GEMDAT

Python molecular dynamics analysis

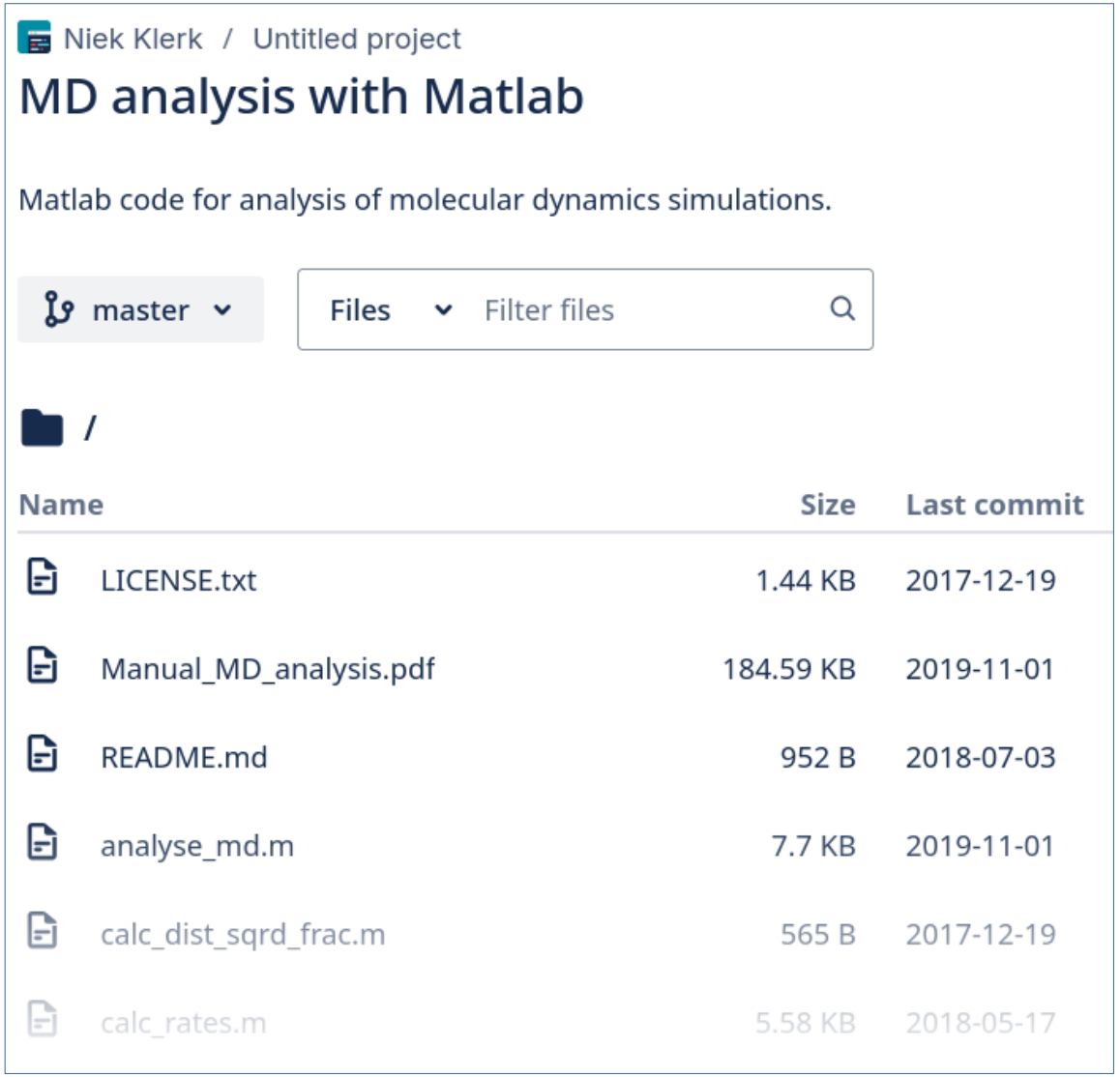
netherlands eScience center

2025-04-10

netherlands eScience center

Why GEMDAT?

- 2 year project with eScience Center (2023 – 2025): **Developing a Generalized Molecular Dynamics Analysis Tool**
- Build on and expand matlab code by Niek de Klerk
- TU Delft team:
  - Alexandros Vasileiadis
  - Theo Famprikis
  - Anastasia Lavrinko
  - Victor Landgraf



https://bitbucket.org/niekdeklerk/md-analysis-with-matlab

N.J.J. de Klerk, E. van der Maas, and M. Wagemaker,
ACS Applied Energy Materials, (2018), doi: 10.1021/acsaem.8b00457

netherlands eScience center

What is GEMDAT?

Python code for molecular dynamics analysis

- Easy-to-use Python API built on top of Pymatgen
- Open source licence (Apache 2.0)
- Works with Gromacs, LAMMPS, and VASP trajectories
- Find jumps and transitions between sites
- Calculate metrics like tracer and jump diffusivity
- Characterize and visualize diffusion pathways
- Plots for all kinds of things

GEMDAT

https://github.com/GEMDAT-repos/GEMDAT

netherlands eScience center

eScience goals

Easy to use and extensible Python API

Follow best practices (code style, testing, docs, pypi, open source)

Reasonably fast for interactive use (e.g. Jupyter Notebooks)

Crystallographically accurate

Compatible with Pymatgen (don’t reinvent the wheel)

pymatgen

https://pymatgen.org/

netherlands eScience center

Getting started

Python 3.10 or newer

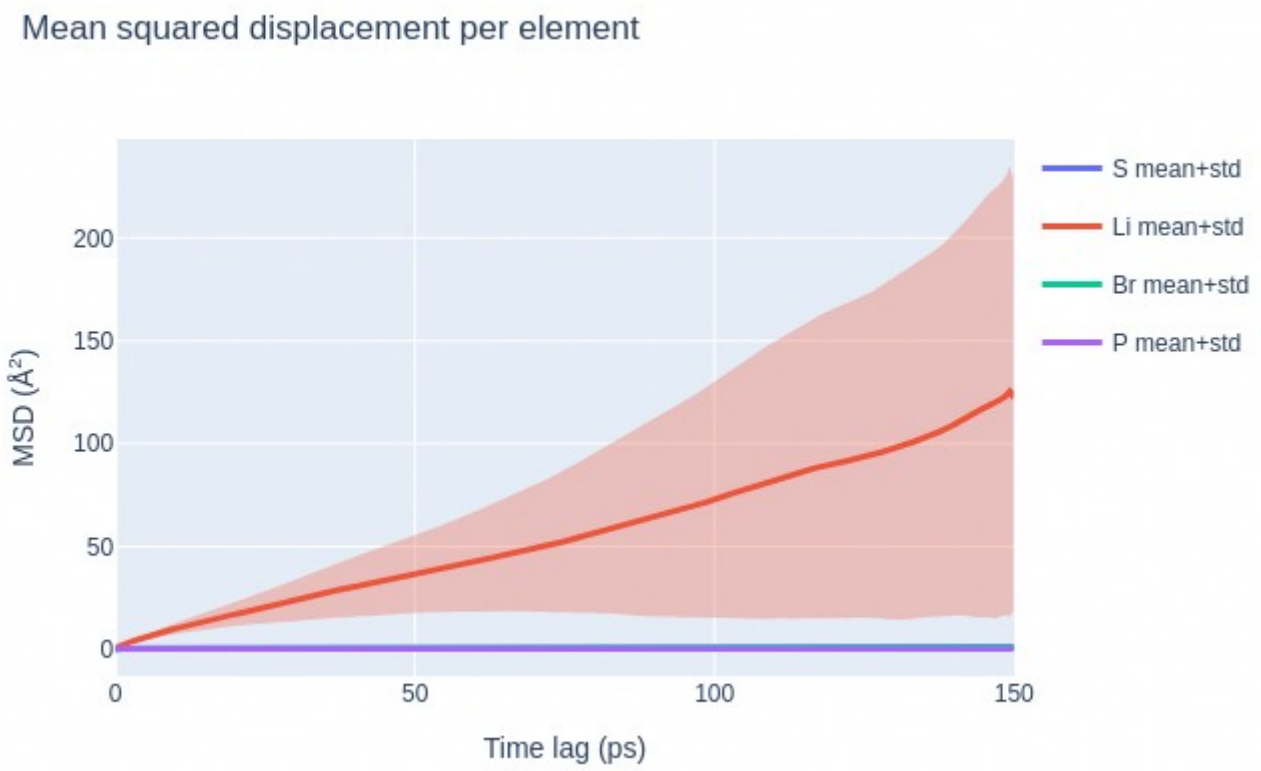
Installation:

```
pip install gemdat
```

Documentation:

https://gemdat.readthedocs.io/

```
from gemdat import Trajectory
traj = Trajectory.from_vasprun('vasprun.xml')
traj.plot_msd_per_element()
```



netherlands eScience center

Crystallographically accurate

Metric tensor

- Easy to calculate
- Fast, vectorized computation
- General (works for orthogonal and non-orthogonal lattices)

$$\mathbf{G} = \begin{bmatrix} a^2 & ab\cos(\gamma) & ac\cos(\beta) \\ ab\cos(\gamma) & b^2 & bc\cos(\alpha) \\ ac\cos(\beta) & bc\cos(\alpha) & c^2 \end{bmatrix}$$

Metric tensor (Dunitz 1978, p227)

e.g. calculate displacements:

Distance

$$|V| = (\mathbf{x}^T \mathbf{G} \mathbf{x})^{\frac{1}{2}}$$

Cartesian coords

Fractional coords

Naive implementation (don't do this):

$$|V| = (x^2a^2 + y^2b^2 + z^2c^2)^{\frac{1}{2}}$$

netherlands eScience center

Caching

- Optimize for fast loading of data
- Cache loads instantly
- Example dataset with 75k timesteps x 104 atoms = **7.5m coords**

```
In [1]: from gemdat import Trajectory

In [2]: %time t = Trajectory.from_vasprun('vasprun.xml')
CPU times: user 1min 15s, sys: 1.64 s, total: 1min 16s
Wall time: 1min 17s

In [3]: %time t = Trajectory.from_vasprun('vasprun.xml')
CPU times: user 4.96 ms, sys: 22.9 ms, total: 27.8 ms
Wall time: 27.3 ms
```

2 plotting backends

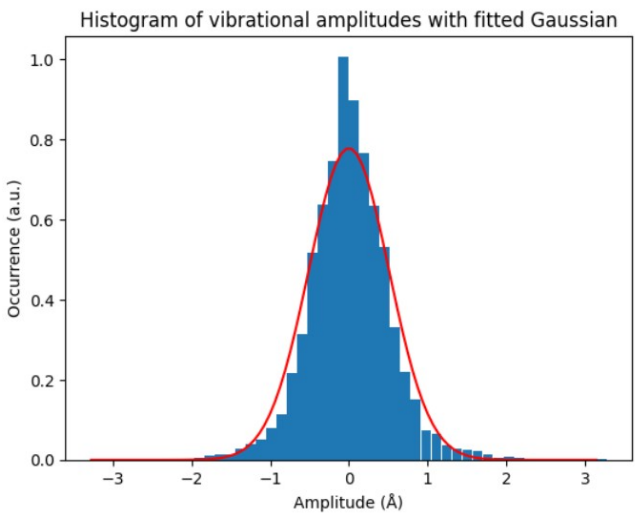
Easy to switch backends

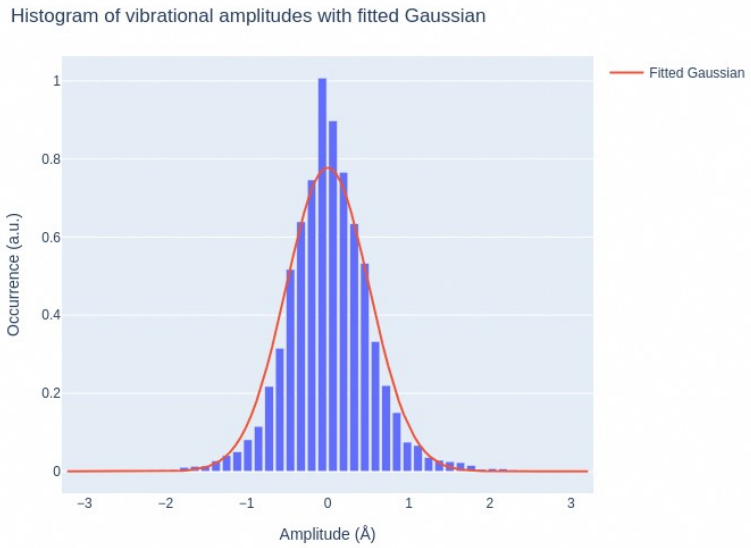
1. Matplotlib

   Reproducible, stable, 'publication-quality'

2. Plotly

   Interactive plots (i.e. Notebooks)






```
diff_traj.plot_vibrational_amplitudes(backend='plotly')
diff_traj.plot_vibrational_amplitudes(backend='matplotlib')
```

netherlands eScience center

Trajectory

MD data
- Vasp
- Gromacs
- LAMMPS
- Pymatgen

Trajectory

Metrics
- Speed
- Density
- Tracer diffusivity
- Haven ratio
- Tracer conductivity
- Attempt frequency
- Vibration amplitude

```
from gemdat import Trajectory

traj = Trajectory.from_vasprun('vasprun.xml')
metrics = traj.filter('Li').metrics()
print(metrics.haven_ratio())
# 24.032
print(metrics.tracer_diffusivity())
# 1.826e-09 m^2 s^-1
```

netherlands eScience center

Trajectory

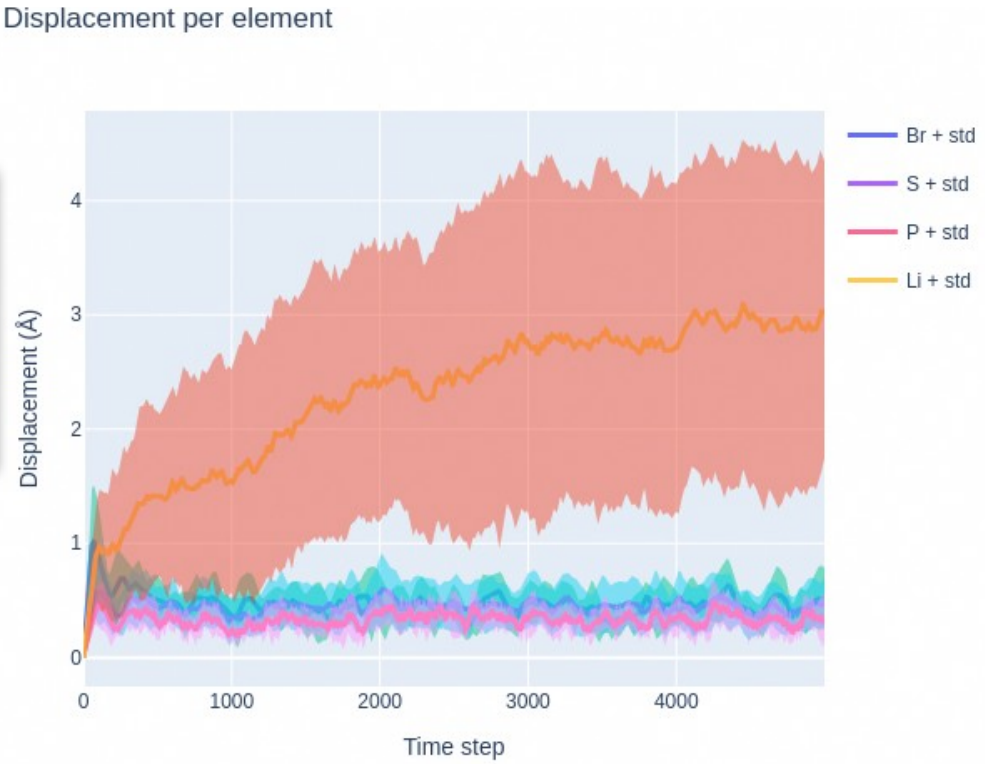
MD data
- Vasp
- Gromacs
- LAMMPS
- Pymatgen

Trajectory

Metrics

Plots
- Displacement per atom
- Displacement per element
- MSD per element
- Displacement histogram
- Frequency vs. Occurencs
- Vibrational amplitudes

```
from gemdat import Trajectory

traj = Trajectory.from_vasprun('vasprun.xml')
traj.plot_displacement_per_element()
```



netherlands eScience center

Trajectory

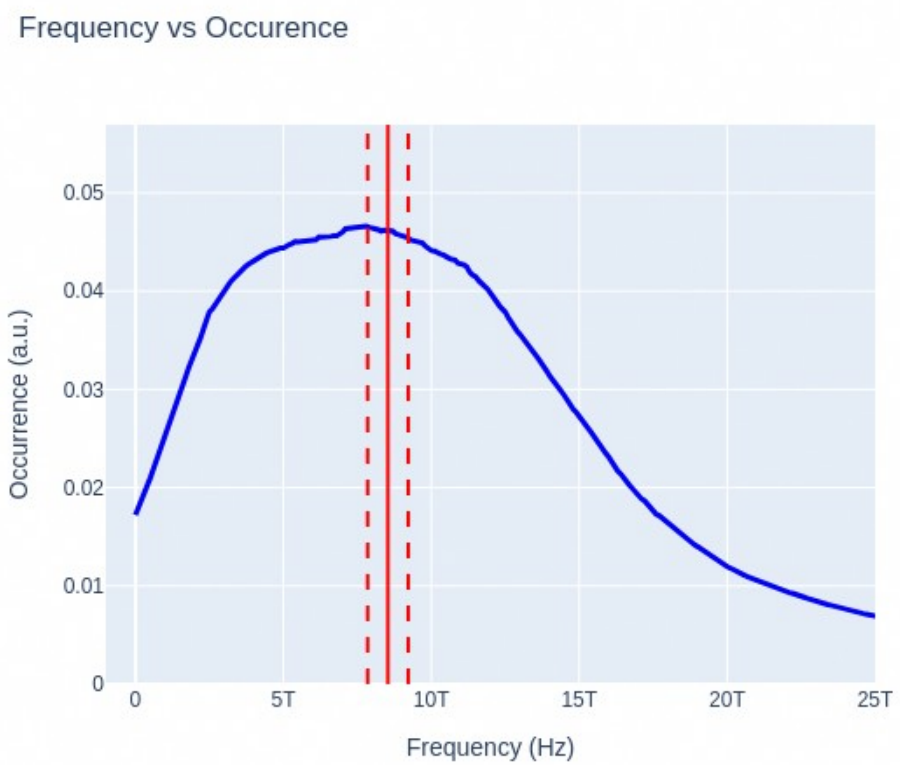
MD data
- Vasp
- Gromacs
- LAMMPS
- Pymatgen

Trajectory

Metrics

Plots
- Displacement per atom
- Displacement per element
- MSD per element
- Displacement histogram
- Frequency vs. Occurencs
- Vibrational amplitudes

```
diff_traj = traj.filter('Li')
diff_traj.plot_frequency_vs_occurence()
```



netherlands eScience center

Trajectory

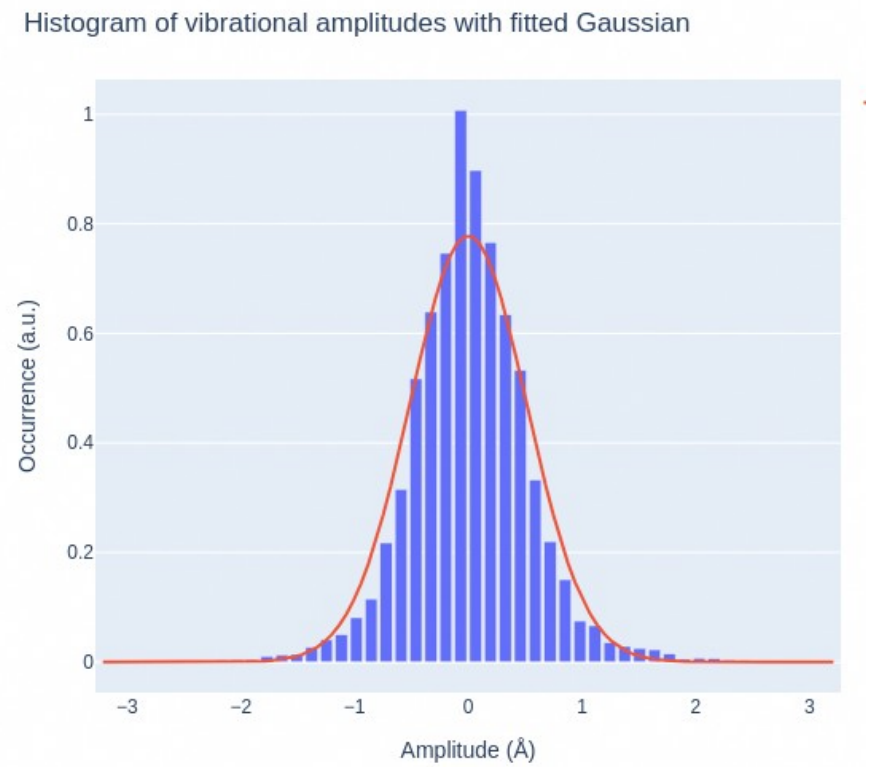
MD data
- Vasp
- Gromacs
- LAMMPS
- Pymatgen

Trajectory

Metrics

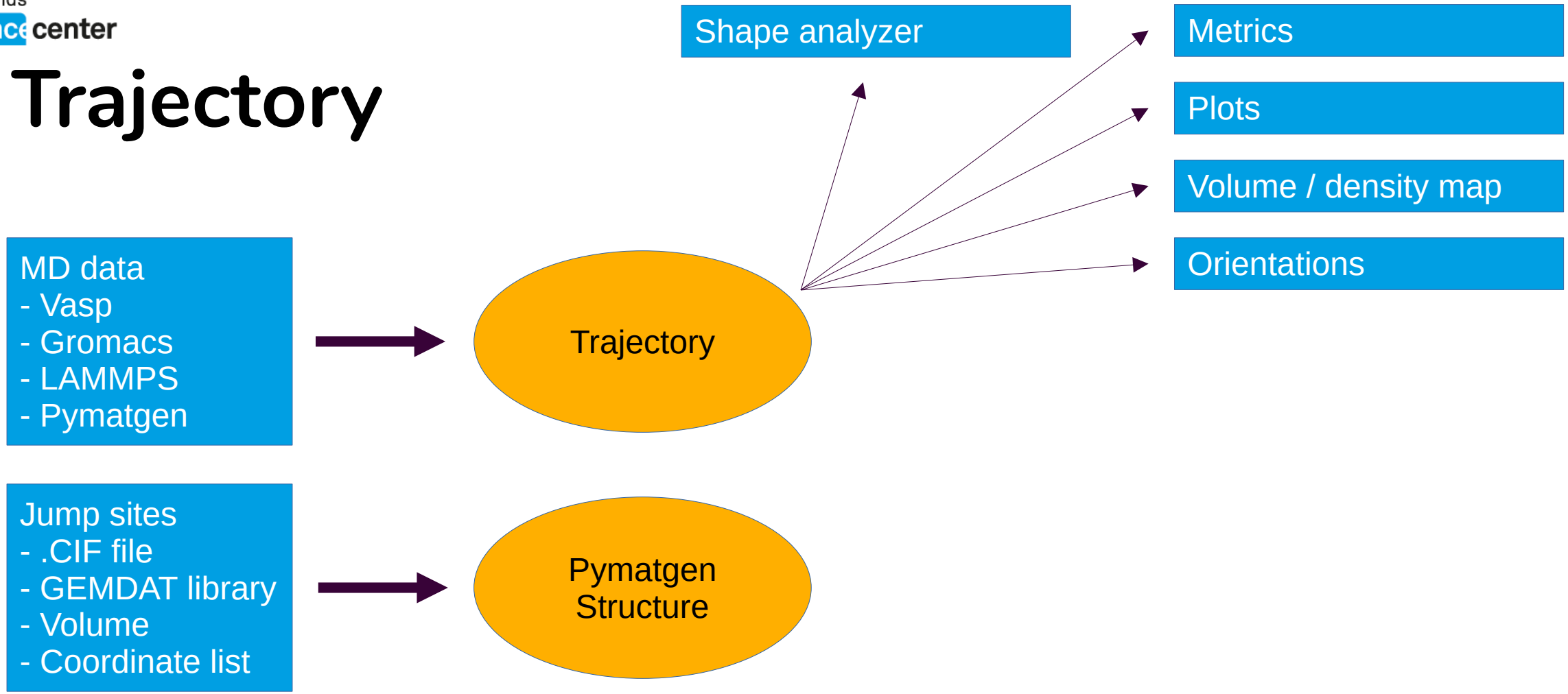
Plots
- Displacement per atom
- Displacement per element
- MSD per element
- Displacement histogram
- Frequency vs. Occurencs
- Vibrational amplitudes

```
diff_traj = traj.filter('Li')
diff_traj.plot_vibrational_ampitudes()
```



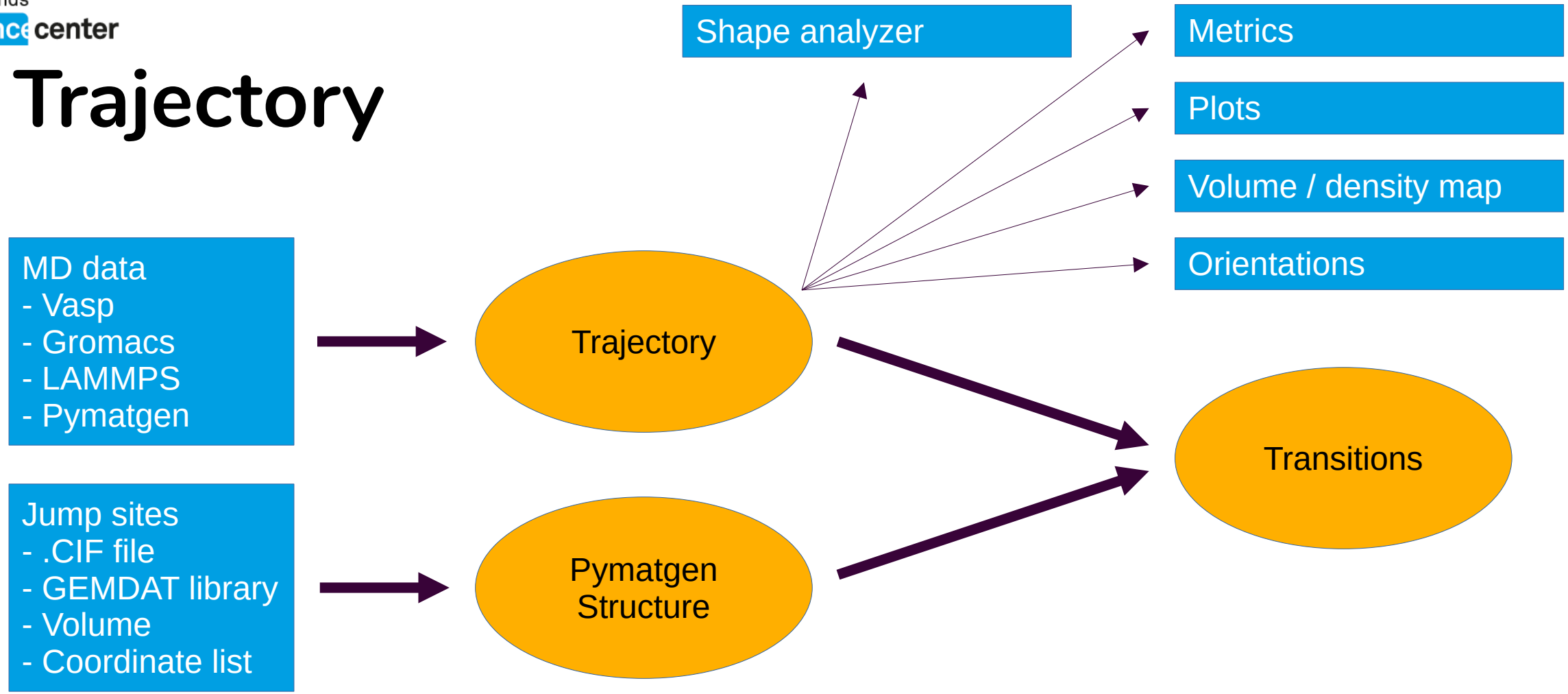
netherlands eScience center

Trajectory



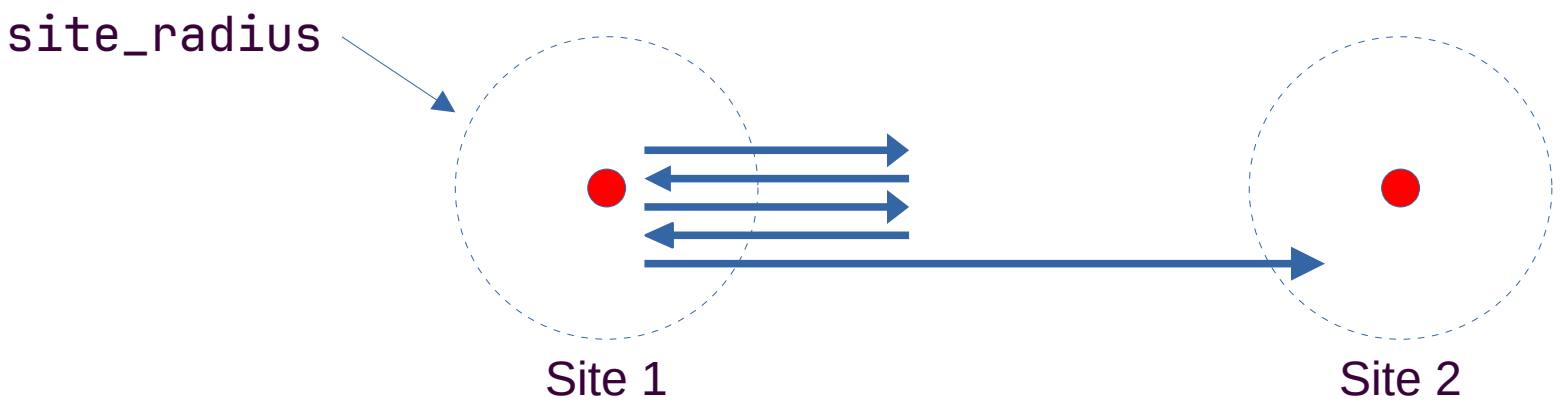
netherlands eScience center

Trajectory



Transitions

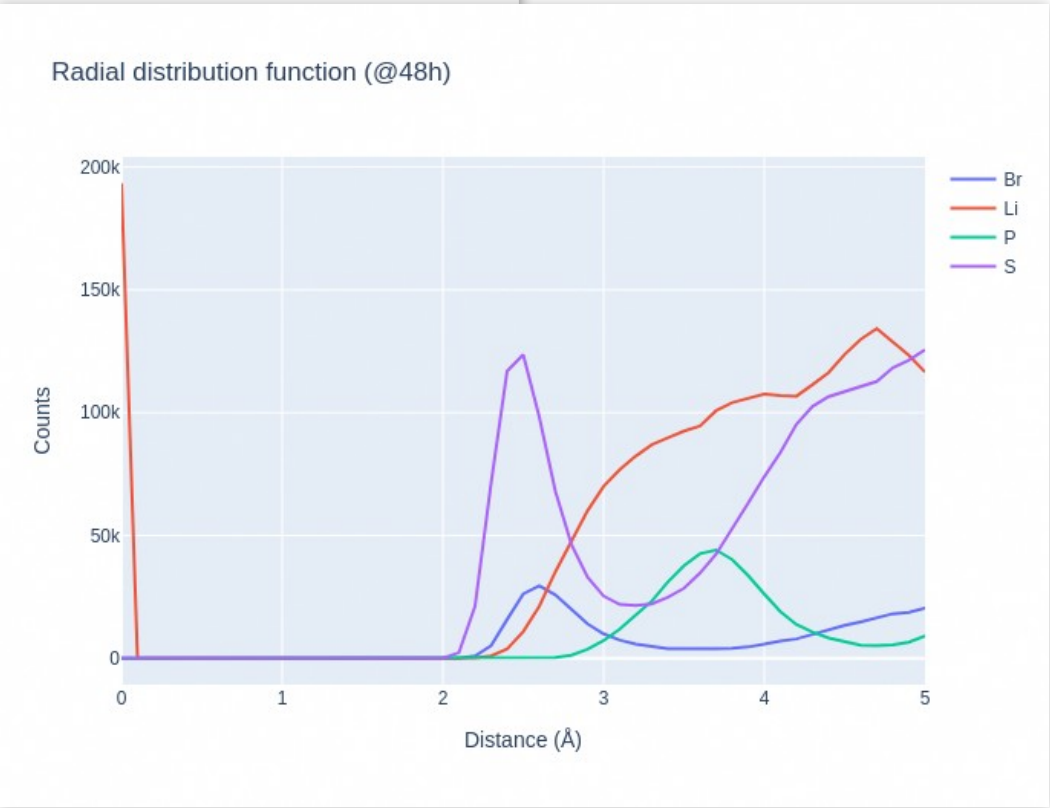
```python
from gemdat.io import load_known_material
sites = load_known_material('argyrodite', supercell=(2, 1, 1))


transitions = traj.transitions_between_sites(
    sites=sites,
    floating_specie='Li',
    site_radius=1.0,
)
transitions.events
```



netherlands eScience center

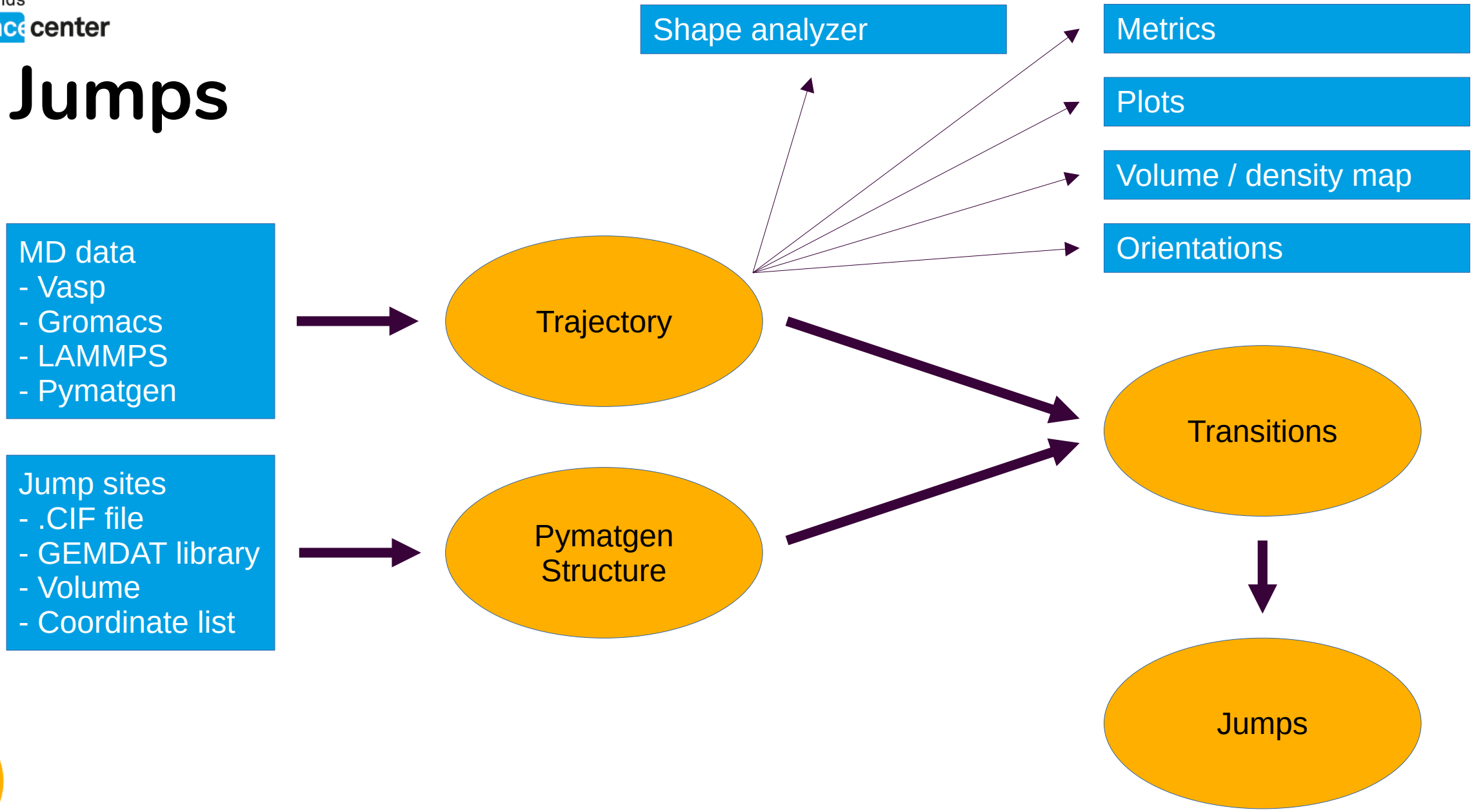
Transitions

```
occupancy = transitions.occupancy()
occupancy[0:3]
# [PeriodicSite: 48h (Li:0.45) (1.816, 1.816, 0.2382) [0.0915, 0.183, 0.024],
#  PeriodicSite: 48h (Li:0.361) (11.74, 1.816, 0.2382) [0.5915, 0.183, 0.024],
#  PeriodicSite: 48h (Li:0.393) (1.816, 6.778, 5.2) [0.0915, 0.683, 0.524]]

rdf_data = transitions.radial_distribution(
    floating_specie='Li',
)

for rdfs in rdf_data.values():
    rdfs.plot()
```



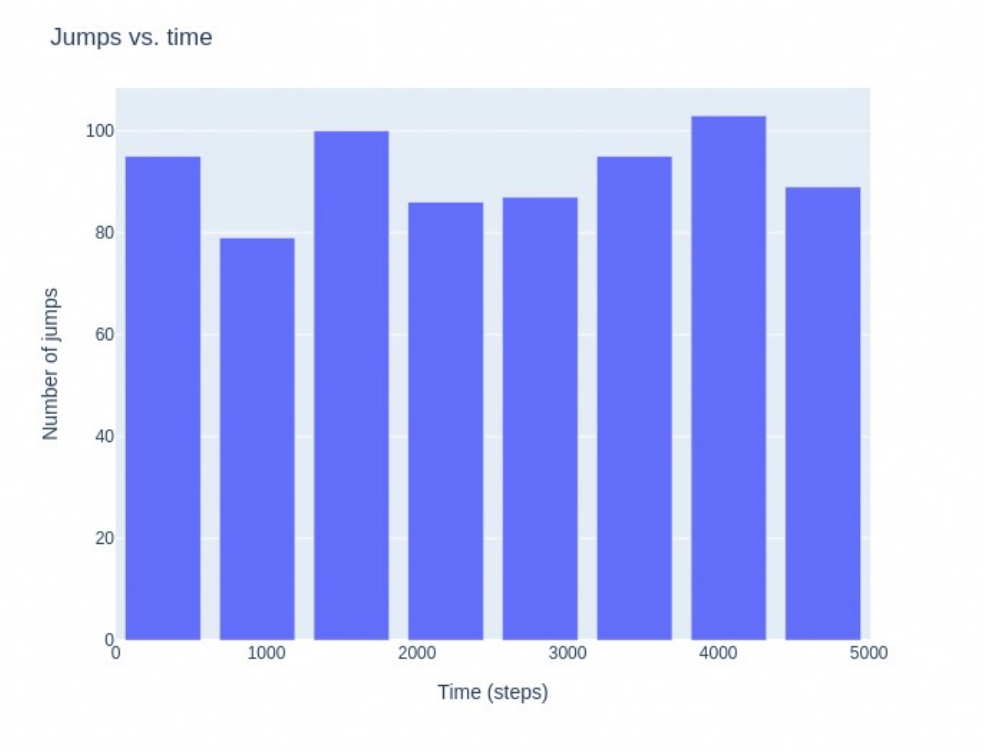
netherlands eScience center

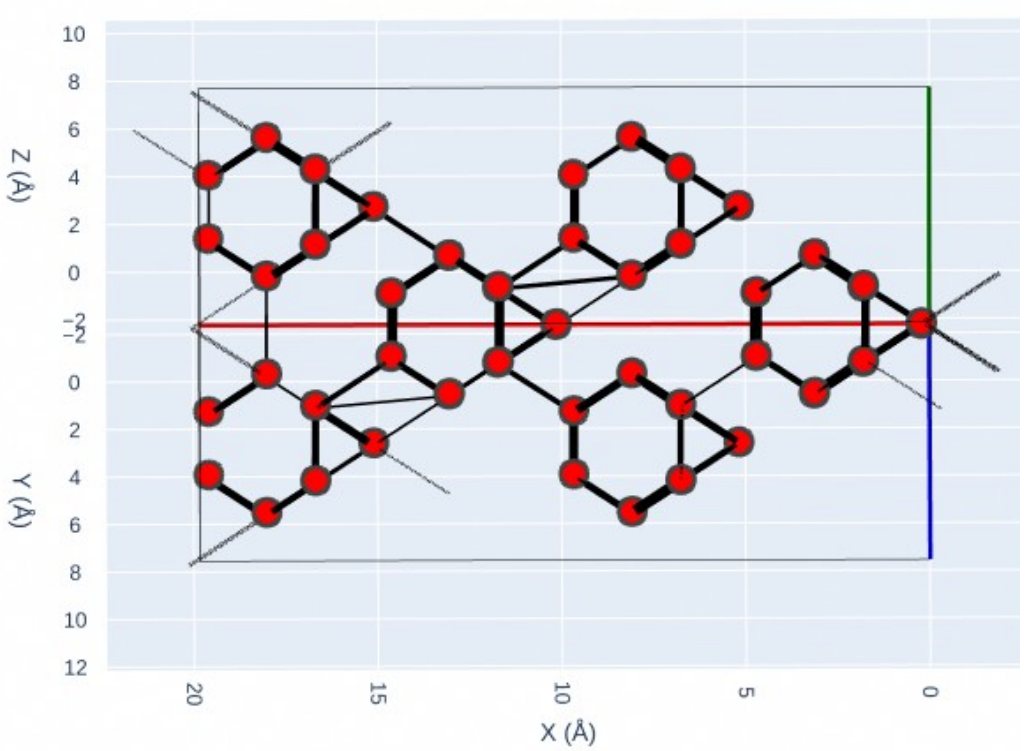
Jumps



Jumps

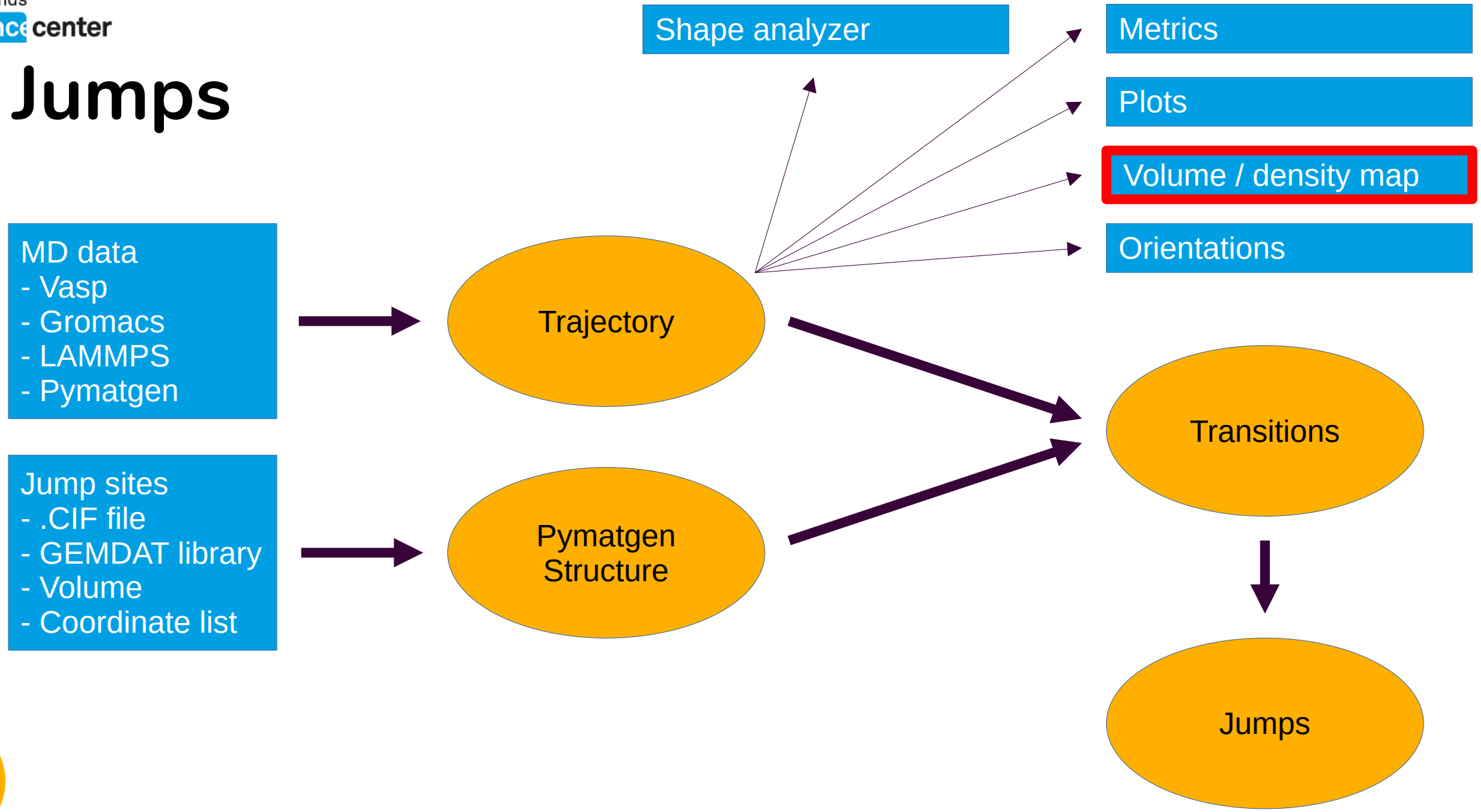
```
jumps = transitions.jumps(minimal_residence=0)

jumps.plot_jumps_vs_time()
Jumps.plot_jumps_3d()
```





netherlands eScience center

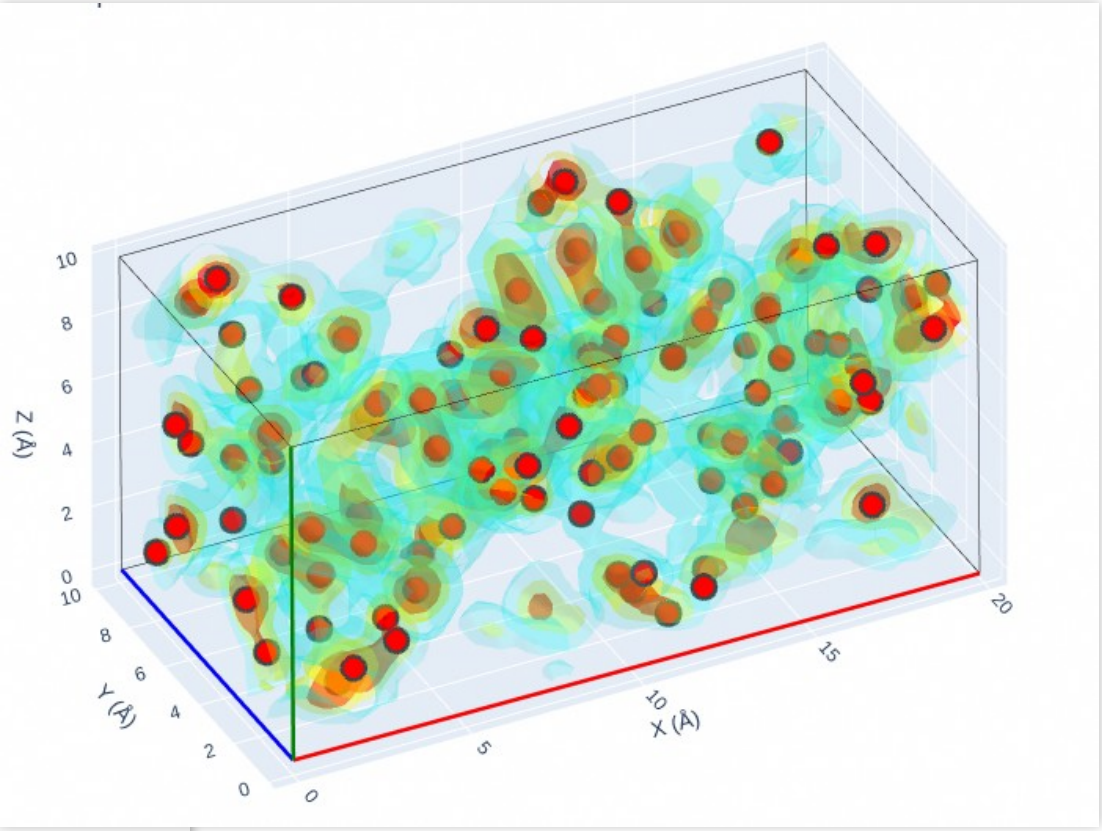
Jumps



netherlands eScience center

Volume

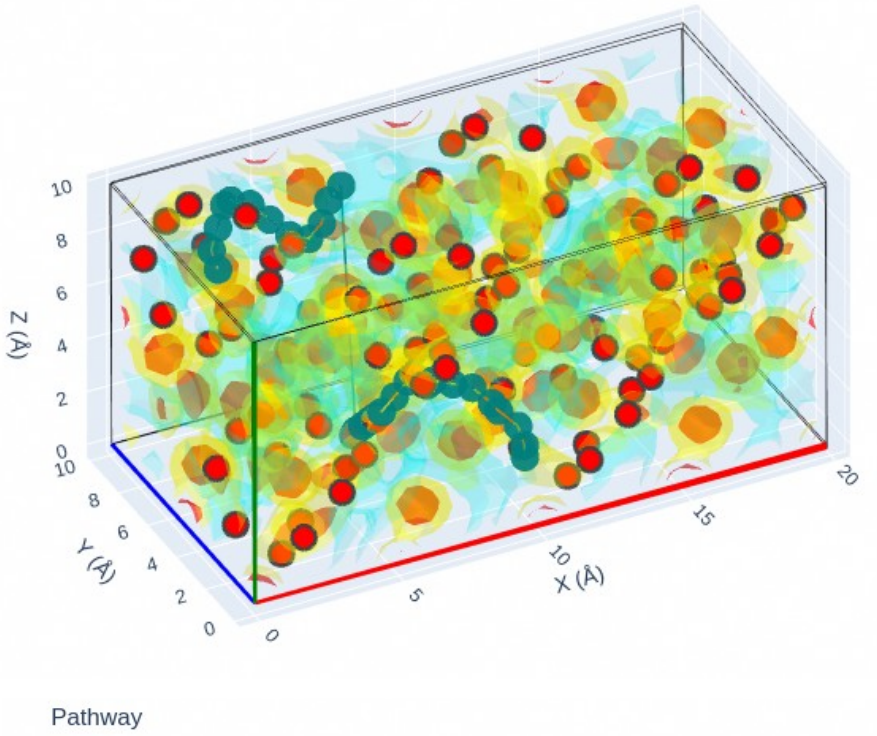
```
diff_traj = traj.filter('Li')

vol = diff_traj.to_volume()

sites = vol.to_structure(specie='Li')

vol.plot_3d(structure=sites)

# export to e.g. VESTA
vol.to_vasp_volume(sites, filename='data.vasp')
```

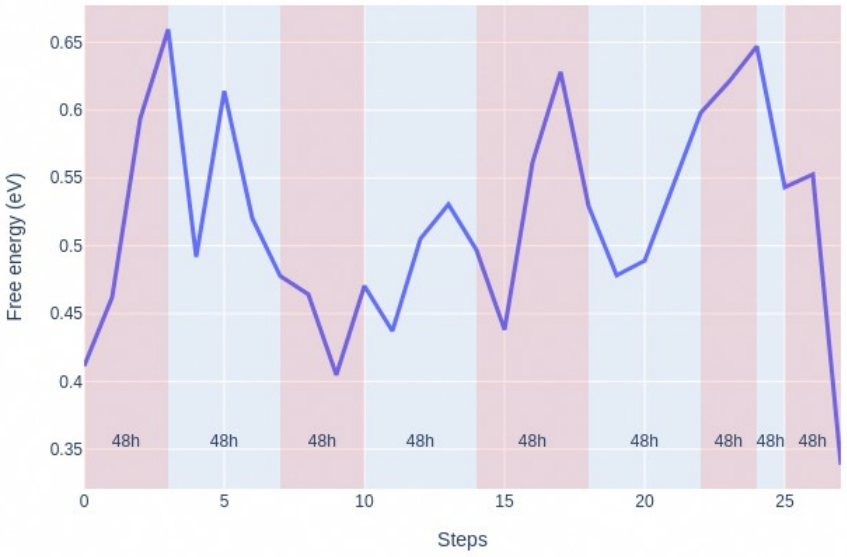


netherlands eScience center

Path optimization

```
free_energy = vol.get_free_energy(temperature=650)

path = free_energy.optimal_path(
    start=(22, 3, 3),
    stop=(5,16,16),
    method='dijkstra',
)
path.total_energy
# 14.51 eV

volume.plot_3d(structure=sites, paths=path)

path.plot_energy_along_path(structure=sites)
```

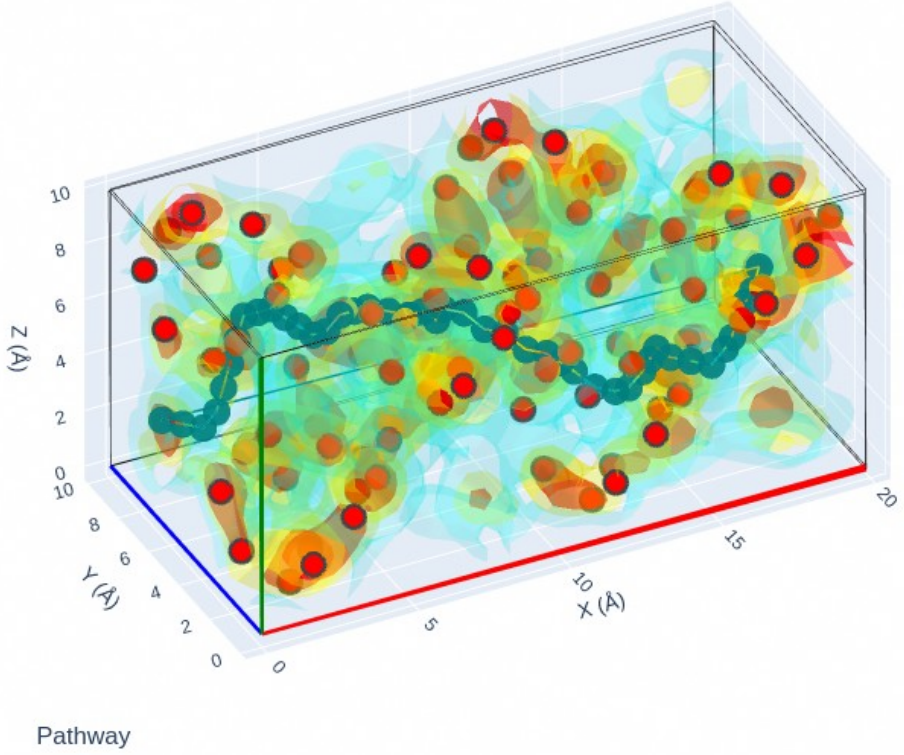


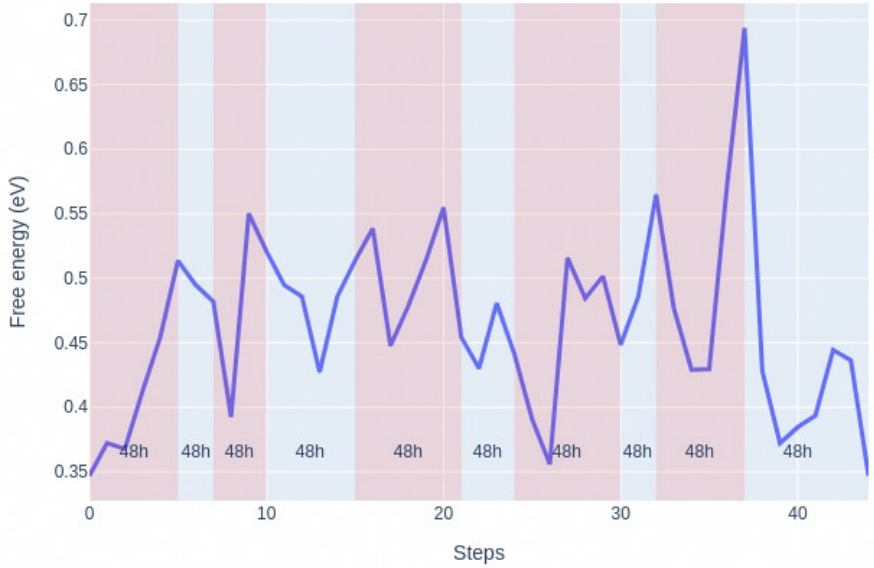


netherlands eScience center

Percolating path

```
path =  free_energy.optimal_percolating_path(
    peaks=peaks,
    percolate='x'
)

path.total_energy
# 20.809 eV

volume.plot_3d(structure=sites, paths=path)

path.plot_energy_along_path(structure=sites)
```





docs passing | Tests for GEMDAT passing | Coverage 84% | python 3.10 | 3.11 | 3.12 | pypi v1.6.0
DOI 10.5281/zenodo.8401669

GEMDAT

Python molecular dynamics analysis

GEMDAT

Gemdat is a Python library for the analysis of diffusion in solid-state electrolytes from Molecular Dynamics simulations. Gemdat is built on top of Pymatgen, making it straightforward to integrate it into your Pymatgen-based workflows.

With Gemdat, you can:

- Explore your MD simulation via an easy-to-use Python API
- Load and analyze trajectories from VASP and LAMMPS simulation data
- Find jumps and transitions between sites
- Effortlessly calculate tracer and jump diffusivity
- Characterize and visualize diffusion pathways
- Plot radial distribution functions

To install:

https://gemdat.readthedocs.io