestock.ssh.check_for_remote_folder(sftp_connect: <function SSHClient.open_sftp at 0x0000025AD7132620>, folder_to_check: str, check_for: str) → bool`

Checks if remote folder exists in the desired location. If do exists the function returns True. Otherwise is creates the folder and then returns True.

Parameters

- **sftp_connect** (`paramiko.SSHClient().open_sftp()`) – SFTP connection
- **folder_to_check** (`str`) – Path where there should be looked.
- **check_for** (`str`) – Folder, which existence is wanted.

Returns True on success

Return type bool

`livestock.ssh.ssh_connection()`

This function opens up a SSH connection to a remote machine (Ubuntu-machine) based on inputs from the `in_data.txt` file. Once it is logged in then function activates the anaconda environment `livestock_env`, sends the commands, awaits their completion (by looking for a `out.txt` file, which is only written upon completion of the commands) and returns the wanted files back to the local machine.

Go Back to:

[Livestock Frontpage](#)

[Livestock PyPi](#)

[Livestock Grasshopper](#)

DOCUMENTATION FOR THE GRASSHOPPER COMPONENTS:

- **Livestock Grasshopper Documentation**
 - Components
 - Component Classes
 - Component Library

Go Back to:

[Livestock Frontpage](#)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

I

livestock.air, [3](#)
livestock.geometry, [8](#)
livestock.hydrology, [9](#)
livestock.misc, [10](#)
livestock.ssh, [10](#)

INDEX

add_tree() (livestock.hydrology.CMFModel method), 9

cell_results() (in module livestock.hydrology), 9

celsius_to_kelvin() (in module livestock.air), 3

centroid_z() (in module livestock.geometry), 8

check_for_remote_folder() (in module livestock.ssh), 10

cmf_results() (in module livestock.hydrology), 9

CMFModel (class in livestock.hydrology), 9

compute_temperature_relative_humidity() (in module livestock.air), 3

config_outputs() (livestock.hydrology.CMFModel method), 9

configure_cells() (livestock.hydrology.CMFModel method), 9

convert_cmf_points() (in module livestock.hydrology), 9

convert_relative_humidity_to_percentage() (in module livestock.air), 3

convert_relative_humidity_to_unitless() (in module livestock.air), 4

convert_vapour_flux_to_kgh() (in module livestock.air), 4

create_boundary_conditions() (livestock.hydrology.CMFModel method), 9

create_stream() (livestock.hydrology.CMFModel method), 9

create_weather() (livestock.hydrology.CMFModel method), 9

diameter_from_area() (in module livestock.air), 4

gather_results() (livestock.hydrology.CMFModel method), 9

kelvin_to_celsius() (in module livestock.air), 4

latent_heat_flux() (in module livestock.air), 4

layer_results() (in module livestock.hydrology), 9

livestock.air (module), 3

livestock.geometry (module), 8

livestock.hydrology (module), 9

livestock.misc (module), 10

livestock.ssh (module), 10

load_cmf_files() (livestock.hydrology.CMFModel method), 9

max_possible_vapour_flux() (in module livestock.air), 4

mesh_to_cells() (livestock.hydrology.CMFModel method), 9

new_mean_relative_humidity() (in module livestock.air), 5

new_mean_temperature() (in module livestock.air), 5

new_mean_vapour_pressure() (in module livestock.air), 5

new_temperature_and_relative_humidity() (in module livestock.air), 6

obj_to_lists() (in module livestock.geometry), 8

obj_to_polygons() (in module livestock.geometry), 8

obj_to_shp() (in module livestock.geometry), 8

print_solver_time() (livestock.hydrology.CMFModel method), 9

relative_humidity_to_vapour_pressure() (in module livestock.air), 6

run_cfd() (in module livestock.misc), 10

run_model() (livestock.hydrology.CMFModel method), 9

run_row() (in module livestock.air), 6

saturated_vapour_pressure() (in module livestock.air), 6

save_results() (livestock.hydrology.CMFModel method), 9

set_vegetation_properties() (livestock.hydrology.CMFModel method), 9

shapely_to_pyshp() (in module livestock.geometry), 8

solve() (livestock.hydrology.CMFModel method), 9

ssh_connection() (in module livestock.ssh), 10

stratification() (in module livestock.air), 6

surface_flux_results() (in module livestock.hydrology), 9

vapour_pressure_to_relative_humidity() (in module livestock.air), 7

wind_speed_to_flux() (in module livestock.air), 7

wind_speed_to_hour_flux() (in module livestock.air), 7