PCLAM - PCLAM Computes Lineloads and Moments

This tool is designed to load a surface mesh with node-centered stress data (pressure and skin friction possibly separate) and approximate the lineloads and line moments by organizing the data into slices along the axis of interest and integrating the data found within each slice.

Developers: Michael W. Lee (michael.w.lee@nasa.gov) and T.J. Wignall (thomas.j.wignall@nasa.gov)

NASA Technical Memorandum: *PCLAM: a Python Module for Computing Surface Lineloads and Moments*

Software Organization

The PCLAM module is organized into a collection of functions saved within four objects:

1. `pclam.io.[function]`;
2. `pclam.util.[function]`;
3. `pclam.calc.[function]`;
4. and functions at the root level `pclam.[function]`.

The last of these contains the main operational functions of the suite. The other three objects contain, respectively, input-output, utility, and calculation functions. All functions that can be made faster with the `numba` just-in-time compiler are in `pclam.calc`. These functions perform a majority of the math and low-level data manipulation that comprises the suite. All functions that support general operation but that are not `numba`-compatible are in `pclam.util`. All functions that relate to loading or saving data in different formats at different points in the process are in `pclam.io`. These include loading TecPlot surface mesh files and saving ASCII lineloads data files.

In all, the PCLAM module requires the other following standard libraries to be available within the Python namespace: `datetime`, `json`, `numpy`, `os`, `scipy`, and `sys`. The `numba` module is optional. The `tecplot` module and associated user licenses are necessary only if the user attempts to handle binary or multi-zone ASCII TecPlot files, either for data input or data output.

The software is also distributed with unit tests which should verify if the package will operate as expected. Within the 'testing' directory, two scripts can be run to verify proper operation of all software components. These should only be run once PCLAM has been installed as a Python package on the user's machine. The script `test_unit.py` is run by calling `pytest test_unit.py`; it tests each PCLAM function individually but otherwise provides no output. The `plate_and_rocket_tests.py` file, again run with a `pytest` call, creates sample geometry files and generates and saves lineloads which can be visualized by the user. Any errors which occur when running these tests should be shared in detail by the `pytest` output itself.

Input Information

The `io` file also contains a `Config` class that tracks user settings and run options. For ease of access, the class is also loaded at the module level: `pclam.Config`. If the user does not specify any of the inputs, this class assigns default values. User options can be specified via an input file in the `.json` format. A brief description of all user inputs is provided below.

- `nll_points` defines the number of lineload points (default 100);
- `use_bin_mapping_file` determines whether the bin mapping matrix and bin areas will be saved and/or loaded for faster re-calculation on an existing surface grid (default False);

- `axis` defines the xyz-ordered axis along which the lineload will be defined (default x);
- `profile_axis` defines the xyz-ordered axis on which a reference profile is generated from the surface, for plotting reference (default z);
- `bin_edges_on_minmax` determines which bin method is to be used: bin edges on the body's extrema or bin centers on the body's extrema (default True, i.e., edges on extrema);
- `output_dir` sets the directory to which the lineloads will be output (default is the operating directory);
- `mapping_file_dir` sets the directory to/from which the bin mapping matrix and bin areas will be saved/loaded (default is the operating directory);
- `base_name` is the base file name to which other run details will be appended (default "placeholder");
- `mapping_file_name` is the file name header for the weighting matrix and bin areas saving/loading procedure (default "placeholder");
- `mrp` denotes the moment reference point in xyz as three decimals (default '[0,0,0]');
- `Lref` denotes the reference length with which the lineloads are nondimensionalized (default 1);
- `Sref` likewise denotes the reference area (default 1);
- `print_fandm` informs the code whether the user wants the integrated forces and moments to be saved as well as the lineloads (default True);
- `output_type` denotes the format for the lineload files (default ".dat", which is ASCII and TecPlot-compatible);
- `variables_saved` lists a subset of `[all, cp, cfx, cfy, cfz]` and denotes which lineload components are to be saved, with `cf_` denoting the shear components (default "all");
- `variables_to_load` labels the variables, in order, which are to be loaded (should be the less than or equal to the number of variables in the surface stress file) with labels consistent with what labels exist in the input file if it itself is labeled, as is the case with TecPlot (default `[x, y, z, cp]`);
- `absolute_pressure_integration` denotes whether the pressure coefficient data should be integrated in the tared form (standard) or with a correction for the absolute pressure, as is necessary with unclosed surface geometries (default False);
- `pinf_by_qref` denotes the reference absolute pressure with which the loaded pressure coefficient data must be corrected if specified by the user (default 0).

Running the Module On Its Own

In the standard application, a user can compute the lineloads on a given set of surface data by calling `pclam.run_pclam` and passing the necessary arguments. A sample implementation is included with the software release in the form of an additional Python script: `run_pclam_sample.py`. This script takes arguments including a job name, an input file (a sample of which can be auto-generated), and a surface data file name. If the data file is in the `.npy` format, it is assumed that the surface data will be organized in a node-centered format and that an associated element connectivity file will share the same filename with `_data.npy` replaced with `_conn.npy`.