LA-UR-15-21189
Approved for public release; distribution is unlimited.

Title: PyFLOTRAN Documentation Release 1.0.0

Author(s): Karra, Satish
Kitay, Cory

Intended for: Report

Issued: 2015-02-18

PyFLOTRAN Documentation

Release 1.0.0

Satish Karra, Cory Kitay

February 18, 2015

Contents

Contents:

CHAPTER 1

Introduction

PyFLOTRAN is a set of classes and methods that the enable use of the massively parallel subsurface flow and reactive transport code PFLOTRAN (http://www.pflotran.org) within the Python scripting environment. This allows the use of a wide range of libraries available in Python for pre- and post- processing. The main advantages of using PyFLOTRAN include

1. PFLOTRAN input file construction from Python environment. Reduces the need to learn all the keywords in PFLOTRAN. (see Chapter *3*).
2. Post-processing of output using matplotlib as well as including Python-based commandline calls to the open-source visualization toolkits such as VisIt and Paraview.
3. Scripting tools that supports Python's built-in multi-processing capabilities for batch simulations. This reduces the time spent in creating separate input files either for multiple realizations or multiple simulations for comparison.
4. Streamlined workflow from pre-processing to post-processing. Typical workflow with PFLOTRAN involves – pre-processing in Python or MATLAB, writing input files, executing the input files and then post-processing using matplotlib, VisIt or Paraview. PyFLOTRAN allows users to perform all these steps using one Python script.

Introductory tutorials (see Chapter *5*) to PyFLOTRAN are also provided.

1.1 Installation

1.1.1 Python

PyFLOTRAN is supported on Python 2.6 and 2.7, but NOT 3.x. Instructions for downloading and installing Python can be found at http://www.python.org. PyFLOTRAN requires NumPy, SciPy, Matplotlib modules to be installed

1.1.2 PyFLOTRAN

A download link for the latest release version of PyFLOTRAN can be found at http://www.pyflotran.lanl.gov

To install, download and extract the zip file, and run the setup script at the command line:

```
python setup.py install
```

1.1.3 PFLOTRAN

PFLOTRAN (http://www.pflotran.org) is a massively parallel subsurface flow and reactive transport code. PFLOTRAN solves a system of partial differential equations for multiphase, multicomponent and multiscale reactive flow and transport in porous media. The code is designed to run on leadership-class supercomputers as well as workstations and laptops.

For successfully using PyFLOTRAN, one needs to install PFLOTRAN. For details to install PFLOTRAN please see the wikipage: https://bitbucket.org/pflotran/pflotran-dev/wiki/Home

1.1.4 VisIt

VisIt is a parallel, open-source visualisation software. PFLOTRAN can output in .h5 and .xmf format. These can be imported in VisIt and visualization can be performed.

Instructions for downloading and installing VisIt can be found at https://wci.llnl.gov/codes/visit/download.html

1.1.5 Paraview

Paraview is a parallel, open-source visualisation software. PFLOTRAN can output in .xmf and .vtk format. These can be imported in Paraview and visualization can be performed.

Instructions for downloading and installing Paraview can be found at http://www.paraview.org

1.2 Import PyFLOTRAN

PyFLOTRAN consists of several Python modules. To access their functionality, the user must include the following line at the top of any Python script

```
from pdata import*
```

Before doing this, one needs to ensure that pyflotran directory is in the PYTHONPATH. This can be done by configuring cshrc or bashrc files. Alternatively, one can add the PyFLOTRAN path using sys.path.append() in their driver script.

1.3 About thise manual

This manual comprises sections for each of the important PyFLOTRAN modules: *pdata* In these, the important classes and their methods are documented, and example usage provided. Examples can be found in the 'tests' directory of the PyFLOTRAN repository. One can get a feel for setting up, running and visualizing PFLOTRAN simulations (both flow and reactive transport) through these examples.

1.4 Contributors

Greg Lackey, CU Boulder – Added the dataset and simulation related keywords.

1.5 Acknowledgements

PyFLOTRAN was partly developed as part of Cory Kitay's undergraduate internship in the Computational Earth Science Group (EES-16) at the Los Alamos National Laboratory in summer 2014. He was supported through U.S. DOE's Student Undergraduate Laboratory Internship (SULI) program and through LANL LDRD project 20140002DR. David Dempsey's guidance and help in developing PyFLOTRAN is highly appreciated. The motivation behind PyFLOTRAN has been the cool capabilities that Dempsey has developed in PyFEHM (http://pyfehm.lanl.gov).

CHAPTER 2

pdata: PFLOTRAN input file generation

The pdata module is the main module in PyFLOTRAN which contains classes and methods to read, manipulate, write and execute PFLOTRAN input files.

2.1 pdata class

The `pdata` is the main class that does the reading, writing, manipulation and execution of the PFLOTRAN input files. The other classes discussed in this section are defined to increase modularity and are used to set the attributes of `pdata`.

class `pdata.`**`pdata`**(*filename=''*, *work_dir=''*)

Class for pflotran data file. Use 'from pdata import*' to access pdata library

2.1.1 Attributes

Simulation

class `pdata.`**`psimulation`**(*simulation_type=''*, *subsurface_flow=''*, *subsurface_transport=''*, *mode=''*)

Class for specifying simulation type and simulation mode.

Parameters

- **simulation_type** (*str*) – Specify simulation type. Options include: 'surface','subsurface.
- **subsurface_flow** (*str*) – Specify the process model.
- **subsurface_transport** (*str*) – Specify the process model.
- **mode** (*str*) – Specify the mode for the subsurface flow model

Grid

class pdata.**pgrid**(*type='structured', lower_bounds=[0.0, 0.0, 0.0], upper_bounds=[1.0, 1.0, 1.0], origin=[0.0, 0.0, 0.0], nxyz=[10, 10, 10], dx=[], dy=[], dz=[], gravity=[0.0, 0.0, -9.8068], filename=''*)

Class for defining a grid. Used to define type, resolution and geometry of the gird

Parameters

- **type** (*str*) – Grid type. Valid entries include: 'structured', 'unstructured'.
- **lower_bounds** (*[float]*3*) – Lower/Minimum 3D boundaries coordinates in order of x_min, y_min, z_min. Input is a list of 3 floats. e.g., [0.e0, 0.e0, 0.e0].
- **upper_bounds** – Upper/Maximum 3D boundaries coordinates in order of x_max, y_max, z_max. Input is a list of 3 floats. e.g., [321.e0, 1.e0, 51.e0].
- **origin** (*[float]*3*) – Coordinates of grid origin. Optional. Input is a list of 3 floats. Default: [0.e0, 0.e0, 0.e0].
- **nxyz** (*[float]*3*) – Number of grid cells in x,y,z directions. Only works with type='structured'. Input is a list of 3 floats. e.g., [107, 1, 51].
- **dx** (*[float]*) – Specifies grid spacing of structured cartesian grid in the x-direction. e.g., [0.1, 0.2, 0.3, 0.4, 1, 1, 1, 1].
- **dy** (*[float]*) – Specifies grid spacing of structured cartesian grid in the y-direction.
- **dz** (*[float]*) – Specifies grid spacing of structured cartesian grid in the z-direction
- **gravity** (*[float]*3*) – Specifies gravity vector in m/s^2. Input is a list of 3 floats.
- **filename** (*str*) – Specify name of file containing grid information. Only works with type='unstructured'.

Material

class pdata.**pmaterial**(*id=None, name='', porosity=None, tortuosity=None, density=None, specific_heat=None, cond_dry=None, cond_wet=None, saturation='', permeability=[]*)

Class for defining a material property. Multiple material property objects can be created.

Parameters

- **id** (*int*) – Unique identifier of material property.
- **name** (*str*) – Name of material property. e.g., 'soil1'.
- **porosity** (*float*) – Porosity of material.
- **tortuosity** (*float*) – Tortuosity of material.

- **density** (*float*) – Rock density of material in kg/m^3.
- **specific_heat** (*float*) – Specific heat of material in J/kg/K.
- **cond_dry** (*float*) – Thermal dry conductivity of material in W/m/K.
- **cond_wet** (*float*) – Thermal wet conductivity of material in W/m/K.
- **saturation** (*str*) – Saturation function of material property. e.g., 'sf2'
- **permeability** (*[float]*3*) – Permeability of material property. Input is a list of 3 floats. Uses diagonal permeability in unit order: k_xx [m^2], k_yy [m^2], k_zz [m^2]. e.g., [1.e-15,1.e-15,1.e-17].

Time

class pdata.**ptime**(*tf*=[ ], *dti*=[ ], *dtf*=[ ], *dtf_list*=[ ])

Class for time. Used to specify final time of simulation, initial timestep size, maximum timestep size (throughout the simulation or a particular instant of time). Time values and units need to be specified. Acceptable time units are: (s, m, h, d, mo, y).

Parameters

- **tf** (*[float, str]*) – final tim. 1st variable is time value. 2nd variable specifies time unit. e.g., [0.25e0, 'y']
- **dti** (*[float, str]*) – delta (change) time initial a.k.a. initial timestep size. 1st variable is time value. 2nd variable specifies time unit. e.g., [0.25e0, 'y']
- **dtf** (*[float, str]*) – delta (change) time final a.k.a. maximum timestep size. 1st variable is time value. 2nd variable specifies time unit. e.g., [50.e0, 'y']
- **dtf_list** (*[[float, str, float, str]]*) – delta (change) time starting at a given time instant. Input is a list that can have multiple lists appended to it. e.g., time.dtf_list.append([1.e2, 's', 5.e3, 's'])

Uniform Velocity

class pdata.**puniform_velocity**(*value_list*=[ ])

Class for specifiying uniform velocity with transport. Optional with transport problem when not coupling with any flow mode. If not specified, assumes diffusion transport only.

Parameters **value_list** (*[float,float,float,str]*) – List of variables of uniform_velocity. First 3 variables are vlx, vly, vlz in unit [m/s]. 4th variable specifies unit.

e.g., [14.4e0, 0.e0, 0.e0, 'm/yr']

Timestepper

class pdata.**ptimestepper**(*ts_mode='flow'*, *ts_acceleration=None*, *num_steps_after_cut=None*, *max_steps=None*, *max_ts_cuts=None*, *cfl_limiter=None*, *initialize_to_steady_state=False*, *run_as_steady_state=False*, *max_pressure_change=None*, *max_temperature_change=None*, *max_concentration_change=None*, *max_saturation_change=None*)

Class for controling time stepping.

Parameters

- **ts_mode** (*string*) – FLOW or TRAN mode
- **ts_acceleration** (*int*) – Integer for time step acceleration ramp.
- **num_steps_after_cut** (*int*) – Number of time steps after a time step cut that the time step size is held constant.
- **max_steps** (*int*) – Maximum time step after which the simulation will be terminated.
- **max_ts_cuts** (*int*) – Maximum number of consecutive time step cuts before the simulation is terminated.
- **cfl_limiter** (*float*) – CFL number for transport.
- **initialize_to_steady_state** (*bool - True or False*) – Boolean flag to initialize a simulation to steady state
- **run_as_steady_state** (*bool - True or False*) – Boolean flag to run a simulation to steady state
- **max_pressure_change** (*float*) – Maximum change in pressure for a time step. Default: 5.d4 Pa.
- **max_temperature_change** (*float*) – Maximum change in temperature for a time step. Default: 5 C.
- **max_concentration_change** (*float*) – Maximum change in pressure for a time step. Default: 1. mol/L.
- **max_saturation_change** (*float*) – Maximum change in saturation for a time step. Default: 0.5.

Linear Solver

class pdata.**plsolver**(*name=''*, *solver=''*)

Class for specifying linear solver. Multiple linear solver objects can be created one for flow and one for transport.

Parameters

- **name** (*str*) – Specify name of the physics for which the linear solver is being defined. Options include: 'tran', 'transport','flow'.
- **solver** (*str*) – Specify solver type: Options include: 'solver', 'krylov_type', 'krylov', 'ksp', 'ksp_type'

Newton Solver

class pdata.**pnsolver**(*name=''*, *atol=None*, *rtol=None*, *stol=None*, *dtol=None*, *itol=None*, *max_it=None*, *max_f=None*)

Class for newton solver card. Multiple newton solver objects can be created, one for flow and one for transport.

Parameters

- **name** (*str*) – Specify newton solver to use: Options include: 'tran', 'transport', 'tran_solver', 'flow_solver'. Default: 'flow_solver'
- **atol** (*float*) – Absolute tolerance.
- **rtol** (*float*) – Relative tolerance w.r.t previous iteration.
- **stol** (*float*) – Relative tolerance of the update w.r.t previous iteration.
- **dtol** (*float*) – Divergence tolerance.
- **itol** (*float*) – Tolerance compared to infinity norm.
- **max_it** (*int*) – Cuts time step if the number of iterations exceed this value.
- **max_f** (*int*) – Maximum function evaluations (useful with linesearch methods.)

Output

class pdata.**poutput**(*time_list=[]*, *print_column_ids=False*, *screen_periodic=None*, *screen_output=True*, *periodic_time=[]*, *periodic_timestep=[]*, *periodic_observation_time=[]*, *periodic_observation_timestep=None*, *format_list=[]*, *permeability=False*, *porosity=False*, *velocities=False*, *mass_balance=False*, *variables_list=[]*)

Class for dumping simulation output. Acceptable time units (units of measurements) are: 's', 'min', 'h', 'd', 'w', 'mo', 'y'.

Parameters

- **time_list** (*[str, float*]*) – List of time values. 1st variable specifies time unit to be used. Remaining variable(s) are floats.
- **print_column_ids** (*bool - True or False*) – Flag to indicate whether to print column numbers in observation and mass balance output files. Default: False

- **screen_output** – Turn the screen output on/off.
- **screen_periodic** (*int*) – Print to screen every <integer> time steps.
- **periodic_time** (*[float, str]*) – 1st variable is value, 2nd variable is time unit.
- **periodic_timestep** (*[float, str]*) – 1st variable is value, 2nd variable is time unit.
- **periodic_observation_time** (*[float, str]*) – Output the results at observation points and mass balance output at specified output time. 1st variable is value, 2nd variable is time unit.
- **periodic_observation_timestep** – Outputs the results at observation points and mass

balance output at specified time steps. :type periodic_observation_timestep: int :param format_list: Specify the file format for time snapshot of the simulation in

time file type. Input is a list of strings. Multiple formats can be specified. File format options include: 'TECPLOT BLOCK' - TecPlot block format, 'TECPLOT POINT' - TecPlot point format (requires a single processor), 'HDF5' - produces single HDF5 file and xml for unstructured grids, 'HDF5 MULTIPLE_FILES' - produces a separate HDF5 file at each output time, 'VTK' - VTK format.

Parameters

- **permeability** (*bool - True or False*) – Turn permeability output on/off.
- **porosity** (*bool - True or False*) – Turn porosity output on/off.
- **velocities** (*bool - True or False*) – Turn velocity output on/off.
- **mass_balance** (*bool - True or False*) – Flag to indicate whether to output the mass balance of the system.
- **variables_list** (*[str]*) – List of variables to be printed in the output file

Fluid Properties

class pdata.**pfluid**(*diffusion_coefficient=1e-09*)

Class for specifying fluid properties.

Parameters **diffusion_coefficient** (*float*) – Unit of measurement is [m^2/s]. Default: 1e-09

Saturation Function

class pdata.**psaturation**(*name=''*, *permeability_function_type=None*, *saturation_function_type=None*, *residual_saturation=None*, *residual_saturation_liquid=None*, *residual_saturation_gas=None*, *a_lambda=None*, *alpha=None*, *max_capillary_pressure=None*, *betac=None*, *power=None*)

Class for specifying saturation functions.

Parameters

- **name** (*str*) – Saturation function name. e.g., 'sf2'
- **permeability_function_type** (*str*) – Options include: 'VAN_GENUCHTEN', 'MUALEM', 'BURDINE', 'NMT_EXP', 'PRUESS_1'.
- **saturation_function_type** (*str*) – Options include: 'VAN_GENUCHTEN', 'BROOKS_COREY', 'THOMEER_COREY', 'NMT_EXP', 'PRUESS_1'.
- **residual_saturation** (*float*) – MODES: RICHARDS, TH, THC
- **residual_saturation_liquid** (*float*) – MODES: MPHASE
- **residual_saturation_gas** (*float*) – MODES: MPHASE
- **a_lambda** (*float*) – lambda
- **alpha** (*float*) – Pa^-1
- **max_capillary_pressure** (*float*) – Pa
- **betac** (*float*) –
- **power** (*float*) –

Region

class pdata.**pregion**(*name=''*, *coordinates_lower=[0.0, 0.0, 0.0]*, *coordinates_upper=[0.0, 0.0, 0.0]*, *face=None*)

Class for specifying a PFLOTRAN region. Multiple region objects can be created.

Parameters

- **name** (*str*) – Region name.
- **coordinates_lower** (*[float]*3*) – Lower/minimum 3D coordinates for defining a volumetric, planar, or point region between two points in space in order of x1, y1, z1. e.g., [0.e0, 0.e0, 0.e0]
- **coordinates_upper** (*[float]*3*) – Upper/maximum 3D coordinates for defining a volumetric, planar, or point region between two points in space in order of x2, y2, z2. e.g., [321.e0, 1.e0, 51.e0]

- **face** (*str*) – Defines the face of the grid cell to which boundary conditions are connected. Options include: 'west', 'east', 'north', 'south', 'bottom', 'top'. (structured grids only).

Observation

class pdata.**pobservation**(*region=None*)

Class for specifying an observation region. Multiple observation objects may be added. Currently, only region is supported in PyFLOTRAN.

Parameters **region** (*str*) – Defines the name of the region to which the observation object is linked.

Flow

class pdata.**pflow**(*name=''*, *units_list=None*, *iphase=None*, *sync_timestep_with_update=False*, *datum=[]*, *varlist=[]*)

Class for specifying a PFLOTRAN flow condition. There can be multiple flow condition objects.

Parameters

- **name** (*str*) – Name of the flow condition.
- **units_list** (*[str]*) – Not currently supported.
- **iphase** (*int*) –
- **sync_timestep_with_update** (*bool - True or False*) – Flag that indicates whether to use sync_timestep_with_update. Default: False.
- **data_unit_type** (*str*) – List alternative, do not use with non-list alternative attributes/parameters.
- **datum** (*Multiple [float, float, float] or str.*) – Input is either a list of [d_dx, d_dy, d_dz] OR a 'file_name' with a list of [d_dx, d_dy, d_dz]. Choose one format type or the other, not both. If both are used, then only the file name will be written to the input deck.
- **varlist** (*[pflow_variable]*) – Input is a list of pflow_variable objects. Subclass of pflow. It is recommended to use dat.add(obj=pflow_variable) for easy appending. Use dat.add(index='pflow_variable.name' or dat.add(index=pflow_variable) to specify pflow object to add pflow_variable to. If no pflow object is specified, pflow_variable will be appended to the last pflow object appended to pdata. E.g., dat.add(variable, 'initial') if variable = pflow_variable and pflow.name='initial'.

Flow Variable

class pdata.**pflow_variable**(*name=''*, *type=None*, *valuelist=[]*, *unit=''*, *time_unit_type=''*, *data_unit_type=''*, *list=[]*)

Sub-class of pflow for each kind of variable (includes type and value) such as pressure, temperature, etc. There can be multiple pflow_variable objects appended to a single pflow object.

Parameters

- **name** (*str*) – Indicates name of the flow variable. Options include: ['PRESSURE', 'RATE', 'FLUX', 'TEMPERATURE', 'CONCENTRATION', 'SATURATION', 'ENTHALPY'].
- **type** (*str*) – Indicates type that is associated with name under keyword TYPE. Options for PRESSURE include: 'dirichlet', 'hydrostatic', 'zero_gradient', 'conductance', 'seepage'. Options for RATE include: 'mass_rate', 'volumetric_rate', 'scaled_volumetric_rate'. Options for FLUX include: 'dirichlet', 'neumann, mass_rate', 'hydrostatic, conductance', 'zero_gradient', 'production_well', 'seepage', 'volumetric', 'volumetric_rate', 'equilibrium'. Options for TEMPERATURE include: 'dirichlet', 'hydrostatic', 'zero_gradient'. Options for CONCENTRATION include: 'dirichlet', 'hydrostatic', 'zero_gradient'. Options for SATURATION include: 'dirichlet'. Options for ENTHALPY include: 'dirichlet', 'hydrostatic', 'zero_gradient'
- **valuelist** (*[float, float]*) – Provide one or two values associated with a single Non-list alternative, do not use with list alternative. The 2nd float is optional and is needed for multiphase simulations.
- **unit** (*str*) – Non-list alternative, do not use with list alternative. Specify unit of measurement.
- **time_unit_type** (*str*) – List alternative, do not use with non-list alternative attributes/parameters.
- **list** (*[pflow_variable_list]*) – List alternative, do not use with non-list alternative attributes/parameters. Input is a list of pflow_variable_list objects. Sub-class of pflow_variable. The add function currently does not support adding pflow_variable_list to pflow_variable objects. Appending to can be done manually. e.g., variable.list.append(var_list) if var_list=pflow_variable_list.

Flow Variable List

class pdata.**pflow_variable_list**(*time_unit_value=None*, *data_unit_value_list=[]*)

Sub-class of pflow_variable. Used for pflow_variables that are lists (as function of time) instead of a single value. Each of these list objects can hold multiple lines (from a Python

input file) with each line holding one time_unit_value and a data_unit_value_list that can hold multiple values.

Parameters

- **time_unit_value** (*float*) –
- **data_unit_value_list** (*[float]*) –

Initial Condition

class pdata.**pinitial_condition**(*flow=None*, *transport=None*, *region=None*)

Class for initial condition - a coupler between regions and initial flow and transport conditions.

Parameters

- **flow** (*str*) – Specify flow condition name
- **transport** (*str*) – Specify transport condition name
- **region** (*str*) – Specify region to apply the above specified flow and transport conditions as initial conditions.

Boundary Condition

class pdata.**pboundary_condition**(*name=''*, *flow=''*, *transport=''*, *region=''*)

Class for boundary conditions - performs coupling between a region and a flow/transport condition which are to be set as boundary conditions to that region. Multiple objects can be created.

Parameters

- **name** (*str*) – Name of boundary condition. (e.g., west, east)
- **flow** (*str*) – Defines the name of the flow condition to be linked to this boundary condition.
- **transport** (*str*) – Defines the name of the transport condition to be linked to this boundary condition
- **region** (*str*) – Defines the name of the region to which the conditions are linked

Source Sink

class pdata.**psource_sink**(*flow=None*, *region=None*)

Class for specifying source sink - this is also a condition coupler that links a region to the source sink condition.

Parameters

- **flow** (*str*) – Name of the flow condition the source/sink term is applied to.

- **region** (*str*) – Name of the region the source/sink term is applied to.

Stratigraphy Coupler

class pdata.**pstrata**(*region=None*, *material=None*)
Class for specifying stratigraphy coupler. Multiple stratigraphy couplers can be created. Couples material properties with a region.

Parameters

- **region** (*str*) – Name of the material property to be associated with a region.
- **material** (*str*) – Name of region associated with a material property.

Checkpoint

class pdata.**pcheckpoint**(*frequency=None*, *overwrite=False*)
Class for specifying checkpoint options.

Parameters

- **frequency** (*int*) – Checkpoint dump frequency.
- **overwrite** (*bool - True or False*) – Intended to be used for the PFLOTRAN keyword OVERWRITE_RESTART_FLOW_PARAMS.

Restart

class pdata.**prestart**(*file_name=''*, *time_value=None*, *time_unit=''*)
Class for restarting a simulation.

Parameters

- **file_name** (*str*) – Specify file path and name for restart.chk file.
- **time_value** (*float*) – Specify time value.
- **time_unit** (*str*) – Specify unit of measurement to use for time. Options include: 's', 'sec','m', 'min', 'h', 'hr', 'd', 'day', 'w', 'week', 'mo', 'month', 'y'.

Chemistry

class pdata.**pchemistry**(*pspecies_list=[]*, *sec_species_list=[]*, *gas_species_list=[]*, *minerals_list=[]*, *m_kinetics_list=[]*, *log_formulation=False*, *database=None*, *activity_coefficients=None*, *molal=False*, *output_list=[]*)
Class for specifying chemistry.

Parameters

- **pspecies_list** (*[str]*) – List of primary species that fully describe the chemical composition of the fluid. The set of primary species must form an independent set of species in terms of which all homogeneous aqueous equilibrium reactions can be expressed.
- **sec_species_list** (*[str]*) – List of aqueous species in equilibrium with primary species.
- **gas_species_list** (*[str]*) – List of gas species.
- **minerals_list** (*[str]*) – List of mineral names.
- **m_kinetics_list** (*[pchemistry_m_kinetic]*) – List of pchemistry_m_kinetic objects. Holds kinetics information about a specified mineral name. Works with add function so that m_kinetics_list does not need to be remembered. e.g., dat.add(mineral_kinetic).
- **log_formulation** (*bool - True or False*) –
- **database** (*str*) –
- **activity_coefficients** (*str*) – Options include: 'LAG', 'NEWTON', 'TIMESTEP', 'NEWTON_ITERATION'.
- **molal** (*bool - True or False*) –
- **output_list** (*[str]*) – To print secondary aqueous complex concentrations, either add the names of the secondary species of interest or the keyword 'SECONDARY_SPECIES' for all secondary species to the CHEMISTRY OUTPUT card. e.g., output_list = 'SECONDARY_SPECIES' or output_list = ['CO2(aq), 'PH']. By default, if ALL or MINERALS are listed under CHEMISTRY OUTPUT, the volume fractions and rates of kinetic minerals are printed. To print out the saturation indices of minerals listed under the MINERAL keyword, add the name of the mineral to the OUTPUT specification.

Chemistry Mineral Kinetic

class pdata.**pchemistry_m_kinetic**(*name=None*, *rate_constant_list=[]*)

Sub-class of pchemistry. Mineral kinetics are assigned to m_kinetics_list in pchemistry. The add function can do this automatically. e.g., dat.add(mineral_kinetic).

Parameters

- **name** (*str*) – Mineral name.
- **rate_constant_list** (*[float, str]*) – Value, Unit of Measurement. e.g., rate_constant_list=[1.e-6, 'mol/m^2-sec']

Transport Condition

class pdata.**ptransport**(*name=''*, *type=''*, *constraint_list_value=[]*, *constraint_list_type=[]*)

Class for specifying a transport condition. Multiple transport objects can be created. Specifies a transport condition based on various user defined constraints with minerals, gases, pH, charge balance, free ion, and total concentrations.

Parameters

- **name** (*str*) – Transport condition name.
- **type** (*str*) – Options include: 'dirichlet', 'dirichlet_zero_gradient', 'equilibrium', 'neumann', 'mole', 'mole_rate', 'zero_gradient'.
- **constraint_list_value** (*[float]*) – List of constraint values. The position of each value in the list correlates with the position of each type in constraint_list_type.
- **constraint_list_type** (*[str]*) – List of constraint types. The position of each value in the list correlates with the position of each value in constraint_list_value. E.g., 'initial_constraint', 'inlet_constraint'.

Constraint Condition

class pdata.**pconstraint**(*name=''*, *concentration_list=[]*, *mineral_list=[]*)

Class for specifying a transport constraint. Multiple constraint objects can be created.

Parameters

- **name** (*str*) – Constraint name.
- **concentration_list** (*[pconstraint_concentration]. Works with add function so that concentration_list does not need to be remembered. e.g., dat.add(concentration). Used for key word CONC or CONCENTRATIONS*) – List of pconstraint_concentration objects.
- **mineral_list** (*[pconstraint_mineral]*) – List of pconstraint_mineral objects. Currently does not work with add function. Used for key word MNRL OR MINERALS.

Constraint Condition Concentration

class pdata.**pconstraint_concentration**(*pspecies=''*, *value=None*, *constraint=''*, *element=''*)

Concentration unit, Sub-class for constraint. There can be multiple pconstraint_concentration objects appended to a single pconstraint object. Works with add function so that concentration_list in pconstraint does not need to be remembered. e.g., dat.add(concentration) instead of dat.constraint.concentration_list.append(concentration).

Parameters

- **pspecies** (*str*) – Primary species name for concentration.
- **value** (*float*) – Concentration value.
- **constraint** (*str*) – Constraint name for concentration. Options include: 'F', 'FREE', 'T', 'TOTAL', 'TOTAL_SORB', 'P', 'pH', 'L', 'LOG', 'M', 'MINERAL', 'MNRL', 'G', 'GAS', 'SC', 'CONSTRAINT_SUPERCRIT_CO2
- **element** (*str*) – Name of mineral or gas.

Constraint Condition Mineral

class pdata.**pconstraint_mineral**(*name=''*, *volume_fraction=None*, *surface_area=None*)

Class for mineral in a constraint with vol. fraction and surface area. There can be multiple pconstraint_concentration objects appended to a single pconstraint object. Currently does not work with add function. pconstraint_mineral can be manually appended to minerals_list in a pconstraint object. e.g., 'constraint.mineral_list.append(mineral)'.

Parameters

- **name** (*str*) – Mineral name.
- **volume_fraction** (*float*) – Volume fraction. [–]
- **surface_area** (*float*) – Surface area. [m^-1]

Regression

class pdata.**pregression**(*cells=[]*, *cells_per_process=2*)

Class for specifying regression details.

Parameters

- **cells** (*list of int*) – Specify cells for regression.
- **cells_per_process** (*int*) – Specify the number cells per process.

Dataset

class pdata.**pdataset**(*dataset_name=''*, *dataset_mapped_name=''*, *name=''*, *file_name=''*, *hdf5_dataset_name=''*, *map_hdf5_dataset_name=''*)

Class for incorporating data within a model.

Parameters

- **dataset_name** (*str*) – Opens the card block with the name of the data set in the string. I name is not given the NAME entry is required.
- **dataset_mapped_name** – Adds the MAPPED flag to the DATASET and allows for the dataset to be named.
- **name** (*str*) – Name of the data set if not included with DATASET card. Note: this string overwrites the name specified with DATASET

- **file_name** (*str*) – Name of the file containing the data
- **hdf5_dataset_name** (*str*) – Name of the group within the hdf5 file where the data resides
- **hdf5_dataset_name** – Name of the group within the hdf5 file where the data resides

2.1.2 Methods

`pdata.`**`read`**(*filename=''*)

Read a given PFLOTRAN input file. This method is useful for reading an existing a PFLOTRAN input deck and all the corresponding PyFLOTRAN objects and data structures are autmatically created.

Parameters **filename** (*str*) – Name of input file.

`pdata.`**`write`**(*filename=''*)

Write pdata object to PFLOTRAN input file. Does not execute the input file - only writes a corresponding PFLOTRAN input file.

Parameters **filename** (*str*) – Name of PFLOTRAN input file.

`pdata.`**`run`**(*input=''*, *num_procs=1*, *exe='/usr/bin/pflotran'*)

Run a pflotran simulation for a given input file with specified number of processors.

Parameters

- **input** (*str*) – Name of input file.
- **exe** (*str*) – Path to PFLOTRAN executable.
- **num_procs** (*int*) – Number of processors

`pdata.`**`add`**(*obj*, *index=''*, *overwrite=False*)

Attach an object associated w/ a list (e.g., pregion) that belongs to a pdata object.

Parameters

- **obj** (*object(e.g., pregion)*) – Object to be added to the data file.
- **index** (*String*) – (Optional) Used to find an object that is using a string as an index in a dictionary. Intended for the super class object. (E.g. Index represents flow.name if instance is pflow_variable.) Default if not specified is to use the last super-class object added to pdata.
- **overwrite** (*bool*) – Flag to overwrite an object if it already exists in a pdata object.

`pdata.`**`delete`**(*obj*, *super_obj=None*)

Delete an object that is assigned to a list of objects belong to a pdata object, e.g., pregion.

Parameters **obj** (*Object (e.g., pregion), list*) – Object to be deleted from the data file. Can be a list of objects.

To plot time histories of variables at various observation points, the following function can be used:

pdata.**plot_observation**(*variable_list=[]*, *observation_list=[]*, *observation_filenames=[]*, *plot_filename=''*, *legend_list=[]*, *fontsize=10*, *xlabel=''*, *ylabel=''*, *xtype='linear'*, *ytype='linear'*, *xrange=()*, *yrange=()*, *xfactor=1.0*, *yfactor=1.0*)

Plot time-series data from observation files at a given set of observation points.

Parameters

- **variable_list** (*[str]*) – List of the variables to be plotted
- **observation_list** (*[str]*) – List of observation names to be plotted
- **observation_filenames** (*[str]*) – List of observation filenames that are to be used for extracting data
- **plot_filename** (*str*) – Name of the file to which the plot is saved
- **legend_list** (*[str]*) – List of legend
- **fontsize** (*float*) – size of the legend font
- **xlabel** (*str*) – label on the x-axis
- **ylabel** (*str*) – label on the y-axis
- **xtype** (*str*) – type of plot in the x-direction, e.g., 'log', 'linear', 'symlog'
- **ytype** (*str*) – type of plot in the y-direction, e.g., 'log', 'linear', 'symlog'
- **xrange** (*(float,float)*) – limits on the x-axis range, e.g., (0,100)
- **yrange** (*(float,float)*) – limits on the y-axis range, e.g., (0,100)

To compare spatial profiles (1D) of the multiple realizations at one instant of time or for the same realization at different instants of time, one can use the following function:

pdata.**plot_data_from_tec**(*direction='X'*, *variable_list=[]*, *tec_filenames=[]*, *legend_list=[]*, *plot_filename=''*, *fontsize=10*, *xlabel=''*, *ylabel_list=[]*, *xtype='linear'*, *ytype='linear'*, *xrange=()*, *yrange=()*, *xfactor=1.0*, *yfactor=1.0*)

CHAPTER 3

Unsupported PFLOTRAN keywords

As of Feb 18, 2015

PFLOTRAN keywords not currently supported by PyFLOTRAN:

- BRINE
- COMPUTE_STATISTICS
- DEBUG
- GEOMECHANICS
- GEOMECHANICS_GRID
- GEOMECHANICS_OUTPUT
- GEOMECHANICS_MATERIAL_PROPERTY
- GEOMECHANICS_REGION
- GEOMECHANICS_CONDITION
- GEOMECHANICS_BOUNDARY_CONDITION
- GEOMECHANICS_STRATA
- MULTIPLE_CONTINUUM
- NONUNIFORM_VELOCITY
- NUMERICAL_JACOBIAN_FLOW
- NUMERICAL_JACOBIAN_RXN
- NUMERICAL_JACOBIAN_MULTI_COUPLE
- ORIGIN (ORIG)
- OVERWRITE_RESTART_TRANSPORT
- PROC

- USE_TOUCH_OPTIONS
- VELOCITY_DATASET
- WALLCLOCK_STOP

Keywords that are supported, but not 100%, listing attributes for keywords that are not supported:

- **CHEMISTRY**
  - REDOX_SPECIES
  - COLLOIDS
  - **MINERAL_KINETICS**
    - Only RATE_CONSTANT works
- **GRID**
  - INVERT_Z
- **TIME**
  - STEADY_STATE
- **TIMESTEPPER**
  - DT_FACTOR
  - PRESSURE_DAMPENING_FACTOR
- **MATERIAL_PROPERTY**
  - LONGITUDINAL_DISPERSIVITY
  - TRANSVERSE_DISPERSIVITY
  - PORE_COMPRESSIBILITY
  - THERMAL_EXPANSIVITY
  - PERMEABILITY - only works as a list
  - PERMEABILITY_POWER
  - PERMEABILITY_CRIT_POR
  - PERMEABILITY_MIN_SCALE_FAC
  - TORTUOSITY_POWER
  - MINERAL_SURFACE_AREA_POWER
  - SECONDARY_CONTINUUM
- **LINEAR_SOLVER**
  - Only SOLVER_TYPE works - Does not support data validation checking
- **NEWTON_SOLVER**
  - INEXACT_NEWTON

  - NO_PRINT_CONVERGENCE
  - NO_INF_NORM (NO_INFINITY_NORM)
  - NO_FORCE_ITERATION
  - PRINT_DETAILED_CONVERGENCE
  - ITOL_UPDATE (INF_TOL_UPDATE)
  - ITOL_SEC (ITOL_RES_SEC, INF_TOL_SEC)
  - MAX_NORM
- **OUTPUT**
  - NO_PRINT_INITIAL
  - NO_PRINT_FINAL
  - VOLUME
  - FLUXES
- **FLUID**
  - PHASE
  - DIFFUSION_ACTIVATION_ENERGY
- **OBSERVATION**
  - Only REGION works
- **REGION**
  - FILE
  - LIST
  - BLOCK
- **TRANSPORT_CONDITION**
  - TIME
- **FLOW_CONDITION**
  - CYCLIC
  - INTERPOLATION
  - TIME
  - GRADIENT, GRAD
  - FLUX, VELOCITY, VEL
  - SAT, SATURATION
  - CONDUCTANCE

- Additionally, we are currently in the process of adding keywords for the GENERAL MODE that is being developed in PFLOTRAN.

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

Index