ESPy

Generated by Doxygen 1.8.13

Contents

Chapter 1

espy

Python API for ESP-r

Please note that these modules are primarily designed for my own use, and functions may change inputs or names without warning or concern for maintaining compatibility with any scripts you may have.

Install procedure

```
git clone https://github.com/johnallison0/espy.git
cd espy
python setup.py install
```

Structure

ESPy is broken up into modules. Many of these echo names of ESP-r modules (e.g. bps, res, clm), that contain functions to automate functionality of these modules. Other modules contain various support facilities, as well as functions for interacting with ESP-r models without using the ESP-r interface.

For full namespace documentation, refer to the ./doc directory.

Usage

If the installation procedure above was followed, then ESPy modules should be included in Python code in the same manner as any other installed modules. A minimal workflow for a typical simulation and results extraction task might be the following:

```
import espy.bps as bps
from espy.res import time_series

bps.run_preset('./cfg/model.cfg','annual')
time_series('./cfg/model.cfg','annual.res',[['all', 'Zone db T']],'res.csv')
```

You would then have dry bulb temperature for all zones, in file res.csv. Alternatively, you could work with a Data↩ Frame of the results:

```
import pandas as pd
from espy.res import time_series

res_df = time_series('./cfg/model.cfg','annual.res',[['all', 'Zone db T']])
res_df['Zone1_dbT'].to_csv('Zone1res.csv')
```

There are many other functions provided by ESPy. For full documentation, refer to the ./doc directory.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 espy.bps Namespace Reference

Classes

- class Bps

Functions

- def run_preset (cfg_file, preset)
- def run_sim (cfg_file, res_file, sim_start_d, sim_start_m, sim_end_d, sim_end_m, start_up_d, tsph, integrate)

4.1.1 Detailed Description

```
Functions to interact with bps.
```

4.1.2 Function Documentation

4.1.2.1 run_preset()

```
def espy.bps.run_preset (
              cfg_file,
              preset )
```

```
Run simulation with preset.
```

4.1.2.2 run_sim()

```
def espy.bps.run_sim (
              cfg_file,
              res_file,
              sim_start_d,
              sim_start_m,
              sim_end_d,
              sim_end_m,
              start_up_d,
              tsph,
              integrate )
```

Run basic simulation.

4.2 espy.clm Namespace Reference

Functions

- def get_avg_degree_days (weather_file, temp_base=15.5)
- def epw_to_espr (epw_file, espr_file="newclim")
- def weather_bin_to_ascii (bin_file, ascii_file="newclim.a")

4.2.1 Detailed Description

Functions to interact with clm.

4.2.2 Function Documentation

4.2.2.1 epw_to_espr()

```
def espy.clm.epw_to_espr (
              epw_file,
              espr_file = "newclim" )
```

Convert EPW file to ESP-r binary weather file.

4.2.2.2 get_avg_degree_days()

```
def espy.clm.get_avg_degree_days (
              weather_file,
              temp_base = 15.5 )
```

Returns the daily average degree days

4.2.2.3 weather_bin_to_ascii()

```
def espy.clm.weather_bin_to_ascii (
              bin_file,
              ascii_file = "newclim.a" )
```

Convert ESP-r binary weather file to ascii file.

4.3 espy.edit Namespace Reference

Functions

- def door_usage (geo_file, original, updated)
- def window_usage (geo_file, original, updated)
- def frame_usage (geo_file, original, updated)

4.3.1 Detailed Description

Functions that directory edit ESP-r files.

4.3.2 Function Documentation

4.3.2.1 door_usage()

```
def espy.edit.door_usage (
              geo_file,
              original,
              updated )
```

Directly edit door usage in geometry file.

4.3.2.2 frame_usage()

```
def espy.edit.frame_usage (
              geo_file,
              original,
              updated )
```

Directly edit frame usage in geometry file.

4.3.2.3 window_usage()

```
def espy.edit.window_usage (
              geo_file,
              original,
              updated )
```

Directly edit window usage in geometry file.

4.4 espy.get Namespace Reference

Functions

- def zone_selection (cfg_file, zone_input)
- def surface_selection (geo_file, surf_input)
- def config (filepath)
- def geometry (filepath)
- def constructions (con_file, geo_file)
- def controls (filepath)
- def pos_from_vert_num_list (vertices_zone, edges)
- def weather (file_path)
- def weather_v2 (file_path)
- def zone_to_predef_entity (geo_file, name, desc, category)

4.4.1 Detailed Description

Functions for importing and reading ESP-r files

4.4.2 Function Documentation

4.4.2.1 config()

```
def espy.get.config (
              filepath )
```

Reads in an ESP-r configuration file.

4.4.2.2 constructions()

```
def espy.get.constructions (
              con_file,
              geo_file )
```

Get data from construction file.

4.4.2.3 controls()

```
def espy.get.controls (
              filepath )
```

Import model controls.

4.4.2.4 geometry()

```
def espy.get.geometry (
              filepath )
```

Reads in an ESP-r geometry file.

Returns the name and description of the zone.

Returns the last modified date.

Returns a list of the vertices, where each element is a list of floats
specifying the x, y, z coordinate in space.

Returns a list of the surface edges, where each element is a list of ints
specifying the vertex numbers that make up the surface.
Note that these are referenced as 1-indexed.

Returns a list of the surface attributes, where each element is:
['surf name', 'surf position', 'child of (surface name)',
'useage1', 'useage2', 'construction name', 'optical name',
'boundary condition', 'dat1', 'dat2']

4.4.2.5 pos_from_vert_num_list()

```
def espy.get.pos_from_vert_num_list (
              vertices_zone,
              edges )
```

Get x,y,z position of vertices that comprise a surface
from the zone vertices and their indices as defined in
the edges list

4.4.2.6 surface_selection()

```
def espy.get.surface_selection (
              geo_file,
              surf_input )
```

Maps requested surface selection to ESP-r menu selection.

4.4.2.7 weather()

```
def espy.get.weather (
              file_path )
```

Read ESP-r ascii weather file.

```
col 1: Diffuse solar on the horizontal (W/m^2)
col 2: External dry bulb temperature   (Tenths °C)
col 3: Direct normal solar intensity   (W/m^2)
col 4: Prevailing wind speed           (Tenths m/s)
col 5: Wind direction                  (clockwise ° from north)
col 6: Relative humidity               (%)
```

4.4.2.8 weather_v2()

```
def espy.get.weather_v2 (
              file_path )
```

Read ESP-r ascii weather file.

4.4.2.9 zone_selection()

```
def espy.get.zone_selection (
              cfg_file,
              zone_input )
```

Maps requested zone selection to ESP-r menu selection.

4.4.2.10 zone_to_predef_entity()

```
def espy.get.zone_to_predef_entity (
              geo_file,
              name,
              desc,
              category )
```

```
Convert a zone geometry file to a predefined entity entry.

Args:
    geo_file: ESP-r geometry file.

Returns:
    A text file that can be copied into an ESP-r predefined entities
    database.
```

4.5 espy.plot Namespace Reference

Classes

- class Component

Functions

- def set_axes_radius (ax, origin, radius)
- def set_axes_equal (ax)
- def set_axes_limits (ax, lims)
- def cuboid_data (o, size=(1, 1, 1))
- def plot_cuboid (pos=(0., 0., 0.), size=(1., 1., 1.), ax=None, kwargs)
- def plot_predef_ents (vis, vertices)
- def plot_zone_surface (vertices, ax=None, facecolour=None, alpha=0.2)
- def plot_zone (geo_file, ax=None, show_roof=True)
- def plot_construction (con_data, vertices_surf, ax=None)
- def construction_schematic (constr_name, constr_data, air_gap_data, figsize=(3.54, 2.655), savefig=False)
- def plot_zone_constructions (con_file, geo_file, ax=None)
- def plot_building_component (geo_file, con_file, idx_surface, ax=None, show_roof=True)
- def vtk_view (actors, edge_actors, outlines)
- def generate_vtk_actors (surf_obj, outer_colour, show_edges=False, show_outline=True)
- def is_point_in_surf (point, verts, tol=0.0001)
- def get_outer_inner (verts, add_intermediate=True)
- def dist (v1, v2)
- def insert_edge (verts, v1, v2, insert_v1=False, insert_v2=True)
- def normalized (X)
- def calculate_plane_intersect (A, B)
- def calculate_normal (p)

4.5.1 Detailed Description

```
@package plot
Functions for visualising ESP-r models.

Note that the pyplot functions in here are quick, but 3D plots may not display
correctly due to intrinsic limitations of matplotlib.

Efforts have been made to implement equivalent functionality in VTK, which is
slower but more robust, and requires an OpenGL implementation.
This is a work in progress.
```

4.5.2 Function Documentation

4.5.2.1 calculate_normal()

```
def espy.plot.calculate_normal (
              p )
```

```
Calculate normal of polygon.

Newell's method for calculating the normal of an arbitrary 3D polygon.

Arguments
    p: list, list (3), float
        polygon vertex coordinates

Returns
    nn: list (3), float
        normal direction vector
```

4.5.2.2 calculate_plane_intersect()

```
def espy.plot.calculate_plane_intersect (
              A,
              B )
```

```
Calculates a line at the intersection of two planes.

Code adapted from:
https://gist.github.com/marmakoide/79f361dd613f2076ece544070ddae6ab

Arguments
    A: list (4), float
        plane equation [a,b,c,d] where ax + by + cz + d = 0
    B: list (4), float
        as A

Returns
    U: list (3), float
        direction vector of intersection line
    numpy.ndarray (3), float
        coordinates of a point on the line.
```

4.5.2.3 construction_schematic()

```
def espy.plot.construction_schematic (
              constr_name,
              constr_data,
              air_gap_data,
              figsize = (3.54, 2.655),
              savefig = False )

Plot 2D construction schematic.

If the figure is saved, uses wand module (ImageMagick)
to trim whitespace from the image.

Arguments
    constr_name: string
        name of construction
    constr_data: list, list
        construction data
        output from get.constructions(...)["layer_therm_props"]
    air_gap_data: list, list
        air gap data
        output from get.constructions(...)["air_gap_props"]
    figsize: list or tuple (2), float
        length and height of figure in inches
    savefig: boolean
        if True save figure in images directory
        if False display plot and pause
        optional, default False

Returns
    None
```

4.5.2.4 cuboid_data()

```
def espy.plot.cuboid_data (
              o,
              size = (1, 1, 1) )

Calculate cuboid data from origin and size.

Code taken from:
https://stackoverflow.com/a/35978146/4124317

Arguments
    o: list or tuple (3), float
        coordinates of cuboid origin
        e.g. [0., 0., 0.]
    size: list or tuple (3), float
        size of cuboid
        optional, default (1, 1, 1)

Returns
    numpy.ndarray, float
        x coordinates of coboid vertices
    numpy.ndarray, float
        y coordinates of cuboid vertices
    numpy.ndarray, float
        z coordinates of cuboid vertices
    float
        largest face area
    float
        cuboid volume
    float
        thickness (volume / largest face area)
```

4.5.2.5 dist()

```
def espy.plot.dist (
              v1,
              v2 )

Get distance between two 3D points v1 and v2.

Arguments
    v1: list (3), float
        first point coordinates
        e.g. [1., 1., 0.]
    v2: list (3), float
        second point coordinates

Returns
    float
        distance between the two points
```

4.5.2.6 generate_vtk_actors()

```
def espy.plot.generate_vtk_actors (
              surf_obj,
              outer_colour,
              show_edges = False,
              show_outline = True )

Generates 3 VTK actors.

Returns 3 VTK actors, which represents an object in a rendered scene.

Arguments
    surf_obj: vtk.vtkPolyDataAlgorithm
        vtkObject that defines the surface
    outer_colour: list
        Colour and opacity of surface
        e.g. ["#f5f2d0", 1]
    show_edges: boolean
        If True show edges of triangle mesh
        optional, default False
    show_outline: boolean
        If True show surface outline
        optional, default True

Returns
    surface_actor: vtk.vtkOpenGLActor
        2D component surface projected on 3D plane
    edge_actor: vtk.vtkOpenGLActor
        surface mesh edges
    outline_actor: vtk.vtkOpenGLActor
        boundary outline of surface

Example
    geo = get.geometry(geometry_file)
    for comp in geo['components']:
        sa,ea,oa = plot.generate_vtk_actors(
            comp.generate_vtk_surface(),
            comp.set_outer_colour())
        sas.append(sa)
        eas.append(ea)
        oas.append(oa)
    plot.vtk_view(sas,eas,oas)
```

4.5.2.7 get_outer_inner()

```
def espy.plot.get_outer_inner (
              verts,
              add_intermediate = True )
```

```
Separate weakly simple polygons into outer and inner.

Process list of vertices, removing duplicates and
separating outer and inner polygons into lists.
Optionally adds intermediate vertices using insert_edge.

Arguments
    verts: list, list (3), float
        list of vertex coordinates
        e.g. [[0., 0., 0.], [1., 0., 0], ...]
    add_intermediate: boolean
        inserts intermediate vertices if True
        optional, default True

Returns
    outer: list, list (3), float
        outer polygon vertex coordinates
    inners: list, list, list (3), float
        vertex coordinates for each inner polygon
```

4.5.2.8 insert_edge()

```
def espy.plot.insert_edge (
              verts,
              v1,
              v2,
              insert_v1 = False,
              insert_v2 = True )
```

```
Insert the edge between vertices v1 and v2 into vertices list.

Also inserts intermediate vertices, ensuring that no edge is
longer than 1m.

Arguments
    verts: list, list (3), float
        list of vertices to insert edge into
    v1: list (3), float
        coordinates for start vertex of edge
    v2: list (3), float
        coordinates for end vertex of edge
    insert_v1: boolean
        inserts v1 into verts if True
        optional, default False
    insert_v2: boolean
        inserts v2 into verts if True
        optional, default True

Returns
    None (modifies verts in-place)
```

4.5.2.9 is_point_in_surf()

```
def espy.plot.is_point_in_surf (
              point,
              verts,
              tol = 0.0001 )
```

```
Checks if point is in a list of vertices within a tolerance

Arguments
    point: list (3), float
        coordinates of point to check
        e.g. [1., 1., 0.]
    verts: list, list (3), float
        list of vertex point coordinates
    tol: float
        tolerance of total distance
        optional, default 0.0001

Returns
    boolean
        True if point is found in verts
    i: integer or None
        index of existing vertex if point found
```

4.5.2.10 normalized()

```
def espy.plot.normalized (
              X )
```

```
Normalises magnitude of vector.

Code adapted from:
https://gist.github.com/marmakoide/79f361dd613f2076ece544070ddae6ab

Arguments
    X: list (3), float
        vector

Returns
    float
        normalised vector
```

4.5.2.11 plot_building_component()

```
def espy.plot.plot_building_component (
              geo_file,
              con_file,
              idx_surface,
              ax = None,
              show_roof = True )
```

```
Plot a particular surface construction in 3D.

This function plots a 3D building component (wall, floor, roof etc.)
from its surface geometry and construction data.

The inside surface is plotted as white, while the external surface colour
is dependent on the surface properties from the geometry file.

Arguments
    geo_file: string
        zone geometry file name
    con_file: string
        zone construction file name
    idx_surface: integer
        surface index
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    show_roof: boolean
        if True show roof surface(s)
        optional, default True

Returns
    None
```

4.5.2.12 plot_construction()

```
def espy.plot.plot_construction (
              con_data,
              vertices_surf,
              ax = None )
```

```
Plot 3D construction.

Arguments
    con_data: list, list
        construction data
        output from get.constructions(...)["layer_therm_props"]
    vertices_surf: list, list (3), float
        list of vertex coordinates
        e.g. [[0., 0., 0.],[0., 1., 0.],...]
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()

Returns
    None
```

4.5.2.13 plot_cuboid()

```
def espy.plot.plot_cuboid (
              pos = (0., 0., 0.),
              size = (1., 1., 1.),
              ax = None,
              kwargs )
```

```
Plot a cuboid element.

Arguments
    pos: list or tuple (3), float
        cuboid origin coordinates
        optional, default (0., 0., 0.)
    size: list or tuple (3), float
        cuboid size
        optional, default (1., 1., 1.)
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    **kwargs
        additional arguments to be forwarded to matplotlib.pyplot.plot_surface

Returns
    None
```

4.5.2.14 plot_predef_ents()

```
def espy.plot.plot_predef_ents (
              vis,
              vertices )


Plot predefined entity.

Create 3D figure and plot visual entities and mass surfaces,
for example for predefined entity, with pyplot.
Displays plot and will pause until user closes it.

Arguments
    vis: list, list (6), float
        list of cuboid visual entities
        e.g. [[origin_x, origin_y, origin_z, size_x, size_y, size_z],...]
    vertices: list, list, list (3), float
        list of vertices for mass surfaces e.g.
        e.g. [[[0.,0.,0.],[0.,1.,0.],[1.,1.,0.],[1.,0.,0.]],...]

Returns
    None
```

4.5.2.15 plot_zone()

```
def espy.plot.plot_zone (
              geo_file,
              ax = None,
              show_roof = True )


Plot zone geometry with pyplot.

Arguments
    geo_file: string
        name of zone geometry file
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    show_roof: boolean
        If True show roof surface(s)
```

```
        optional, default True

Returns
    None

Example
    fig = plt.figure()
    ax = fig.gca(projection='3d')
    plot.plot_zone(geo_file, ax=ax)
    plt.show()
```

4.5.2.16 plot_zone_constructions()

```
def espy.plot.plot_zone_constructions (
              con_file,
              geo_file,
              ax = None )

Plot all zone constructions in 3D.

Arguments
    con_file: string
        zone construction file name
    geo_file: string
        zone geometry file name
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()

Returns
    None
```

4.5.2.17 plot_zone_surface()

```
def espy.plot.plot_zone_surface (
              vertices,
              ax = None,
              facecolour = None,
              alpha = 0.2 )

Plots a surface with pyplot.

Arguments
    vertices: list, list (3), float
        vertex coordinates
        e.g. [[0., 0., 0.],[0., 1., 0.],...]
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    facecolour: string
        surface colour hash code
        e.g. '#c19a6b'
        optional, default None (i.e. white)
    alpha: float
        opacity
        0.0 - 1.0
        optional, default 0.2

Returns
    None
```

4.5.2.18 set_axes_equal()

```
def espy.plot.set_axes_equal (
              ax )
```

```
Set all axes equal.

Make axes of 3D plot have equal scale so that spheres appear as spheres,
cubes as cubes, etc..  This is one possible solution to Matplotlib's
ax.set_aspect('equal') and ax.axis('equal') not working for 3D.

Arguments
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()

Returns
    None (modifies ax in-place)
```

4.5.2.19 set_axes_limits()

```
def espy.plot.set_axes_limits (
              ax,
              lims )
```

```
Set axis limits.

Call set_axes_equal after this function to ensure objects display correctly.

Arguments
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    lims: list (3), list (2), float
        e.g. [[xmin,xmax],[ymin,ymax],[zmin,zmax]]

Returns
    None (modifies ax in-place)
```

4.5.2.20 set_axes_radius()

```
def espy.plot.set_axes_radius (
              ax,
              origin,
              radius )
```

```
Set axes radius.

Arguments
    ax: matplotlib.axes.Axes
        e.g. output from plt.gca()
    origin: list or tuple (3), float
        axes origin coordinates
        e.g. [0.,0.,0.]
    radius: float
        axes radius

Returns
    None (modifies ax in-place)
```

4.5.2.21 vtk_view()

```
def espy.plot.vtk_view (
              actors,
              edge_actors,
              outlines )
```

```
VTK visualisation setup and render.

Arguments
    actors: list, vtk.vtkOpenGLActor
        surface actors
        elements are output from generate_vtk_actors(...)[0]
    edge_actors: list, vtk.vtkOpenGLActor
        mesh edge actors
        elements are output from generate_vtk_actors(...)[1]
    outlines: list, vtk.vtkOpenGLActor
        outline actors
        elements are output from generate_vtk_actors(...)[2]

Returns
    None

Example
    geo = get.geometry(geometry_file)
    for comp in geo['components']:
        sa,ea,oa = plot.generate_vtk_actors(
            comp.generate_vtk_surface(),
            comp.set_outer_colour())
        sas.append(sa)
        eas.append(ea)
        oas.append(oa)
    plot.vtk_view(sas,eas,oas)
```

4.6 espy.prj Namespace Reference

Functions

- def edit_material_prop (cfg_file, change_list)
- def edit_layer_thickness (cfg_file, change_list)
- def gen_qa_report (cfg_file, filename)
- def rebuild_con_files (cfg_file)
- def add_door (cfg_file, door_name, zone_surf1, zone_surf2, x_off, size)
- def add_window (cfg_file, zone, surf, location, size, sill=None, reveal=None)
- def add_zone (cfg_file, name, vertices, description=None, z_base=0, z_top=2.7, rot_angle=0)

4.6.1 Detailed Description

```
Functions to interact with prj.
```

4.6.2 Function Documentation

4.6.2.1 add_door()

```
def espy.prj.add_door (
              cfg_file,
              door_name,
              zone_surf1,
              zone_surf2,
              x_off,
              size )
```

Adds door between two zones.

4.6.2.2 add_window()

```
def espy.prj.add_window (
              cfg_file,
              zone,
              surf,
              location,
              size,
              sill = None,
              reveal = None )
```

Adds window to a surface in a zone.

4.6.2.3 add_zone()

```
def espy.prj.add_zone (
              cfg_file,
              name,
              vertices,
              description = None,
              z_base = 0,
              z_top = 2.7,
              rot_angle = 0 )
```

Adds new zone to model.

4.6.2.4 edit_layer_thickness()

```
def espy.prj.edit_layer_thickness (
              cfg_file,
              change_list )
```

```
Edit layer thickness of multi-layered construction.
This function will build the command list to edit the layer thickness in
the MLC db via prj.
```

4.6.2.5 edit_material_prop()

```
def espy.prj.edit_material_prop (
              cfg_file,
              change_list )
```

```
Edit material properties.
This function will build the command list to edit material properties in
the materials db via prj.
```

4.6.2.6 gen_qa_report()

```
def espy.prj.gen_qa_report (
              cfg_file,
              filename )
```

```
Generate model QA report.
```

4.6.2.7 rebuild_con_files()

```
def espy.prj.rebuild_con_files (
              cfg_file )
```

```
Updates the zone construction files.
```

4.7 espy.res Namespace Reference

Functions

- def calc_airtightness (res_file, mfr_file, volume, zones)
- def air_supply (res_file, mfr_file, zones)
- def time_series (cfg_file, res_file, param_list, out_file=None, time_fmt='DateTime')
- def abovebelow (cfg_file, res_file, is_below=False, out_file=None, query_point=25)
- def energy_balance (cfg_file, res_file, out_file=None, group=None)
- def get_pv (res_file, elr_file, out_file=None)

4.7.1 Detailed Description

Module to automate retrieval of data from res.

4.7.2 Function Documentation

4.7.2.1 abovebelow()

```
def espy.res.abovebelow (
              cfg_file,
              res_file,
              is_below = False,
              out_file = None,
              query_point = 25 )
```

Get hours above or below a value.

4.7.2.2 air_supply()

```
def espy.res.air_supply (
              res_file,
              mfr_file,
              zones )
```

Retreive air supply from ambient to zones.

```
Args:
    res_file: ESP-r results database.
    mfr_file: ESP-r mass flow results database.
    zones: List of strings with zones to include e.g.
        zones = ["a", "b"] to get air flow from those air flow nodes

Returns:
    df: Pandas dataframe with volume flow rate to/from ambient per zone.
```

4.7.2.3 calc_airtightness()

```
def espy.res.calc_airtightness (
              res_file,
              mfr_file,
              volume,
              zones )
```

```
Calculate building airtightness at 50 Pa.

Args:
    res_file: ESP-r results database.
    mfr_file: ESP-r mass flow results database.
    volume: Heated volume of building (m^3).
    zones: List of strings with zones to include e.g.
        zones = ["a", "b"] to get air flow from those air flow nodes

Returns:
    n_50: Air change rate (1/h)
    q_50: Air permeability (m^3/(h.m^2))
    w_50: Specific leakage rate (m^3/(h.m^2))
```

4.7.2.4 energy_balance()

```
def espy.res.energy_balance (
              cfg_file,
              res_file,
              out_file = None,
              group = None )
```

```
Get zone energy balance.
```

4.7.2.5 get_pv()

```
def espy.res.get_pv (
              res_file,
              elr_file,
              out_file = None )
```

```
Get PV output.
```

4.7.2.6 time_series()

```
def espy.res.time_series (
              cfg_file,
              res_file,
              param_list,
              out_file = None,
              time_fmt = 'DateTime' )
```

```
Extract results from results database to CSV.

Args:
    cfg_file: ESP-r configuration file.
    res_file: ESP-r results database.
    param_list: List of parameters to extract.
        Examples -
        param_list = [['all', 'Zone db T']]
        param_list = [['id:reception', 'Zone db T']]
        param_list = [[['id:roof_space', 'id:reception'], 'Zone db T']]
        param_list = [[['a', 'b'], 'Zone db T'], [['id:reception', 'b'], 'Wind direction']]
    out_file (optional): Name of exported CSV file.
    time_fmt (optional): Format of datetime in exported CSV. Julian or DateTime, default DateTime.

Returns:
    res: DataFrame containing results.
```

4.8 espy.utils Namespace Reference

Functions

- def **header** (str_in, lvl=0)
- def split_to_float (string)
- def space_data_to_list (item, convert="int")
- def sed (pattern, replace, source, dest=None, count=0)
- def area (poly)
- def dtparse_espr (d)

4.8.1 Detailed Description

```
Low level utilities
```

4.8.2 Function Documentation

4.8.2.1 area()

```
def espy.utils.area (
              poly )
```

```
area of polygon poly
Source: https://stackoverflow.com/a/12643315
Source 2: http://geomalgorithms.com/a01-_area.html#3D%20Polygons
```

4.8.2.2 dtparse_espr()

```
def espy.utils.dtparse_espr (
              d )
```

Parser for esp-r datetime format.
This is needed because days do not match time steps intuitively for Python.

4.8.2.3 sed()

```
def espy.utils.sed (
              pattern,
              replace,
              source,
              dest = None,
              count = 0 )
```

Reads a source file and writes the destination file.

In each line, replaces pattern with replace.

Args:
    pattern (str): pattern to match (can be re.pattern)
    replace (str): replacement str
    source (str): input filename
    count (int): number of occurrences to replace
    dest (str):   destination filename, if not given, source will be over written.

4.8.2.4 space_data_to_list()

```
def espy.utils.space_data_to_list (
              item,
              convert = "int" )
```

Transform space separated data into specified type list

4.8.2.5 split_to_float()

```
def espy.utils.split_to_float (
              string )
```

Transform CSV string into list of floats.

4.9 espy.write Namespace Reference

Functions

- def construction (fout, constr_name, constr_data, air_gap_data, mat_names)
- def img_to_md (fout, img_file, caption)

4.9.1 Detailed Description

```
Write out various files.
```

4.9.2 Function Documentation

4.9.2.1 construction()

```
def espy.write.construction (
              fout,
              constr_name,
              constr_data,
              air_gap_data,
              mat_names )
```

```
Write out construction data in markdown format.

Args:
    constr_name: str
        Construction name.

    constr_data: list
        List of construction data layers and thermophysical properties.

    air_gap_data: list
        List of air gap locations and properties.

    mat_names: list
        List of str of length N with name of each material layer.

Returns:
    out_file: str
        Filename of open out file.
```

4.9.2.2 img_to_md()

```
def espy.write.img_to_md (
              fout,
              img_file,
              caption )
```

```
Generate markdown format image text.
```

Chapter 5

Class Documentation

5.1 espy.bps.Bps Class Reference

Public Member Functions

- def **__init__** (self)
- def **__del__** (self)

Static Public Attributes

- int **counter** = 0

5.1.1 Detailed Description

```
Instance of BPS.
```

The documentation for this class was generated from the following file:

- espy/bps.py

5.2 espy.plot.Component Class Reference

Public Member Functions

- def __init__ (self, property_list, child_verts, vertices_surf)
- def generate_vtk_surface (self)
- def set_outer_colour (self)

Public Attributes

- **name**
- **position**
- **child**
- **usage**
- **construction**
- **optical_type**
- **boundary**
- **child_verts**
- **vertices_surf**

5.2.1 Detailed Description

```
Class defining a zone surface.

Properties
    name: string
    position: string
        VERT, CEIL, FLOR, SLOP
    child: string or None
        name of the parent surface
    usage: list (2) or None
        usage tags
    construction: string
    optical_type: string
        OPAQUE or transparent properties
    boundary: string
        other side boundary condition
    child_verts: list, list, list (3), float
        vertex cordinates for each child surface
    vertices_surf: list, list (3), float
        surface vertex coordinates
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def espy.plot.Component.__init__ (
              self,
              property_list,
              child_verts,
              vertices_surf )

Constructor.

Arguments
    property_list: list
surface properties
output from get.geometry(...)["props"])
    child_verts: list, list, list (3), float
child surface vertex coordinates
    vertices_surf: list, list (3), float
surface vertex coordinates

Returns
    Component object

Example
    surface_component = plot.Component(property_list, child_verts, vertices_surf)
```

5.2.3 Member Function Documentation

5.2.3.1 generate_vtk_surface()

```
def espy.plot.Component.generate_vtk_surface (
              self )
```

```
Generate building component surface as a VTK object.

Arguments
    None

Returns
    surf_obj: vtk.vtkPolyDataAlgorithm
suitable for input to generate_vtk_actors(...)

Example
    geo = get.geometry(geometry_file)
    for comp in geo['components']:
sa,ea,oa = plot.generate_vtk_actors(
    comp.generate_vtk_surface(),
    comp.set_outer_colour())
sas.append(sa)
eas.append(ea)
oas.append(oa)
    plot.vtk_view(sas,eas,oas)
```

5.2.3.2 set_outer_colour()

```
def espy.plot.Component.set_outer_colour (
              self )
```

```
Set colour of otherside surface based on boundary conditions.

Arguments
    None

Returns
    surf_colour: list (2)
colour hash code and opacity
suitable for input to generate_vtk_actors(...)
e.g. ["#F8F4FF", 1.]

Example
    geo = get.geometry(geometry_file)
    for comp in geo['components']:
sa,ea,oa = plot.generate_vtk_actors(
    comp.generate_vtk_surface(),
    comp.set_outer_colour())
sas.append(sa)
eas.append(ea)
oas.append(oa)
    plot.vtk_view(sas,eas,oas)
```

The documentation for this class was generated from the following file:

- espy/plot.py