Livestock CPython Package Documentation

Release 2018.3

Christian Kongsgaard

Mar 07, 2018

CONTENTS

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.

CHAPTER

ONE

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 m^3 :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 m^3/day to kg/h Density of water: 1000kg/m^3 Hours per day: 24h/day Conversion: 1000kg/m^3/ 24h/day

Parameters **vapour_flux** (*numpy.array*) – Vapour flux in m^3/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 = π * $(d/2)^2$ => 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 m^3 :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 m^3
- **`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 m^3
- **`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 m^3
- **`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 m^3/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 m^3/h m/s to m^3/s = 1:sup:2 m^3/s to m^3/h = 3600s/h

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

Returns Wind flux in m^3/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 0x0000028D8561B840>, 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

CHAPTER

TWO

DOCUMENTATION FOR THE GRASSHOPPER COMPONENTS:

Go Back to:

Livestock Frontpage

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

INDEX