

# Practical Course on Molecular Dynamics and Trajectory Analysis

## Episode 5: Trajectory analysis

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya  
Facultat de Ciències, Tecnologia i Enginyeries (FCTE)

*jordi.villa@uvic.cat*

MD Course and Trajectory Analysis  
Concepcion, January 2026

# Contents

## 1 Episode 5: Trajectory analysis

- Preprocessing and cleanup
- Velocities and inertia
- Reduction and clustering
- Free energies and surfaces
- Statistics and confidence
- Summary

# Notation used throughout the notes

- Phase-space point:  $z = (q, p) \in \mathbb{R}^{3N} \times \mathbb{R}^{3N}$ , with positions  $q$  and momenta  $p$ .
- Potential energy (force field):  $U(q)$ ; forces  $\mathbf{F}(q) = -\nabla_q U(q)$ .
- Hamiltonian:  $H(q, p) = K(p) + U(q)$  with  $K(p) = \sum_i \frac{\|p_i\|^2}{2m_i}$ .
- Temperature  $T$ , Boltzmann constant  $k_B$ , inverse temperature  $\beta = (k_B T)^{-1}$ .
- Time step  $\Delta t$ ; lag time for kinetic models  $\tau$ .

## Trajectory analysis: estimators, bias, uncertainty

Given an observable  $A(q_t)$  sampled from a (possibly correlated) trajectory, the naive estimator

$$\hat{\mu}_A = \frac{1}{T} \sum_{t=1}^T A(q_t)$$

has variance inflated by time correlation:

$$\text{Var}(\hat{\mu}_A) \approx \frac{\sigma_A^2}{T} 2\tau_{\text{int}}, \quad \tau_{\text{int}} = \frac{1}{2} + \sum_{k \geq 1} \rho_A(k),$$

where  $\rho_A(k)$  is the autocorrelation at lag  $k$ . This motivates block-averaging and effective sample size.

## Suggested figure: autocorrelation function / effective sample size

**Figure:** Example autocorrelation decay and its relation to integrated autocorrelation time. Source: Wikipedia article on autocorrelation (choose an image from the page or linked Commons).

# Pre-analysis pipeline

- ① Convert and clean trajectories with `simulatePdb.py` or `simulateGromacs.py`.
- ② Align structures and strip irrelevant solvent.
- ③ Define atom subsets and write DCD/NetCDF for downstream analysis.

Scripts from the OpenMM Application Layer. [2]

# Velocities and distributions

- ‘StateDataReporter’ and ‘DCDReporter’ build histograms of  $v_i$  and energies.
- By the equipartition theorem:  $\langle K \rangle = \frac{3N}{2} k_B T$  for classical systems.
- Monitoring  $C(t) = \langle A(0)A(t) \rangle$  reveals the relevant timescales.

# RMSD and alignments

$$\text{RMSD}(t) = \sqrt{\frac{1}{N} \sum_{i=1}^N \|\mathbf{r}_i(t) - \mathbf{r}_i^{\text{ref}}\|^2}.$$

- Use the Kabsch algorithm for optimal rotations.
- Document with `simulateAmber.py` and per-replica CSV output.

# Radius of gyration and inertia tensor

$$I_{ab} = \sum_i m_i (r_i^2 \delta_{ab} - r_{ia} r_{ib}) .$$

- Eigenvalues of  $I$  describe global anisotropy.
- Monitor  $R_g^2 = \frac{1}{N} \sum_i m_i \|\mathbf{r}_i - \mathbf{r}_{CM}\|^2$  per replica.

# Extremes and contacts

- Binary maps with a 0.45 nm cutoff generate contact matrices.
- Columns represent domain changes detected by `simulateCharmm.py`.

# Covariance and PCA

$$C = \langle (\mathbf{x} - \langle \mathbf{x} \rangle)(\mathbf{x} - \langle \mathbf{x} \rangle)^T \rangle.$$

- Project onto the leading eigenvectors and define event axes.
- Relate to `coarse_grained_polymer.py` to compare slow modes of polymers and proteins.

# Hierarchical clustering

- Use average linkage and silhouette index to decide  $k$ .
- Poor choices yield empty microstates; prefer progressive trials.

# Transitions and graphs

- State graphs (see ‘graph.png’) represent kinetic channels.
- Annotated edges reflect fluxes measured by OpenMMTools scripts.

# Eigenvalues and modes

$$C\mathbf{w} = \lambda\mathbf{w}, \quad \lambda \in \mathbb{R}_+.$$

- Dominant modes identify collective motions.
- Save projections to feed Markov models.

# Density functions

$$g(r) = \frac{1}{4\pi r^2 \rho} \left\langle \sum_{i \neq j} \delta(r - r_{ij}) \right\rangle.$$

- Compare with `argon-chemical-potential.py` for validation.
- Use reweighted histograms to estimate  $F(r) = -k_B T \ln g(r)$ .

# Free energy surfaces and barriers

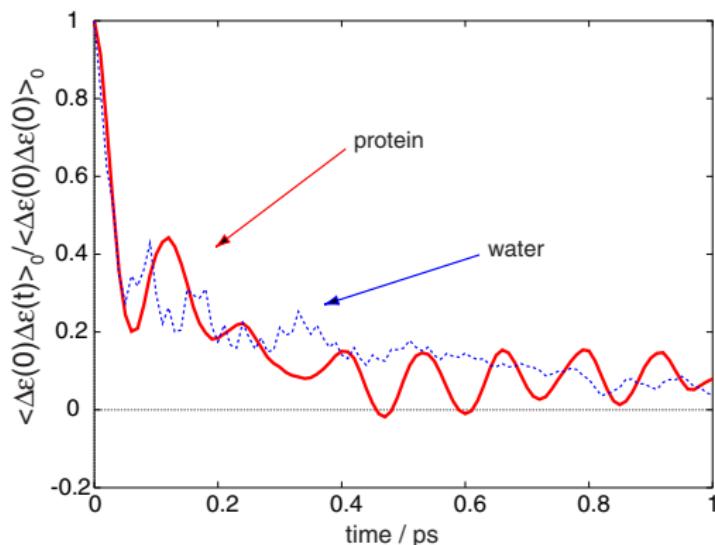
- Projection  $F(s) = -k_B T \ln P(s)$  requires robust CVs.
- Integrate with ‘umbrella sampling’ windows and combine with MBAR.
- Compare 1D profiles with `simulateAmber.py`.
- Validate using manually overlapping histograms.

# Bootstrap and blocking

- Resample blocks to estimate variances.
- Autocorrelations (see `bs_autocorrelation.pdf`) determine the block size.

# Block autocorrelations

Autocorrelation of the energy gap  
in water and protein for the LADH system



# Error series

$$\sigma_{\bar{A}} = \frac{\sigma_A}{\sqrt{N_{\text{eff}}}}, \quad N_{\text{eff}} = \frac{N}{g}, \quad g = 1 + 2 \sum_t \rho(t).$$

# Final reports

- Generate RMSD/RMSF/radius of gyration tables with `simulateCharmm.py`.
- Export CSV/HDF for easy handoff to PyEMMA/Deeptime.

## Summary and next step

- Preprocess without artifactual PBC.
- Stable CVs and converged histograms.
- Reports with errors and documented trajectories.
- Use the prepared features as input for PyEMMA/Deeptime.

# References I

- [1] OpenMM Cookbook. *Autocorrelation example from Selecting Values for Simulation Parameters*. CC BY-SA 4.0. URL: [https://openmm.github.io/openmm-cookbook/latest/notebooks/tutorials/simulation\\_parameters.html](https://openmm.github.io/openmm-cookbook/latest/notebooks/tutorials/simulation_parameters.html) (visited on 01/12/2026).
- [2] OpenMM developers. *OpenMM application guide*. Reference for scripting and workflow examples. 2024. URL: <https://docs.openmm.org/latest/userguide/application.html>.