flows$_{analysis}$

Release 0.7.0

Apr 02, 2022

Contents:

1 Installation **3**

2 Examples **5**

3 API Documentation **7**

4 Analysis Workflow Design **9**

5 Modules **11**

5.1 Data Loading and Generation . . . 11

5.2 Calculation and Analysis . . . 12

5.3 Visualization and Rendering . . . 13

6 Indices and tables **15**

Index **17**

flowws-analysis is an in-development set of flowws modules to create reusable analysis pipelines for scientific simulations. Although it is currently mostly useful for analyzing structures found in molecular simulation (together with flowws-freud), the framework can be used as a base for analysis and visualization in jupyter notebooks or a standalone GUI for other application domains.

CHAPTER 1

Installation

Install `flowws-analysis` from PyPI (note that most modules require dependencies; use the second `pip install` command below to install those) :

```
# this installs flowws-analysis without any prerequisites
pip install flowws-analysis

# optional prerequisites can be installed via extras, for example:
pip install flowws-analysis[garnett,gtar,notebook,plato,pyriodic,qt]
```

Alternatively, install from source:

```
pip install git+https://github.com/klarh/flowws-analysis.git#egg=flowws-analysis
```

CHAPTER 2

Examples

Consult the flowws-examples project for examples using `flowws-analysis` modules.

CHAPTER 3

API Documentation

Browse more detailed documentation online or build the sphinx documentation from source:

```
git clone https://github.com/klarh/flowws-analysis
cd flowws-analysis/doc
pip install -r requirements.txt
make html
```

CHAPTER 4

Analysis Workflow Design

In general, consider the flow of data when considering the order of the workflow steps you would like to perform: place modules that generate or load data first (such as *Garnett* and *Pyriodic*), followed by modules that modify or compute quantities (such as *Center*). Finally, use visualization or rendering modules, such as *ViewNotebook* or *Save*. Modules are presented below in approximately this order.

CHAPTER 5

Modules

5.1 Data Loading and Generation

class flowws_analysis.**Garnett**(**args*, ***kwargs*)

Emit the contents of a garnett-readable trajectory.

The Garnett module outputs frames from a trajectory to be used for analysis and visualization.

Parameters

- **filename** – Filename to open
- **frame** – Frame to load
- **loop_frames** – If True, loop the workflow over frames found in the trajectory file, beginning at the given frame

class flowws_analysis.**GTAR**(**args*, ***kwargs*)

Emit the contents of a libgetar-format file into the scope.

The GTAR module outputs the records found in a getar-format file directly into the scope. It provides a notion of frames in a trajectory using the discretely-varying record with the most indices as the basis.

Parameters

- **filename** – Getar-format filename to open
- **frame** – Frame to load
- **loop_frames** – If True, loop the workflow over frames found in the trajectory file, beginning at the given frame
- **group** – GTAR group to restrict results to

class flowws_analysis.**Pyriodic**(***kwargs*)

Browse structures available in pyriodic.

This module provides all the structures available in the pyriodic default database (which uses all available pyriodic libraries installed on the system). Systems are resized to a minimum of the given size and noise may be added before the rest of the workflow is run.

Parameters

- **structure** – Structure to display
- **size** – Minimum size of the system
- **noise** – Gaussian noise to apply to each position

5.2 Calculation and Analysis

class flowws_analysis.**Center**(***kwargs*)

Center the system through periodic boundary conditions.

This module modifies the positions of the system to have either the center of mass of the system or a single indicated particle at (0, 0, 0).

Parameters **particle** – Particle index to center with (default: use center of mass of the system)

class flowws_analysis.**Diffraction**(***kwargs*)

Compute a 3D diffraction pattern of the system and display its slice or projection.

This stage computes a 3D histogram of the system based on the given periodic system box and particle coordinates and performs the FFT in 3D. Either a slice or full projection through the Fourier space is displayed with the current system orientation.

Note: This module should be considered experimental in terms of stability for the time being; the inputs and outputs may change drastically in the future, or the module may be removed entirely.

Parameters

- **bin_count** – Number of bins to use in the x, y, and z directions
- **projection** – If True, project the diffraction pattern all the way through fourier space
- **min_value** – Minimum value of intensity to clip to
- **max_value** – Maximum value of intensity to clip to
- **sigma** – Lengthscale of blurring the FFT

class flowws_analysis.**Colormap**(***kwargs*)

Access and use matplotlib colormaps on scalar quantities.

This module emits a *color* value, calculated using a given scalar argument and matplotlib colormap name.

Valid scalars quantities can be provided to this module by saving them in the scope and adding their name to the *color_scalars* list.

Parameters

- **colormap_name** – Name of the matplotlib colormap to use
- **argument** – Name of the value to map to colors
- **range** – Minimum and maximum values of the scalar to be mapped

class flowws_analysis.**Selection**(***kwargs*)

Filter the set of displayed particles manually or by specified criteria.

This module removes particles by filtering all per-particle quantities according to a series of criteria. These criteria use the "state" scope that is used to pass data between modules, such as *scope["potential_energy"] < -0.5*.

When used interactively with vispy scenes, selections can be made with the mouse, or particles on the convex hull of a droplet can be removed.

Parameters **criteria** – List of criteria to filter by. Particles satisfying all criteria will be included.

5.3 Visualization and Rendering

class flowws_analysis.**Plato**(***kwargs*)

Render shapes via plato.

This module uses the *position*, *orientation*, *type*, *color*, and *type_shapes.json* quantities found in the scope, if provided, to produce a scene of plato shapes.

The *type_shapes.json* value should provide a string of a json-encoded list containing one *shape description* object (described below) for each type. These will be converted into plato primitives in conjunction with the *type* and other arrays.

Shape description objects are JSON objects with the following keys:

- type: one of "ConvexPolyhedron", "Disk", "Mesh", "Polygon", "Sphere"
- rounding_radius (only if type is "ConvexPolyhedron" or "Polygon"): rounding radius of a rounded shape, or 0 for perfectly faceted shapes
- vertices (only if type is "ConvexPolyhedron", "Mesh", or "Polygon"): coordinates in the shape's reference frame for its vertices; 2D for polygon and 3D for other shapes
- indices (only if type is "Mesh"): Array of triangle indices associated with the given set of vertices

Parameters

- **outline** – High-quality outline for spheres and polyhedra
- **cartoon_outline** – Cartoon-like outline mode for all shapes
- **color_scale** – Factor to scale color RGB intensities by
- **draw_scale** – Scale to multiply particle size by
- **display_box** – Display the system box
- **transparency** – Enable special translucent particle rendering
- **additive_rendering** – Use additive rendering for shapes
- **fast_antialiasing** – Use Fast Approximate Antialiasing (FXAA)
- **ambient_occlusion** – Use Screen Space Ambient Occlusion (SSAO)
- **disable_rounding** – Disable spheropolyhedra and spheropolygons
- **disable_selection** – Don't allow selection of particles for this scene

class flowws_analysis.**ViewNotebook**(**args*, ***kwargs*)

Provide an interactive view of the entire workflow using jupyter widgets.

Interactive widgets will be created inside the notebook. Arguments for each stage can be adjusted while viewing the visual results.

Parameters

- **controls** – Display controls
- **plato_backend** – Plato backend to use for associated visuals
- **vispy_backend** – Vispy backend to use for plato visuals

class flowws_analysis.**ViewQt**(**args*, ***kwargs*)

Provide an interactive view of the entire workflow using Qt.

An interactive display window will be opened that displays visual results while allowing the arguments of all stages in the workflow to be modified.

Parameters **controls** – Display controls

class flowws_analysis.**Save**(***kwargs*)

Save all visuals created to individual files.

Parameters

- **matplotlib_format** – Format to save matplotlib figures in
- **matplotlib_figure_kwargs** – Additional keyword arguments to pass to matplotlib Figure creation
- **plato_format** – Format to save plato figures in
- **plato_backend** – Plato backend to use for associated visuals
- **vispy_backend** – Vispy backend to use for plato visuals
- **file_modifiers** – List of additional filename modifiers to use

class flowws_analysis.**SaveGarnett**(***kwargs*)

Save trajectory quantities using Garnett.

This stage currently only saves an individual frame, but saving an entire trajectory is intended to work in the future.

Parameters **filename** – Name of file to save trajectory to

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index

C

Center (*class in flowws_analysis*), 12
Colormap (*class in flowws_analysis*), 12

D

Diffraction (*class in flowws_analysis*), 12

G

Garnett (*class in flowws_analysis*), 11
GTAR (*class in flowws_analysis*), 11

P

Plato (*class in flowws_analysis*), 13
Pyriodic (*class in flowws_analysis*), 11

S

Save (*class in flowws_analysis*), 14
SaveGarnett (*class in flowws_analysis*), 14
Selection (*class in flowws_analysis*), 12

V

ViewNotebook (*class in flowws_analysis*), 13
ViewQt (*class in flowws_analysis*), 14