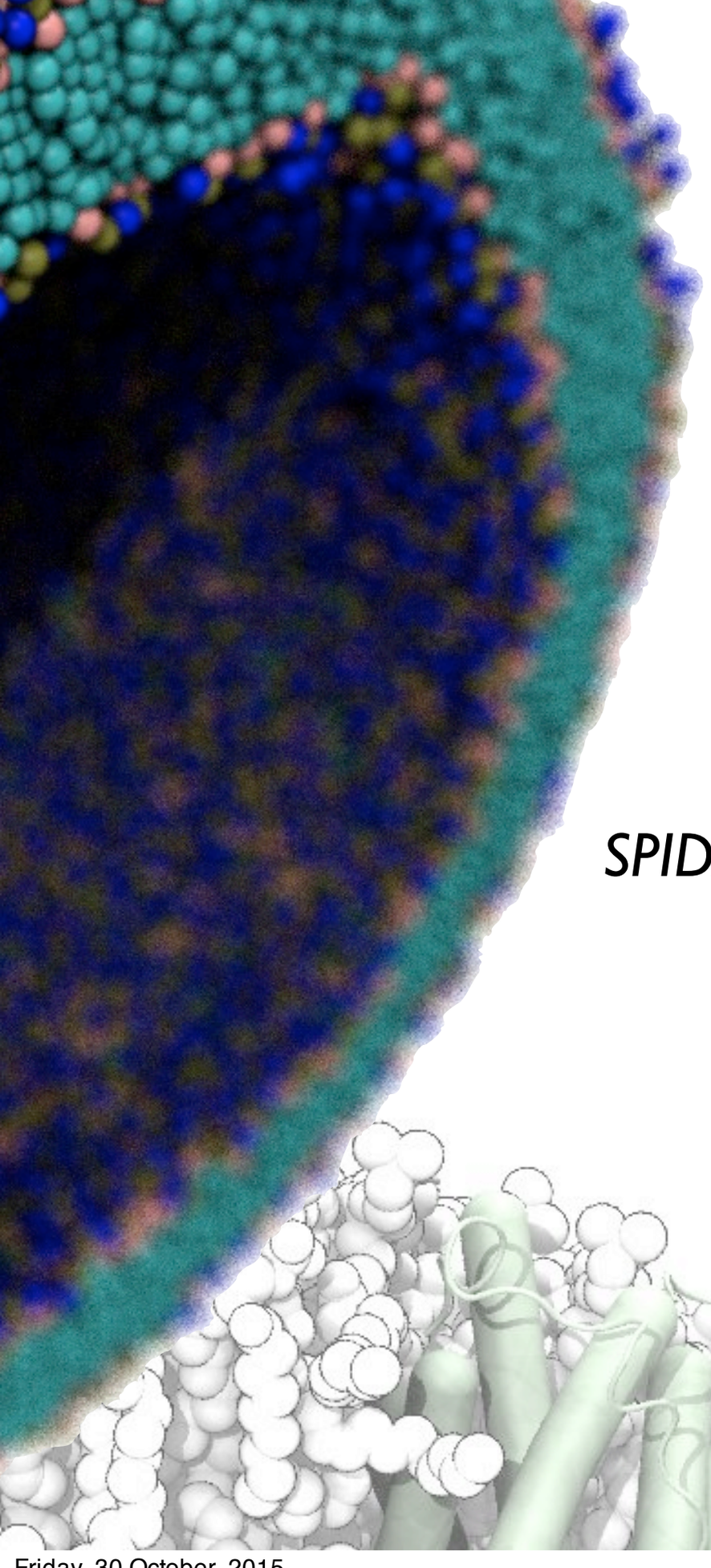


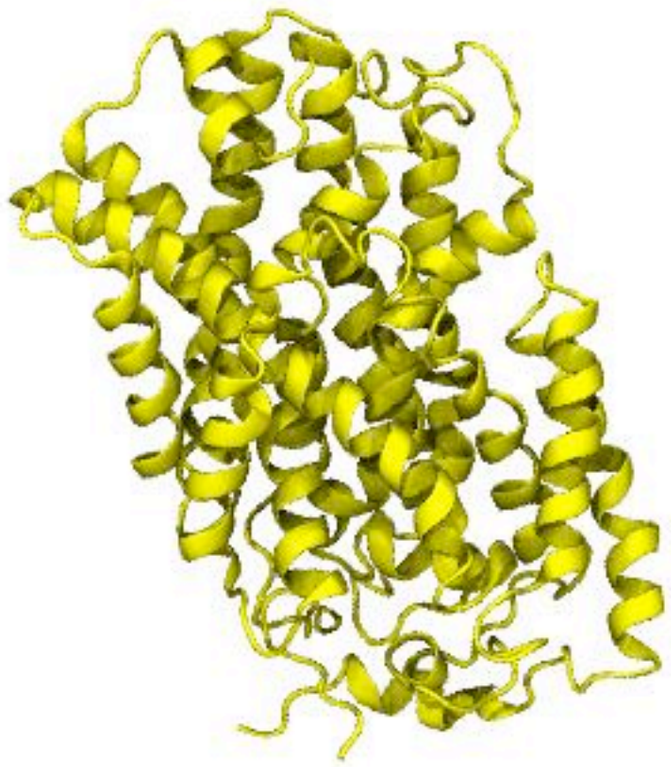
MD ANALYSIS +

SPIDAL Teleconference, 2015-10-30

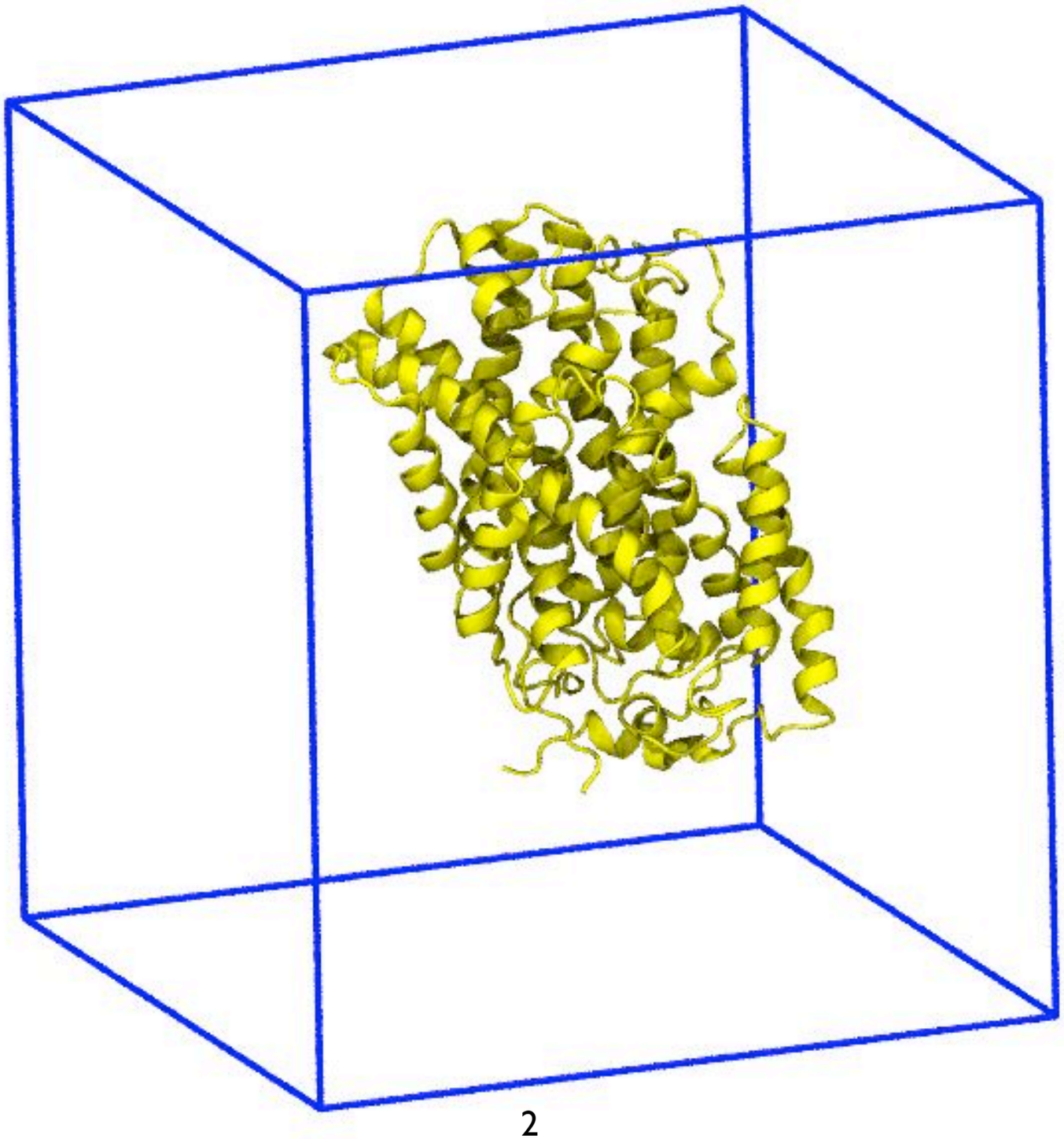
Oliver Beckstein

Arizona State University

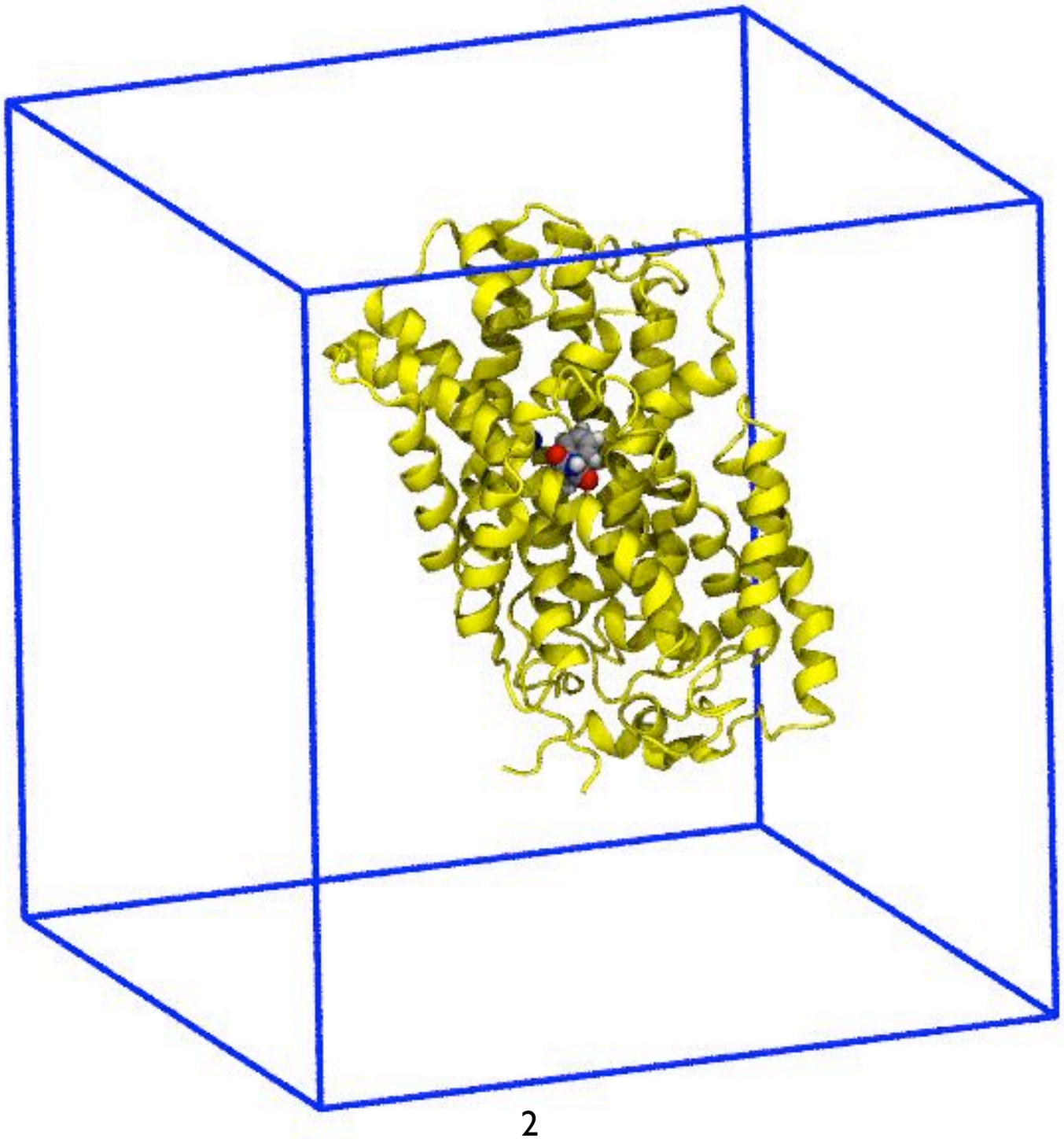
Membrane protein simulation system



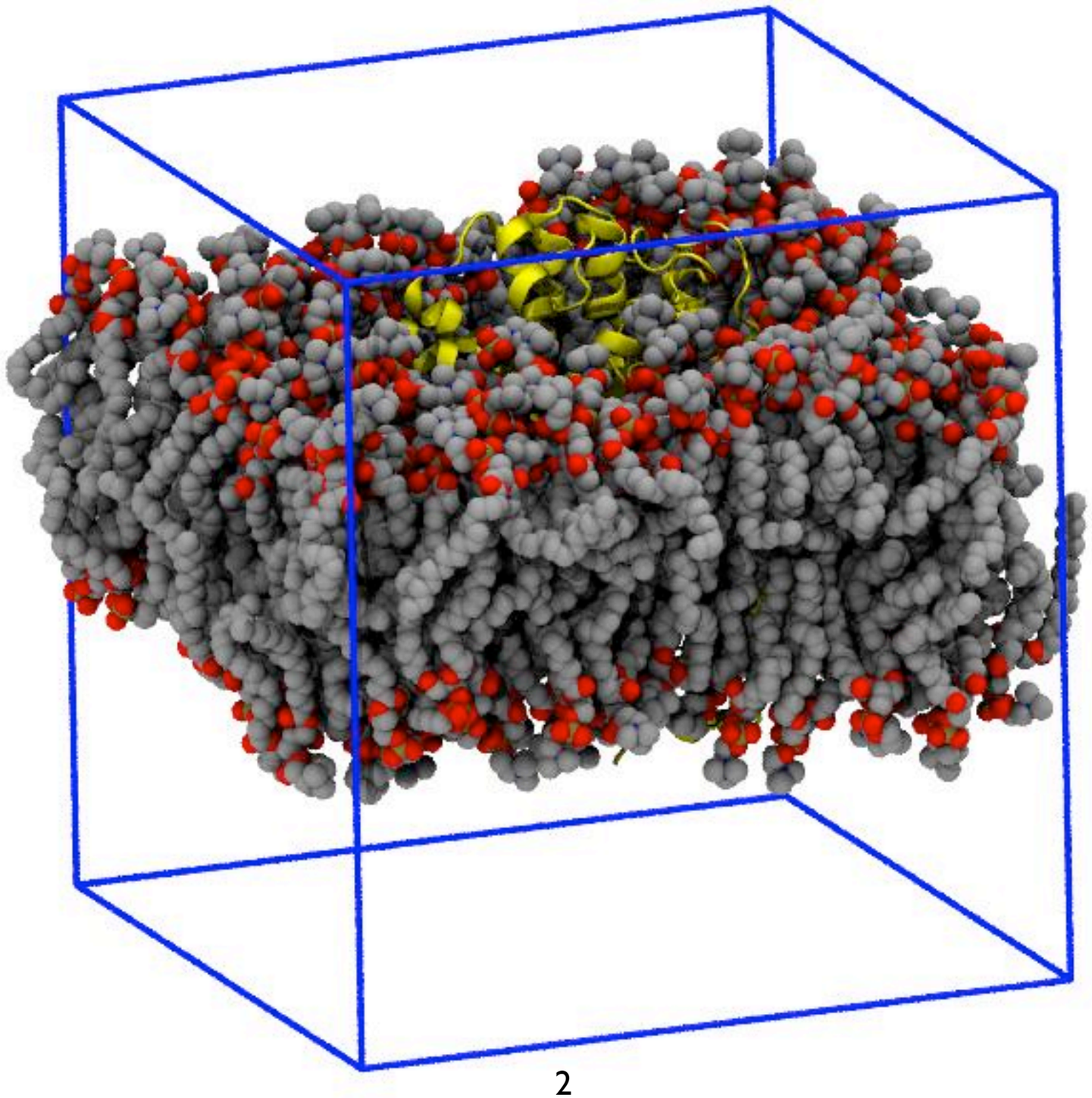
Membrane protein simulation system



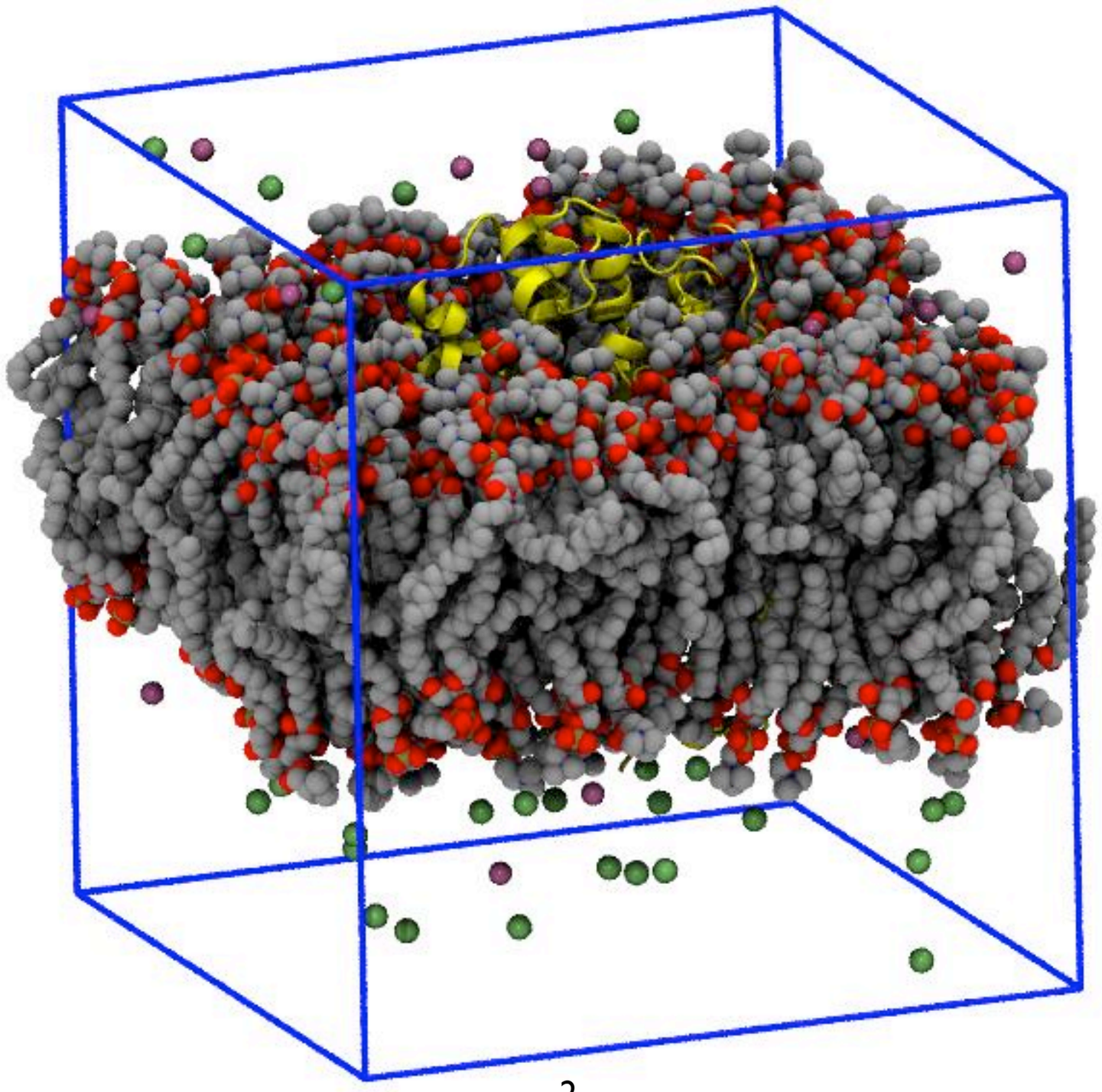
Membrane protein simulation system



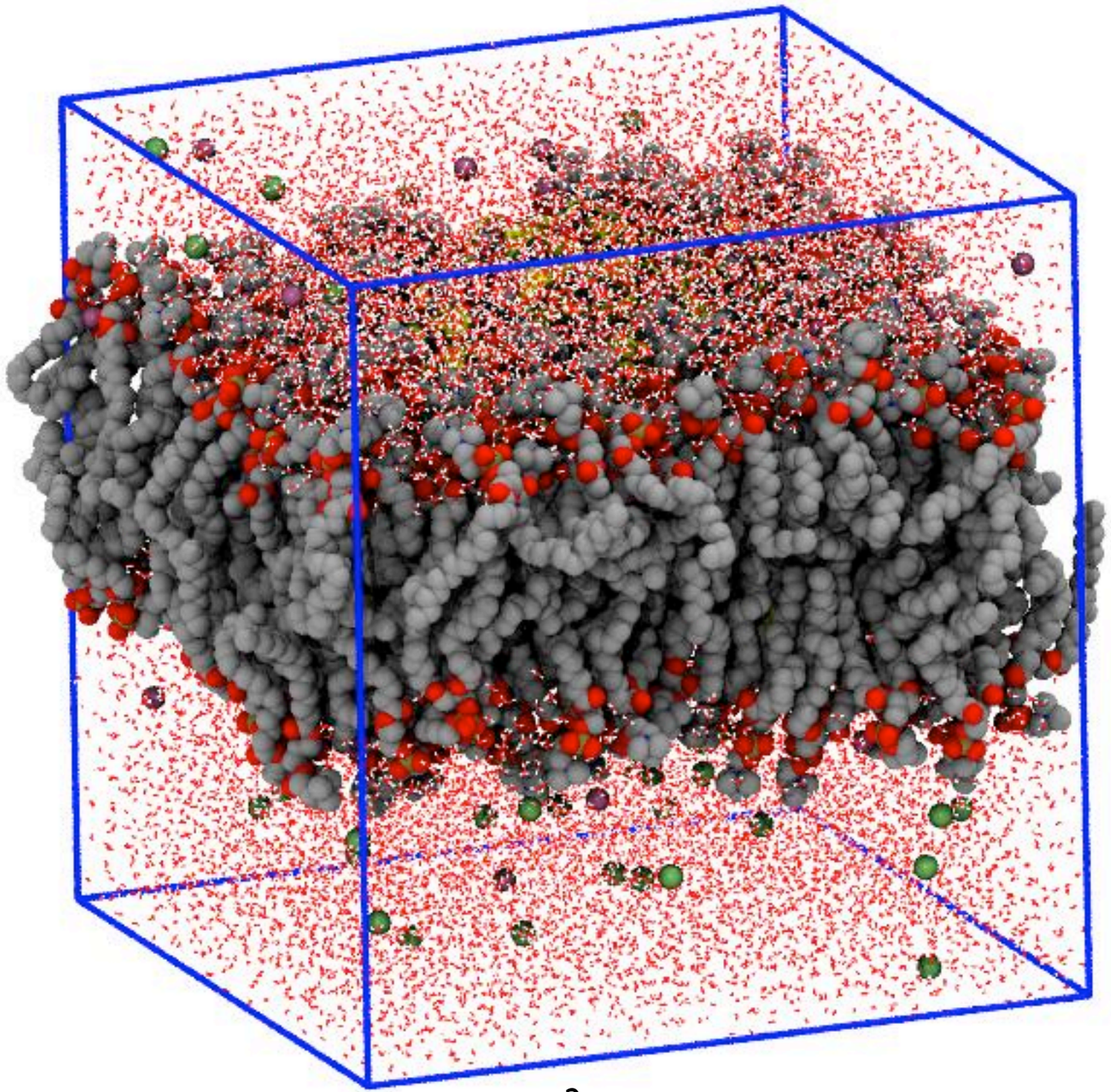
Membrane protein simulation system



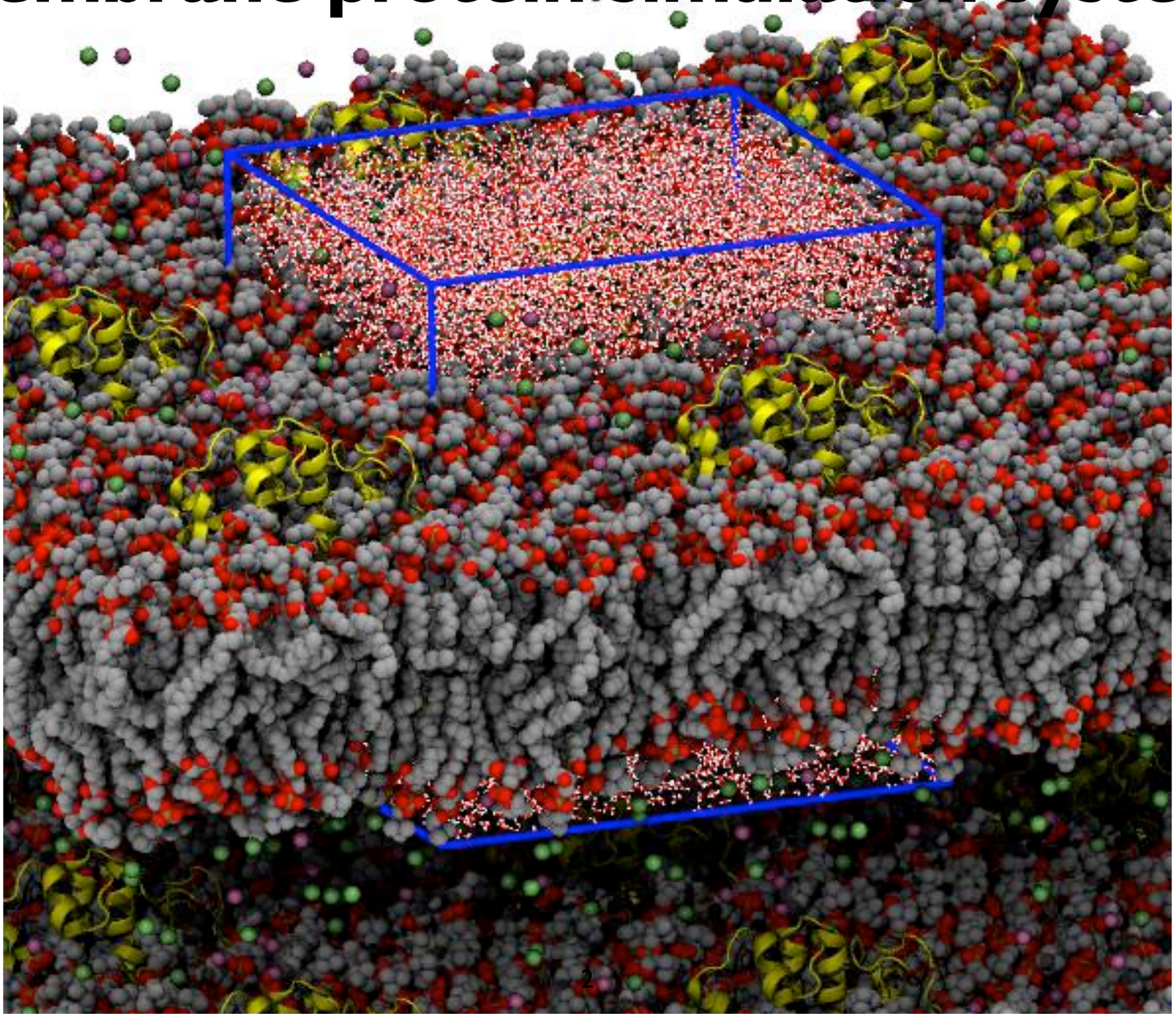
Membrane protein simulation system



Membrane protein simulation system



Membrane protein simulation system



Molecular Dynamics (MD) Simulations

(classical)

$$U(\mathbf{r}_1, \ldots, \mathbf{r}_N) = U_{\text{bonded}} + U_{\text{nonbonded}}$$

$$\mathbf{F}_i = -\nabla_i U(\mathbf{r}_1, \ldots, \mathbf{r}_i, \ldots, \mathbf{r}_N)$$

"force field"

$$\mathbf{F}_i = m_i \mathbf{a}_i$$

Newton's 2nd law

$$\frac{d^2\mathbf{r}_i(t)}{dt^2} = \frac{\mathbf{F}_i}{m_i}$$

integrator

$$\mathbf{r}_i(t + \Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \Delta t) + \frac{\mathbf{F}_i}{m_i}\Delta t^2$$

trajectory

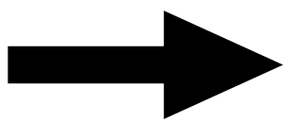
$$(\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t))$$

$$0 \le t \le \tau$$

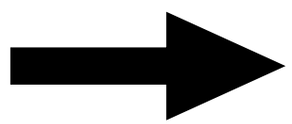
	max (2015)	typical (2015)
size *N*	~10^7	~10^5
simulated time τ	~10 μs	0.1–1 μs
trajectory frames	~10^9	~10^5
trajectory size	< 10 TiB	150 GiB

see also: T. Cheatham and D. Roe. Computing in Science Engineering, 17:30–39, 2015.

simulation trajectory



"accessible" structured data



analysis algorithm

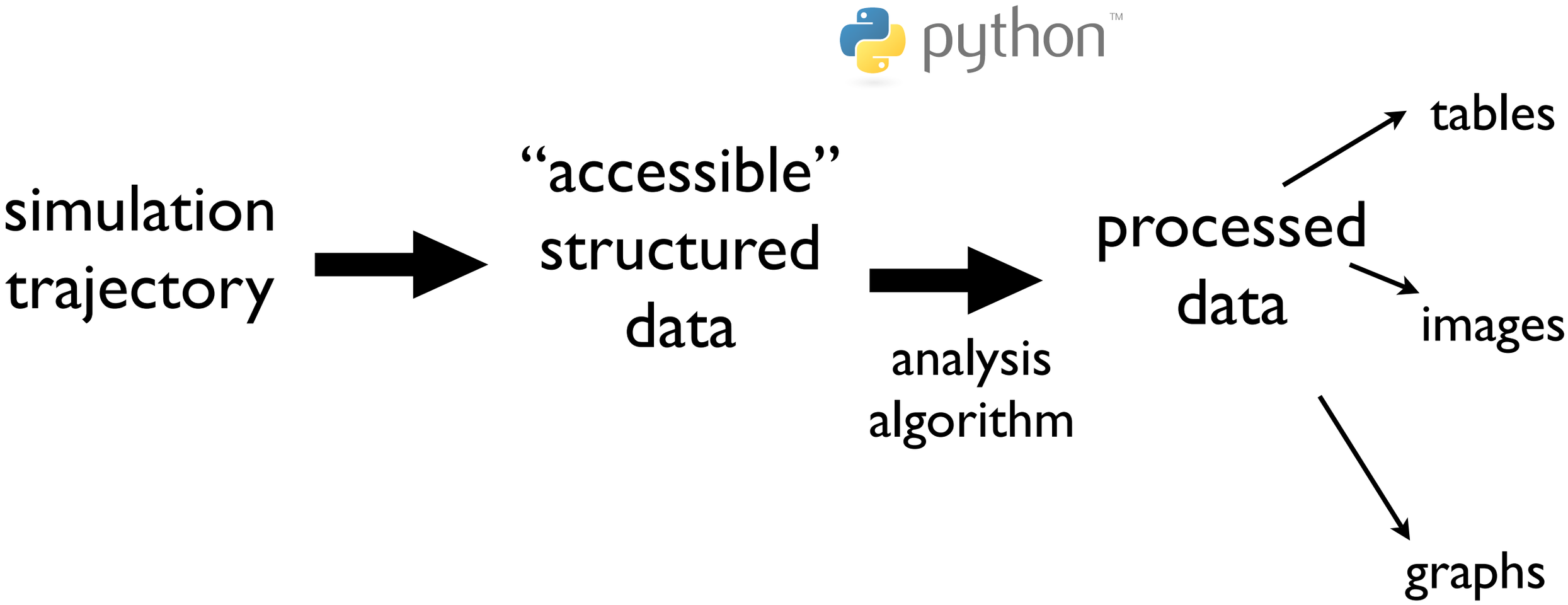
processed data

tables

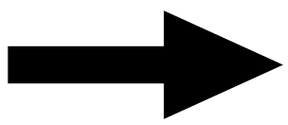
images

graphs

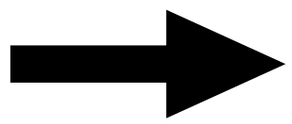
RESULTS!


python™
simulation trajectory
"accessible" structured data
analysis algorithm
processed data
tables
images
graphs
RESULTS!

simulation trajectory



"accessible" structured data



analysis algorithm

processed data

tables

images

graphs

RESULTS!

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

processed data

tables

images

graphs

RESULTS!

MD ANALYSIS

NumPy

python™

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

processed data

tables

images

graphs

RESULTS!

MD ANALYSIS

NumPy

python™

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

Cα RMSF

processed data

tables

images

graphs

RESULTS!

MD ANALYSIS

NumPy

python™

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

```python
import numpy as np
import MDAnalysis as mda

u = mda.Universe("topol.tpr", "trj.xtc")
ca = u.select_atoms("name CA")
means = np.zeros((len(ca), 3))
sumsq = np.zeros_like(means)
for k, ts in enumerate(u.trajectory):
    sumsq += (k/(k+1.0)) * (ca.positions - means)**2
    means[:] = (k*means + ca.positions)/(k+1.0)
rmsf = np.sqrt(sumsq.sum(axis=1)/(k+1.0))

matplotlib.pyplot.plot(ca.residues.resids, rmsf)
```

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

Cα RMSF

processed data

tables

images

graphs

RESULTS!

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

processed data

tables

images

graphs

RESULTS!

```python
import numpy as np
import MDAnalysis as mda

u = mda.Universe("topol.tpr", "trj.xtc")
ca = u.select_atoms("name CA")
means = np.zeros((len(ca), 3))
sumsq = np.zeros_like(means)
for k, ts in enumerate(u.trajectory):
    sumsq += (k/(k+1.0)) * (ca.positions - means)**2
    means[:] = (k*means + ca.positions)/(k+1.0)
rmsf = np.sqrt(sumsq.sum(axis=1)/(k+1.0))

matplotlib.pyplot.plot(ca.residues.resids, rmsf)
```

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

$C\alpha$ RMSF

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

processed data

- tables
- images
- graphs

```python
import numpy as np
import MDAnalysis as mda

u = mda.Universe("topol.tpr", "trj.xtc")
ca = u.select_atoms("name CA")
means = np.zeros((len(ca), 3))
sumsq = np.zeros_like(means)
for k, ts in enumerate(u.trajectory):
    sumsq += (k/(k+1.0)) * (ca.positions - means)**2
    means[:] = (k*means + ca.positions)/(k+1.0)
rmsf = np.sqrt(sumsq.sum(axis=1)/(k+1.0))

matplotlib.pyplot.plot(ca.residues.resids, rmsf)
```

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

$C\alpha$ RMSF

RESULTS!

MD ANALYSIS

NumPy

python™

simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

```
import numpy as np
import MDAnalysis as mda

u = mda.Universe("topol.tpr", "trj.xtc")
ca = u.select_atoms("name CA")
means = np.zeros((len(ca), 3))
sumsq = np.zeros_like(means)
for k, ts in enumerate(u.trajectory):
    sumsq += (k/(k+1.0)) * (ca.positions - means)**2
    means[:] = (k*means + ca.positions)/(k+1.0)
rmsf = np.sqrt(sumsq.sum(axis=1)/(k+1.0))

matplotlib.pyplot.plot(ca.residues.resids, rmsf)
```

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

$C\alpha$ RMSF

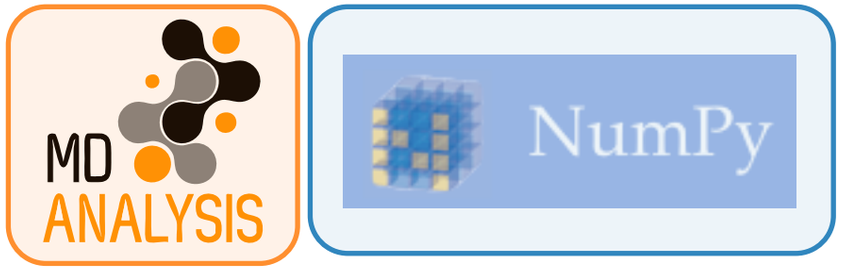
processed data

tables

images

graphs

RESULTS!



simulation trajectory

dcd, xtc, trr, ncdf, trj, pdb, pqr, gro, crd, dms, trz, mol2, xyz, config, history, gms, ...

psf, tpr, prmtop, dms, mol2, hoomd xml, ...

"accessible" structured data

analysis algorithm

processed data

tables

images

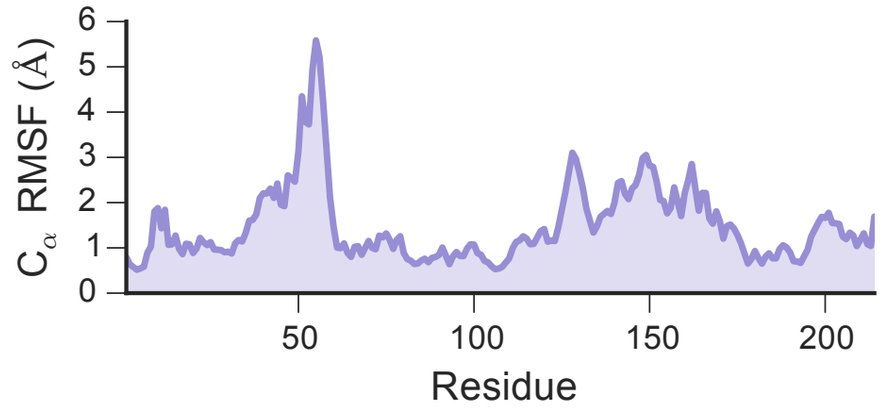
graphs

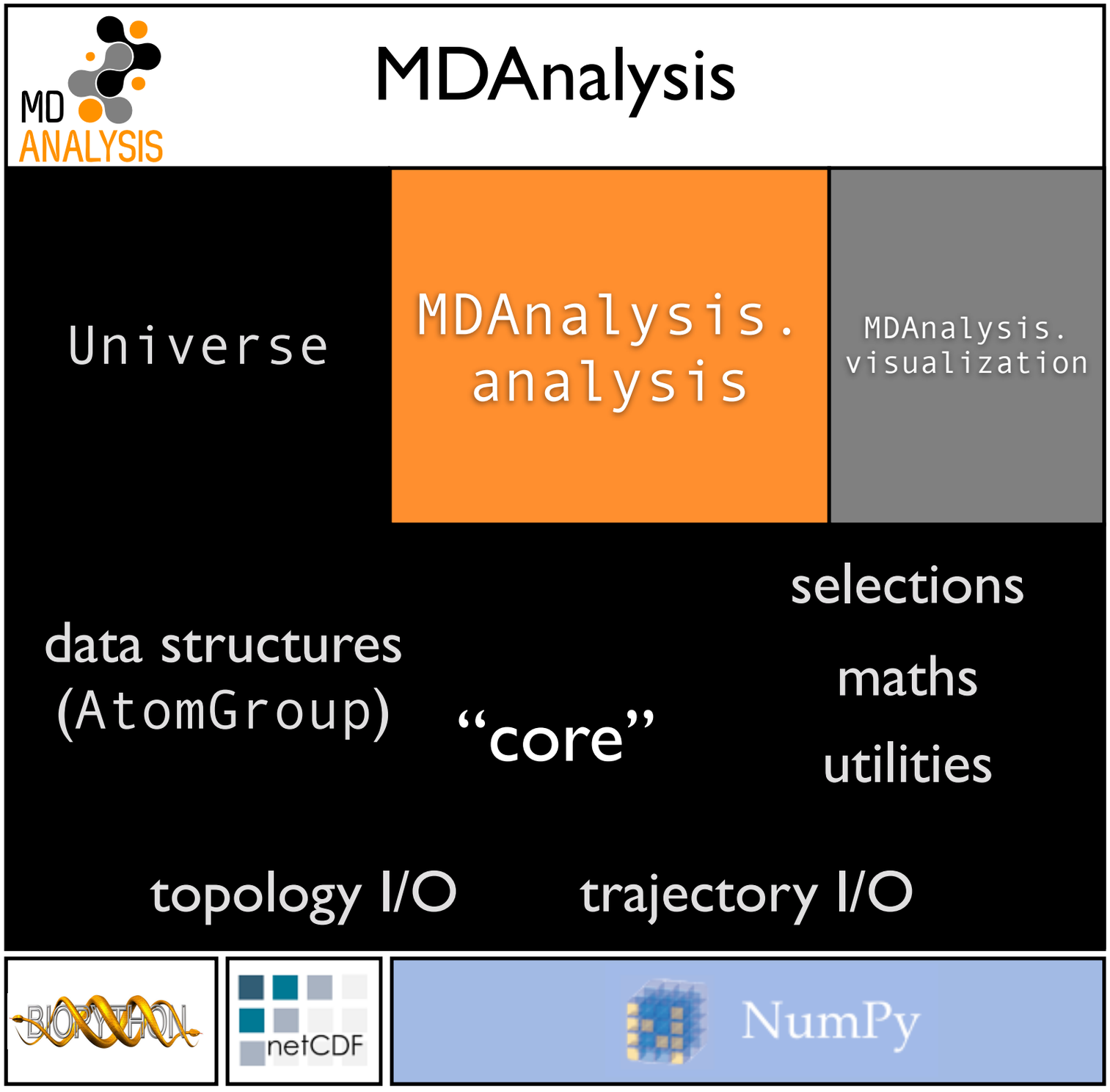
```python
import numpy as np
import MDAnalysis as mda

u = mda.Universe("topol.tpr", "trj.xtc")
ca = u.select_atoms("name CA")
means = np.zeros((len(ca), 3))
sumsq = np.zeros_like(means)
for k, ts in enumerate(u.trajectory):
    sumsq += (k/(k+1.0)) * (ca.positions - means)**2
    means[:] = (k*means + ca.positions)/(k+1.0)
rmsf = np.sqrt(sumsq.sum(axis=1)/(k+1.0))

matplotlib.pyplot.plot(ca.residues.resids, rmsf)
```

$$\rho_i = \sqrt{\left\langle (\mathbf{x}_i(t) - \langle \mathbf{x}_i \rangle)^2 \right\rangle}$$

Cα RMSF

RESULTS!





Code base:

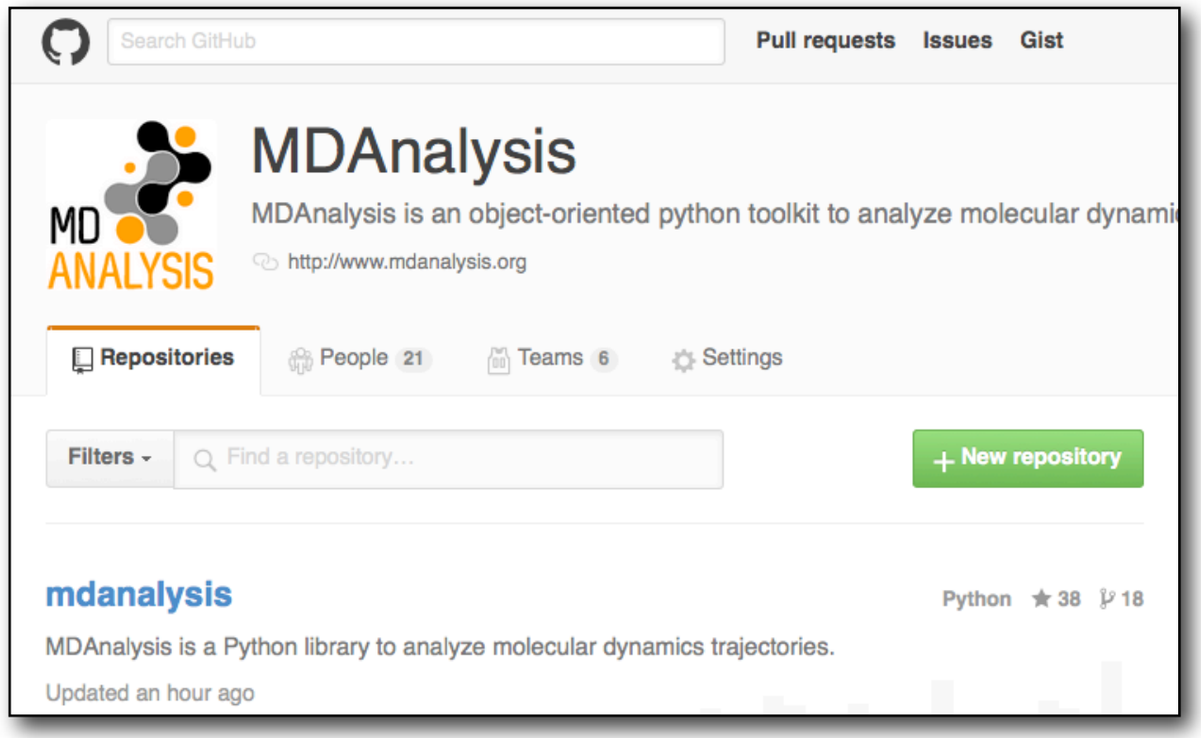
- python 2.7
- cython
- C
- ~75k LOC
- ~37k lines comments

Runs on:

- Linux
- Mac OS X

Open source

- GPL v2
- github.com/MDAnalysis



Open and inclusive community:

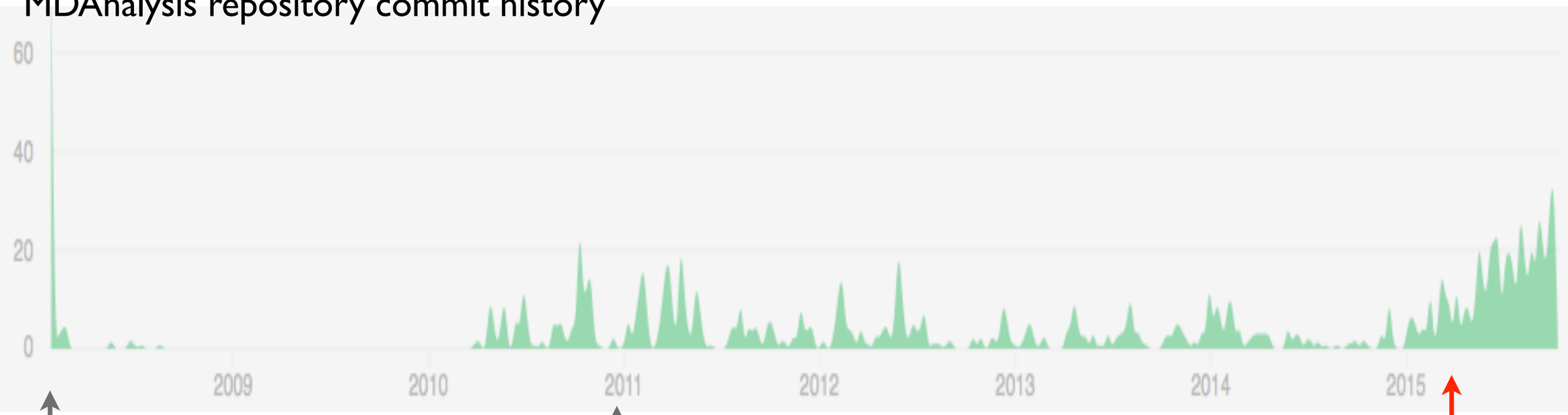
- questions are answered (mailing list)
- *pull requests* welcome!
- community code review
- continuous integration with > 2,500 unit tests
- 36 contributing authors (Oct 2015)

Naveen Michaud-Agrawal, Elizabeth J. Denning, Joshua Adelman, Jonathan Barnoud, Christian Beckstein (logo), Alejandro Bernardin, Sébastien Buchoux, David Caplan, Matthieu Chavent, David L. Dotson, Xavier Deupi, Jan Domański, Lennard van der Feltz, Philip Fowler, Joseph Goose, Richard J. Gowers, Lukas Grossar, Benjamin Hall, Joe Jordan, Max Linke, Jinju Lu, Robert McGibbon, Alex Nesterenko, Manuel Nuno Melo, Caio S. Souza, Danny Parton, Joshua L. Phillips, Tyler Reddy, Paul Rigor, Sean L. Seyler, Andy Somogyi, Lukas Stelzl, Gorman Stock, Isaac Virshup, Zhuyi Xue, Carlos Yáñez S, and Oliver Beckstein

Join us at www.mdanalysis.org

github.com/MDAnalysis

MDAnalysis repository commit history




```
Author: Naveen Michaud-Agrawal
Date:   Thu Jan 31 11:42:33 2008 +0000

initial import
```

GitHub

Software News and Updates
MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations

NAVEEN MICHAUD-AGRAWAL,[1] ELIZABETH J. DENNING,[1,2] THOMAS B. WOOLF,[1,3] OLIVER BECKSTEIN[3,4]

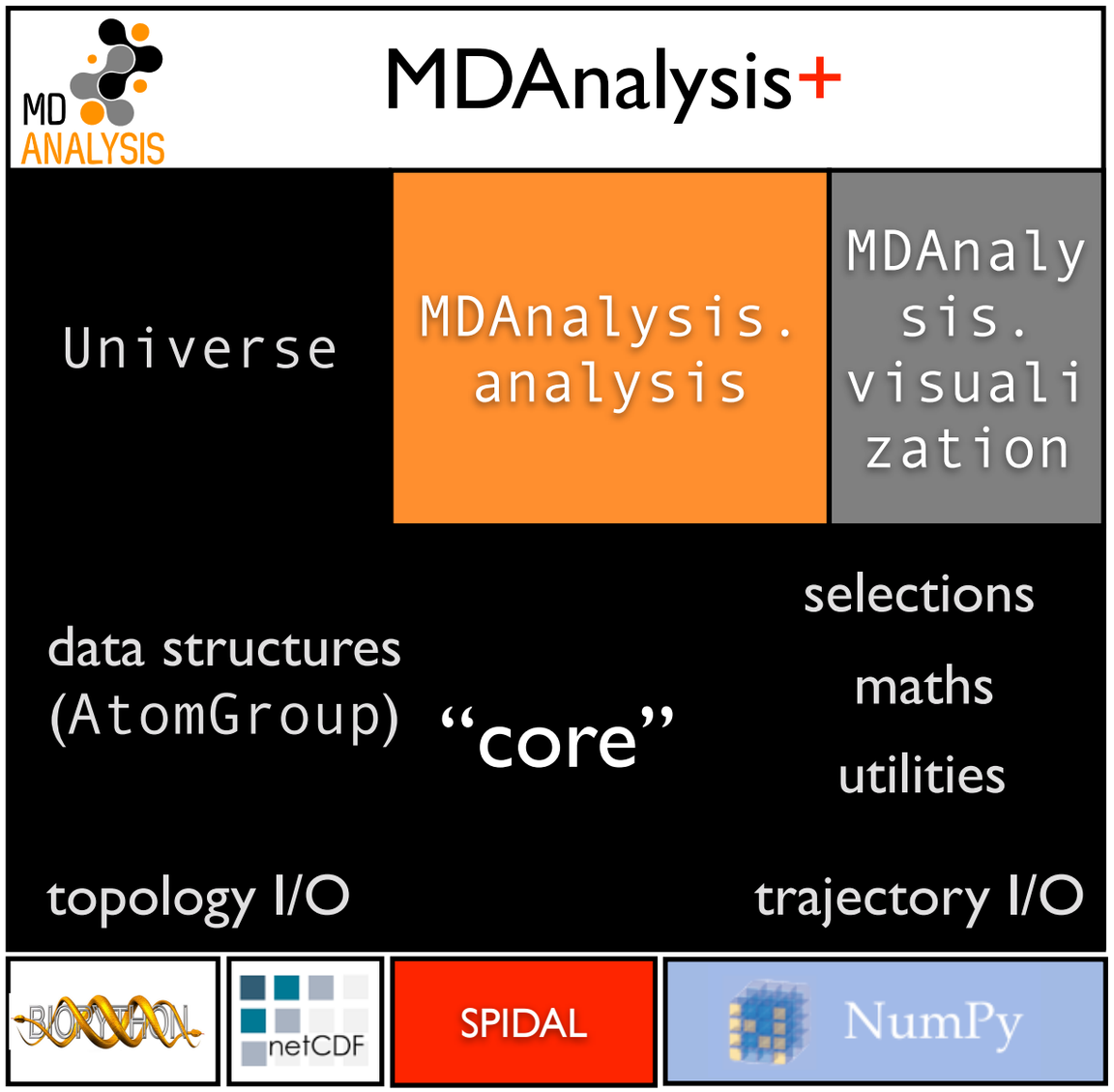
Received 23 October 2010; Revised 6 February 2011; Accepted 12 February 2011
DOI 10.1002/jcc.21787
Published online 15 April 2011 in Wiley Online Library (wileyonlinelibrary.com).

J Comput Chem 32: 2319–2327, 2011

(~140 citations on GoogleScholar (Oct 2015))

Goal: MDAnalysis + SPIDAL

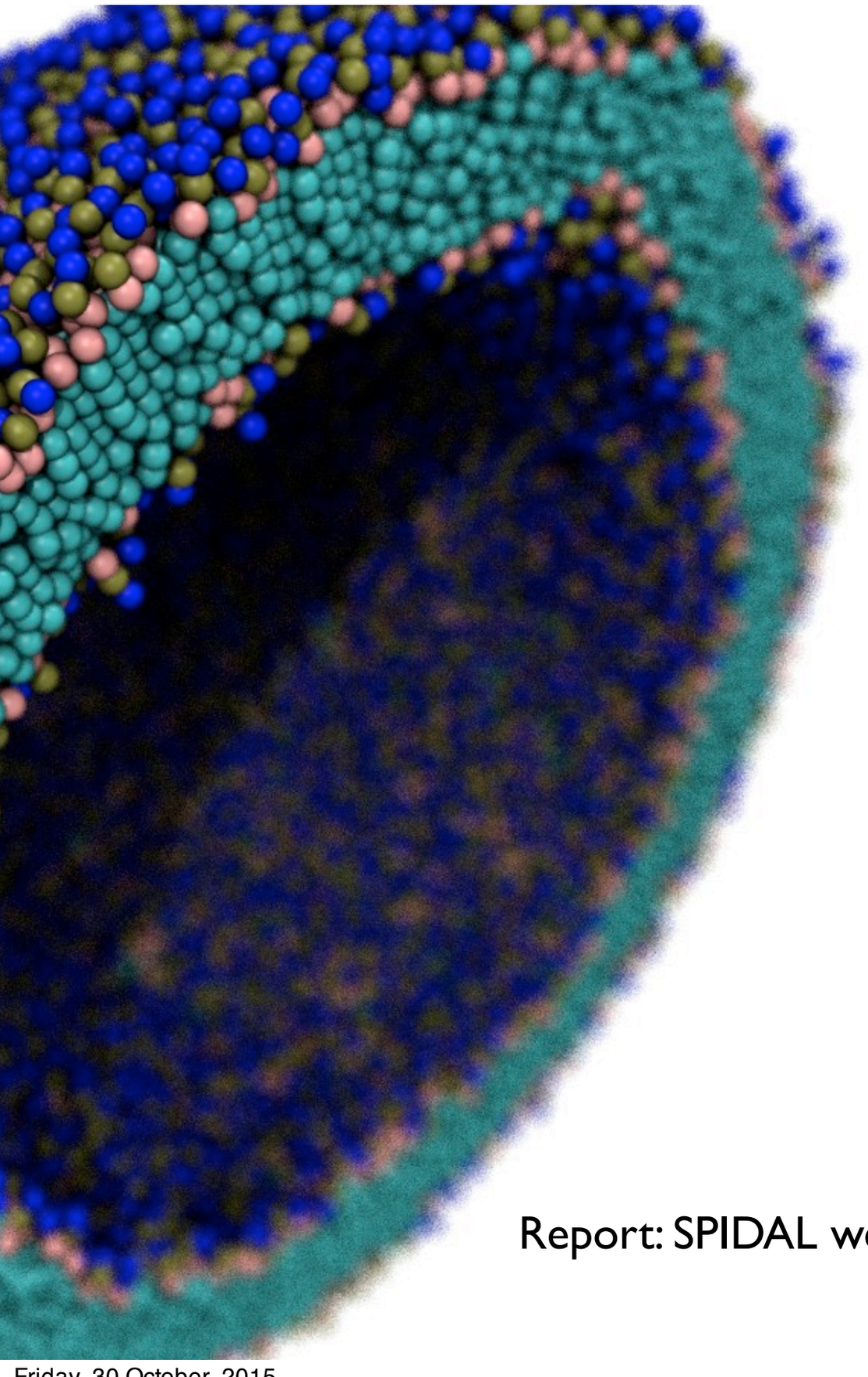
- make SPIDAL algorithms available inside MDAnalysis
- bring "BigData" approaches to the molecular simulation community (with low barrier to entry)



Challenges and Areas of Interest

- large systems (> 1 million particles)
- long trajectories (> 100,000 frames)
  - only one frame in memory
  - can be I/O limited
- multiple related trajectories (replica exchange, windowed free energy calculations)
- specific algorithms (distance search, clustering, ...)

Ian Kenney (REU): Large system benchmark

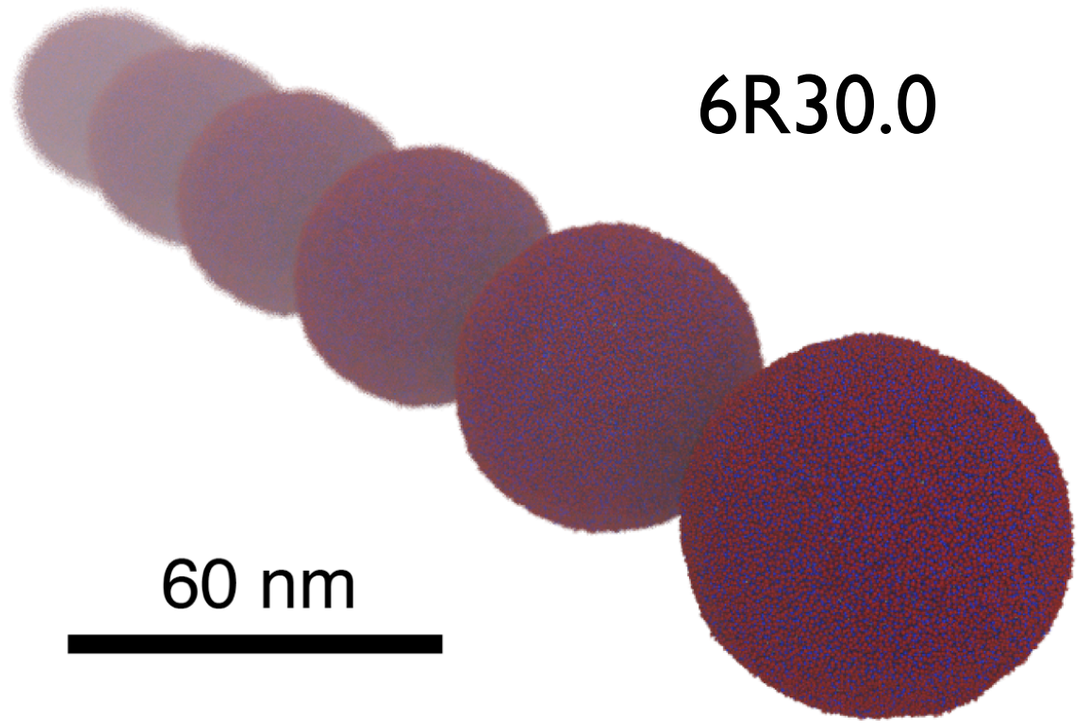


- non-trivial biologically relevant system with adjustable number of particles
- *lipid vesicles* (molecular packages – drug delivery, neurotransmission, cancer)
- *Dry Martini* force field (implicit solvent, coarse-grained lipids)
- science: study vesicle interactions (e.g. fusion)

Report: SPIDAL webpage and http://dx.doi.org/10.6084/m9.figshare.1588804

Benchmark: Vesicle Library

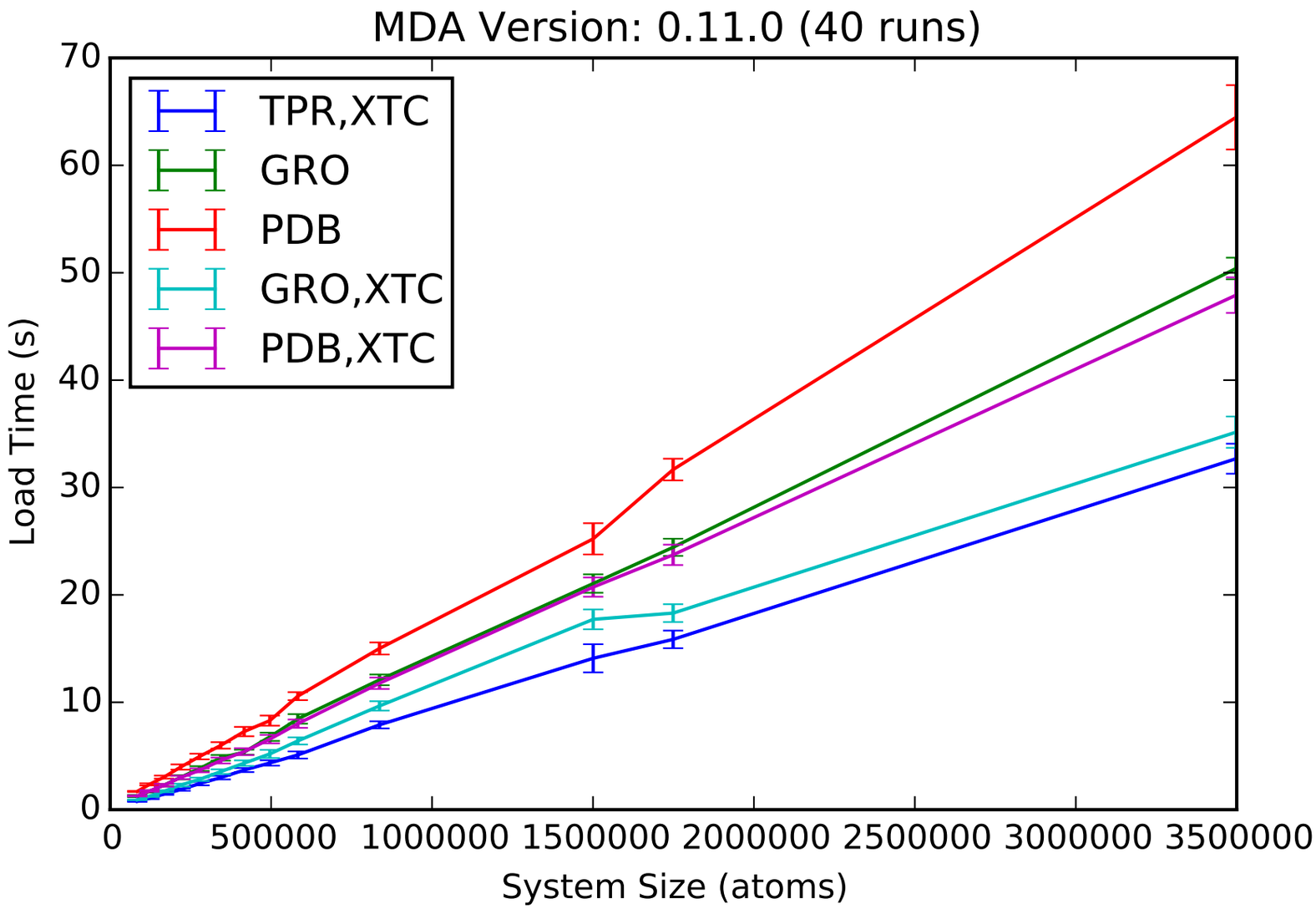
label	vesicles	radius (nm)	atoms	lipids
R10.0	1	10	84192	7016
R12.5	1	12.5	122208	10184
R15.0	1	15	167352	13946
R17.5	1	17.5	219000	18250
R20.0	1	20	277728	23144
R22.5	1	22.5	343500	28625
R25.0	1	25	416208	34684
R27.5	1	27.5	496044	41337
R30.0	1	30	582984	48582
3R30.0	3	30	1748952	145746
6R30.0	6	30	3497904	291492



Modular approach: combine vesicles into larger systems

library on GitHub: https://github.com/Becksteinlab/vesicle_library

MDAnalysis: system load time

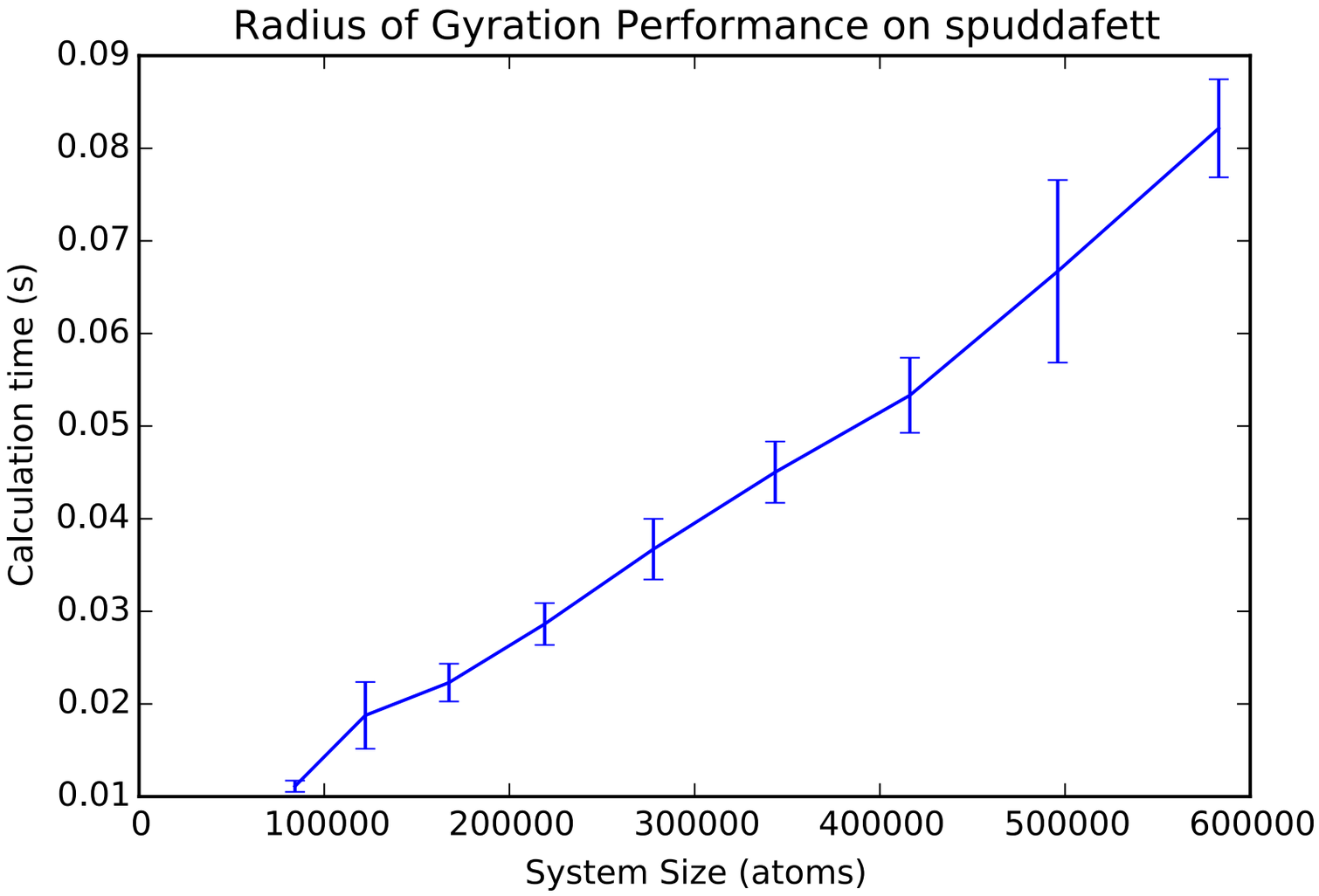


- Which parts need to be improved for handling large systems?
- Initial load: fixed cost (possibly important for parallel analysis of trajectories)

Simple frame-based analysis: radius of gyration

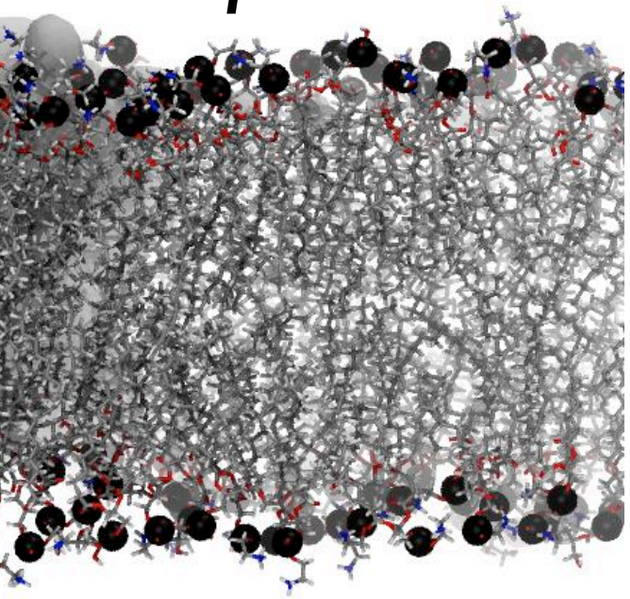
- calculate time series (one calculation per frame)

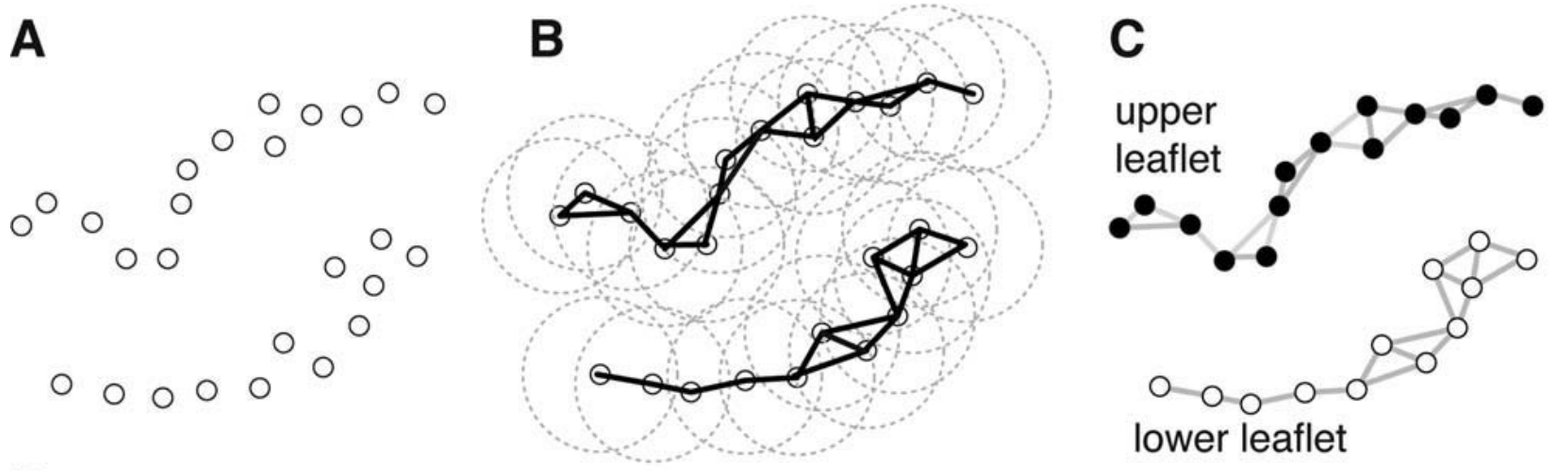
$$R_g(t) = \sqrt{\frac{1}{M}\sum_{i=1}^{N} m_i(\mathbf{r}_i(t) - \mathbf{R}(t))}$$



- serial (1 core)
- 100 ms per frame
- … but for 1M frames would take 28h
- pleasingly parallel (work with Shantenu Jha on parallel approaches, e.g. RADICAL.pilot)

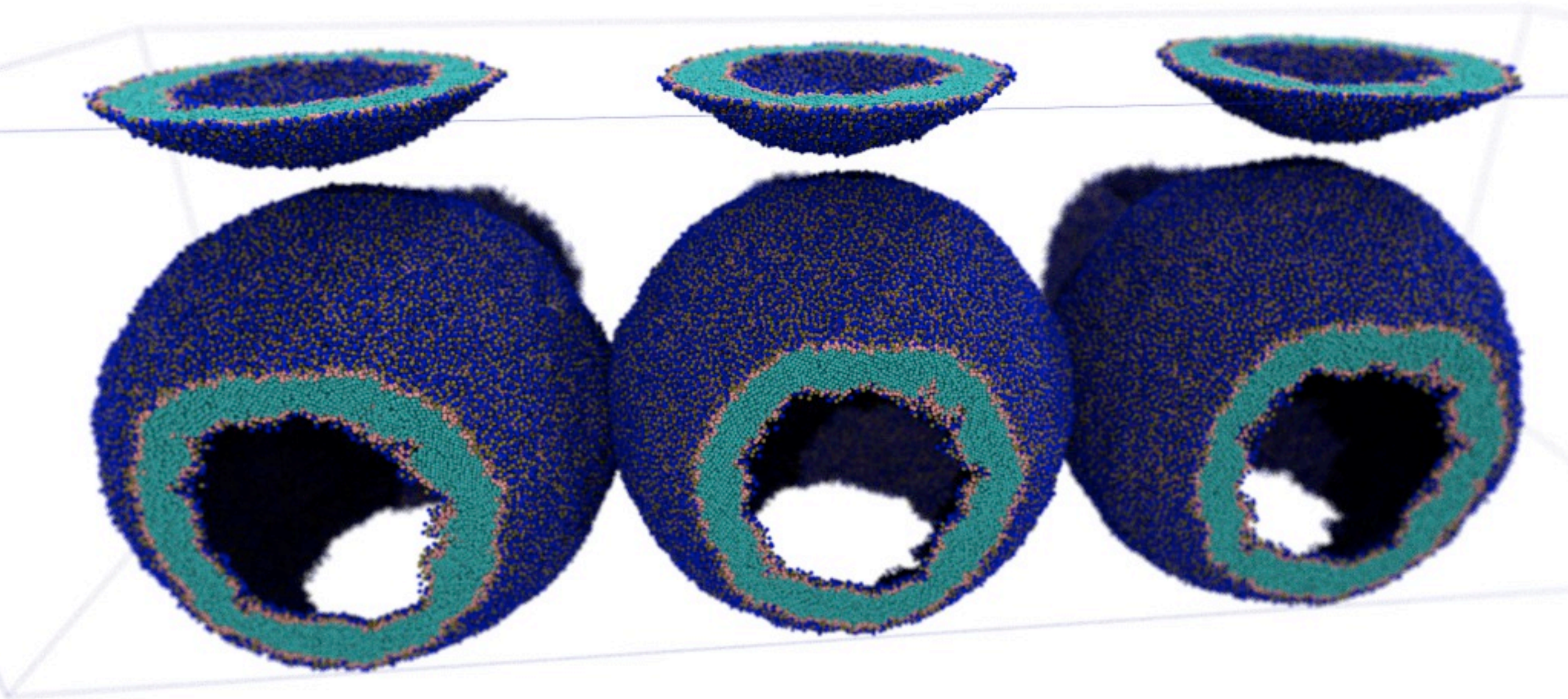
LeafletFinder

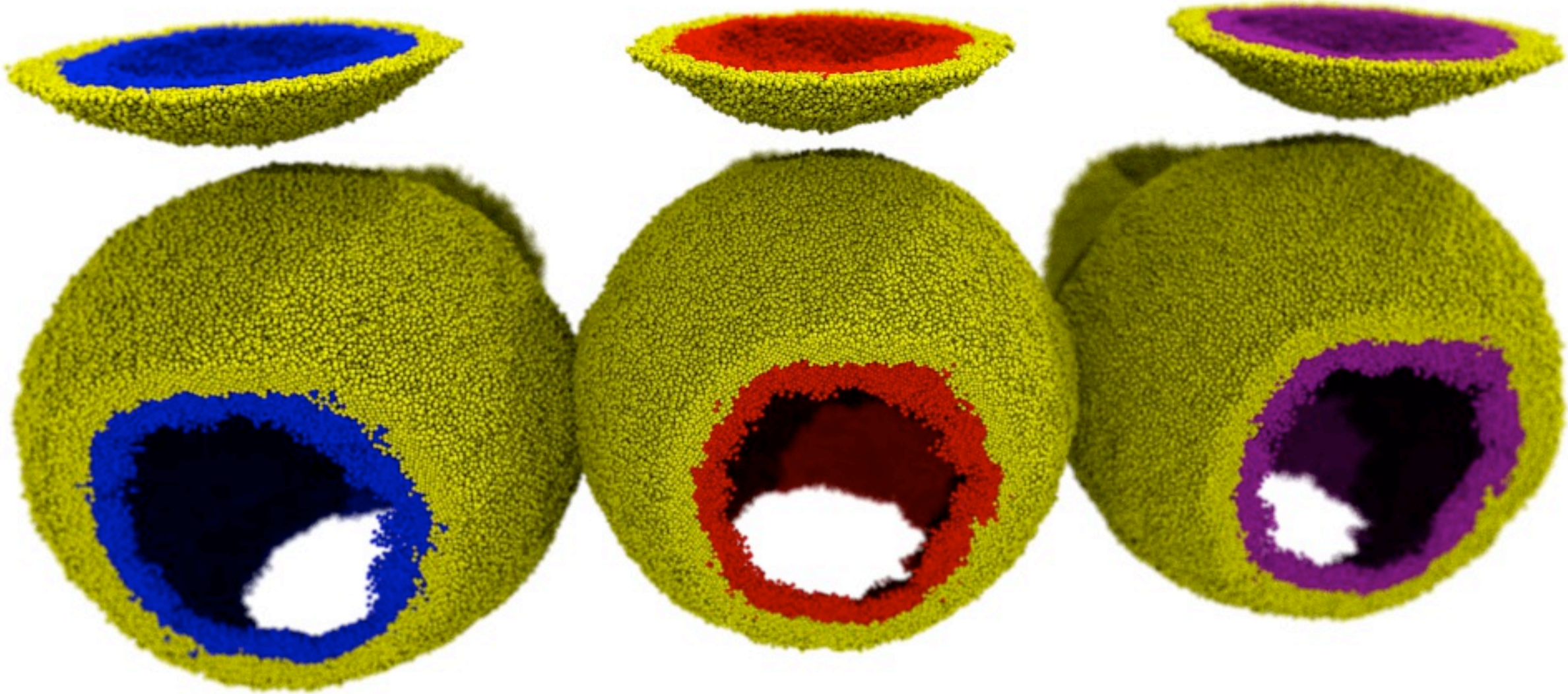


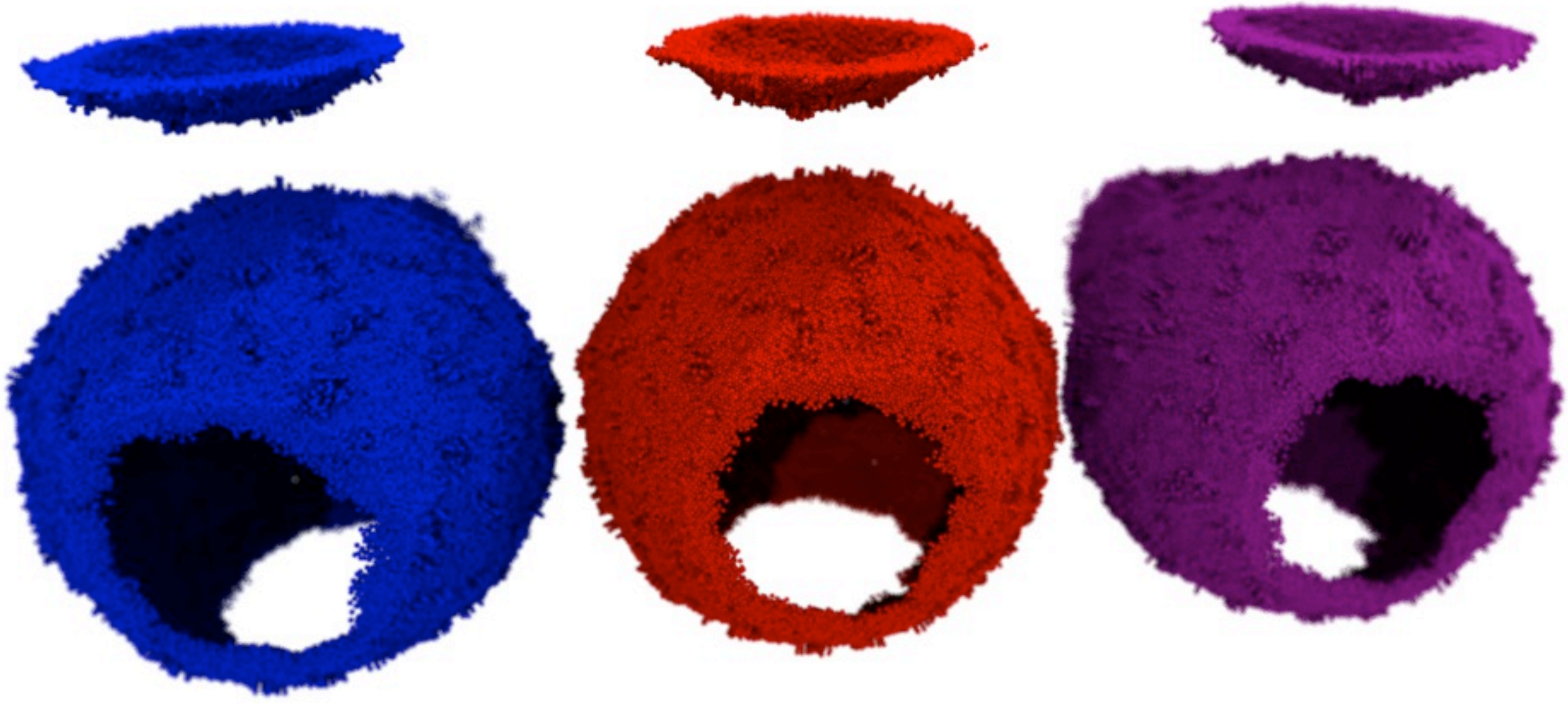


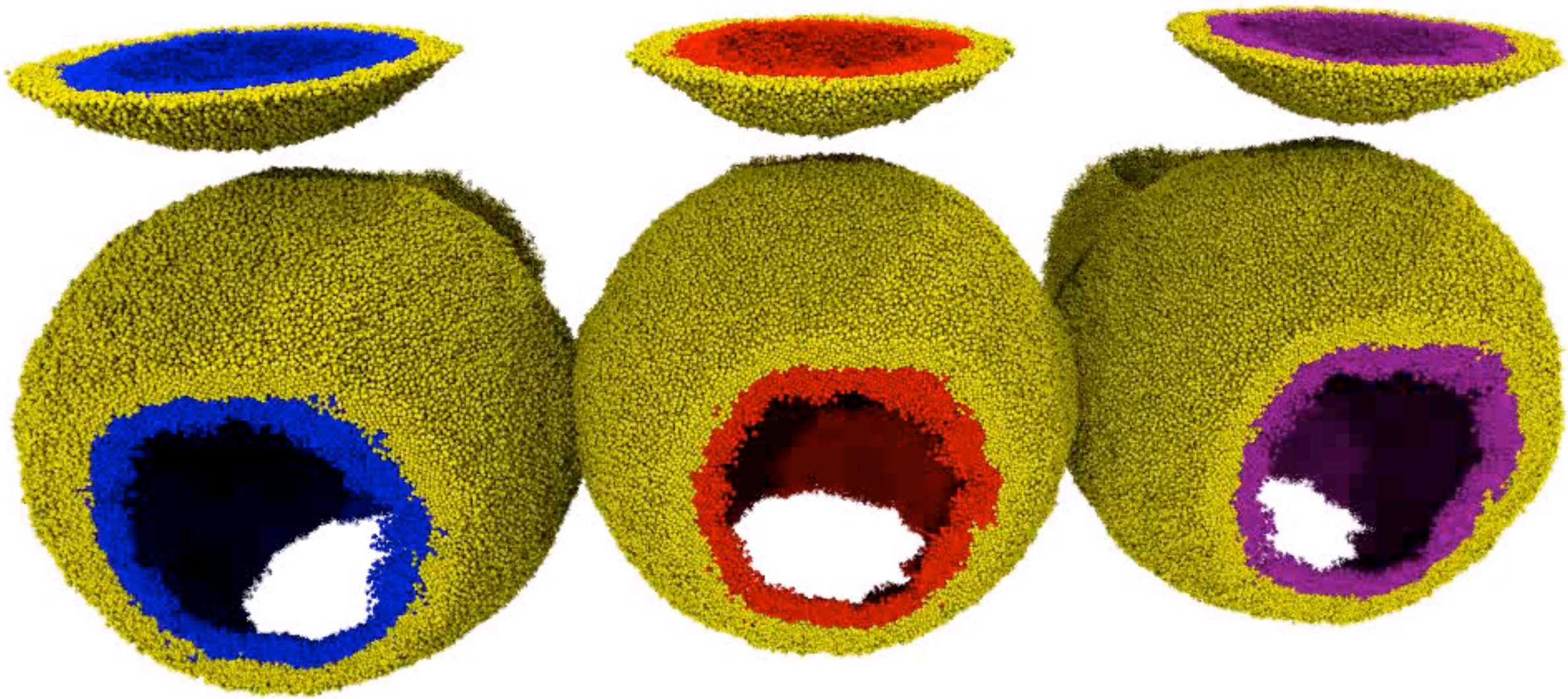
A
B
C

```
import MDAnalysis as mda
import networkx as nx
from MDAnalysis.lib.distances import distance_array

u = mda.Universe(pdb, xtc)
headgroup_atoms = u.select_atoms("name P*")
x = headgroup_atoms.positions

adj = (distance_array(x, x) < 12)
leaflets = sorted(nx.connected_components(nx.Graph(adj)), key=len, reverse=True)

A_lipids = headgroup_atoms[leaflets[0]].residues
B_lipids = headgroup_atoms[leaflets[1]].residues
```

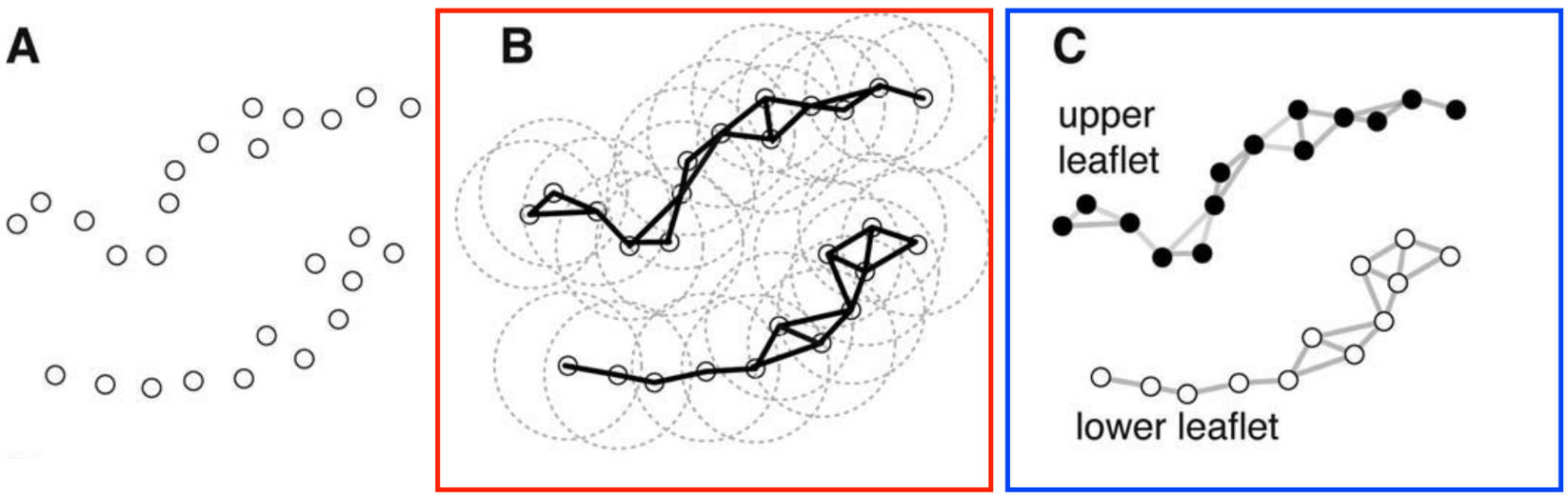








- *LeafletFinder* (and other "interaction network" approaches): slow on big systems
  - need to use sparse distance matrix (full matrix ~2 TiB RAM)
  - 4 min per frame (!)
- need to improve the *nearest neighbor* step (SPIDAL!)
- work with Shantenu Jha to explore different algorithms for the network analysis

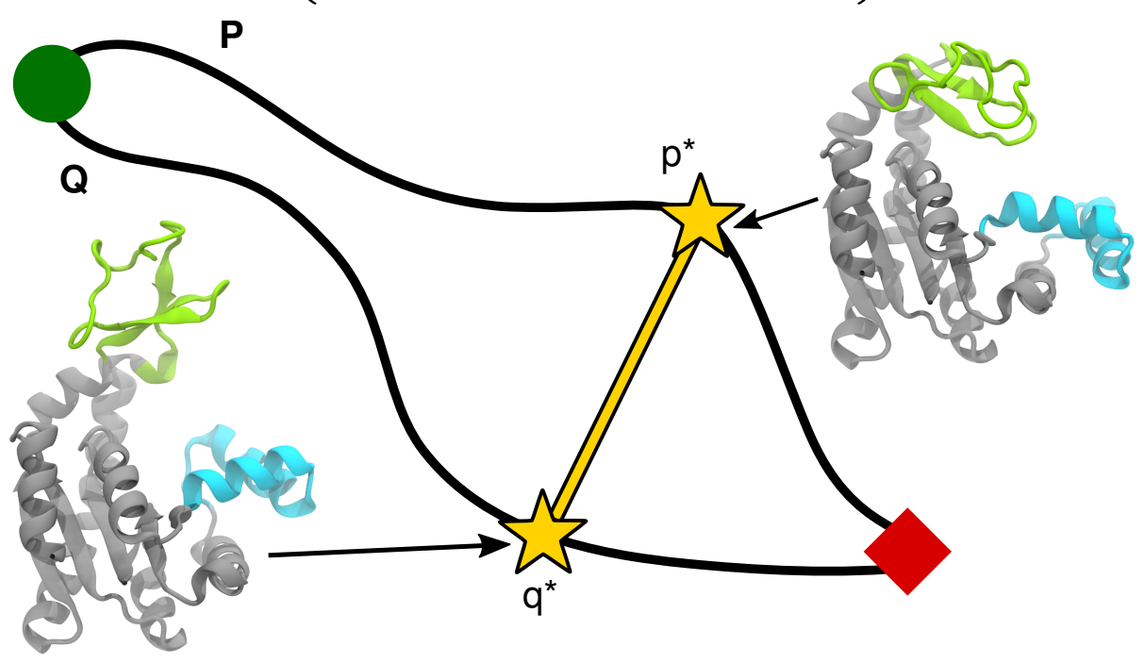


Other areas of interest

- fast histogramming of coordinates in 3D space to calculate densities
- nearest neighbor / distance matrices $d_{ij}(t) = |\mathbf{r}_i(t) - \mathbf{r}_j(t)|$
- fast RMSD calculations after optimum rigid body superposition

$$\delta_H(P,Q) = \max\left\{\max_{p\in P}\min_{q\in Q} d(p,q), \max_{q\in Q}\min_{p\in P} d(q,p)\right\}$$

- clustering (e.g. for Markov State Models)
- *Path Similarity Analysis** (with Shantenu Jha)



*S. L. Seyler, A. Kumar, M. F. Thorpe, and O. Beckstein. Path similarity analysis: A method for quantifying macromolecular pathways. PLoS Comput Biol, 11(10):e1004568, 10 2015. doi: 10.1371/journal.pcbi. 1004568..

Acknowledgements

Beckstein Lab
Computational Biophysics at Arizona State University

Ian Kenney

Sean Seyler

David Dotson

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Ioannis Paraskevakos,
Andre Luckow
Shantenu Jha

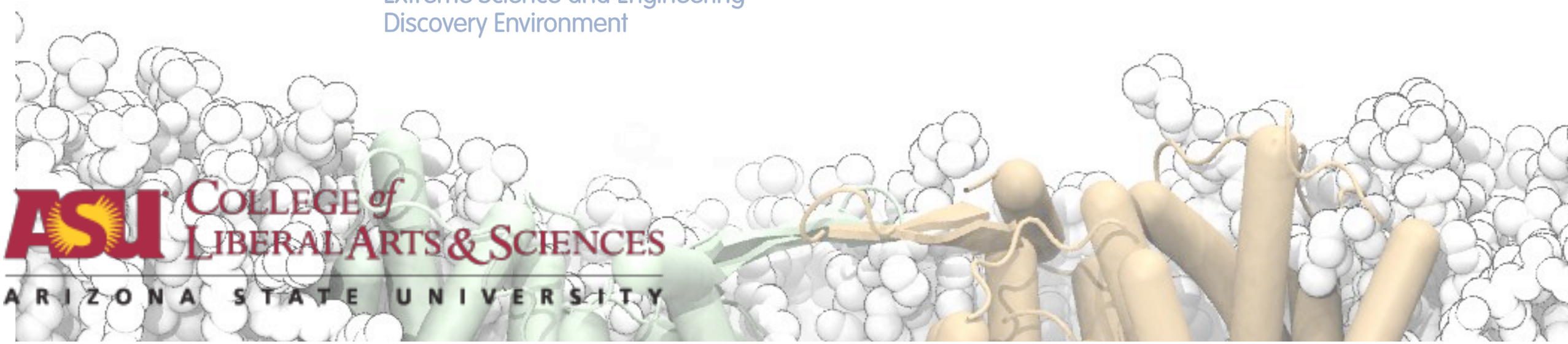
MDANALYSIS

Naveen Michaud-Agrawal, Elizabeth J. Denning, Joshua Adelman, Jonathan Barnoud, Christian Beckstein (logo), Alejandro Bernardin, Sébastien Buchoux, David Caplan, Matthieu Chavent, David L. Dotson, Xavier Deupi, Jan Domański, Lennard van der Feltz, Philip Fowler, Joseph Goose, Richard J. Gowers, Lukas Grossar, Benjamin Hall, Joe Jordan, Max Linke, Jinju Lu, Robert McGibbon, Alex Nesterenko, Manuel Nuno Melo, Caio S. Souza, Danny Parton, Joshua L. Phillips, Tyler Reddy, Paul Rigor, Sean L. Seyler, Andy Somogyi, Lukas Stelzl, Gorman Stock, Isaac Virshup, Zhuyi Xue, Carlos Yáñez S, and Oliver Beckstein

NSF

XSEDE
Extreme Science and Engineering Discovery Environment

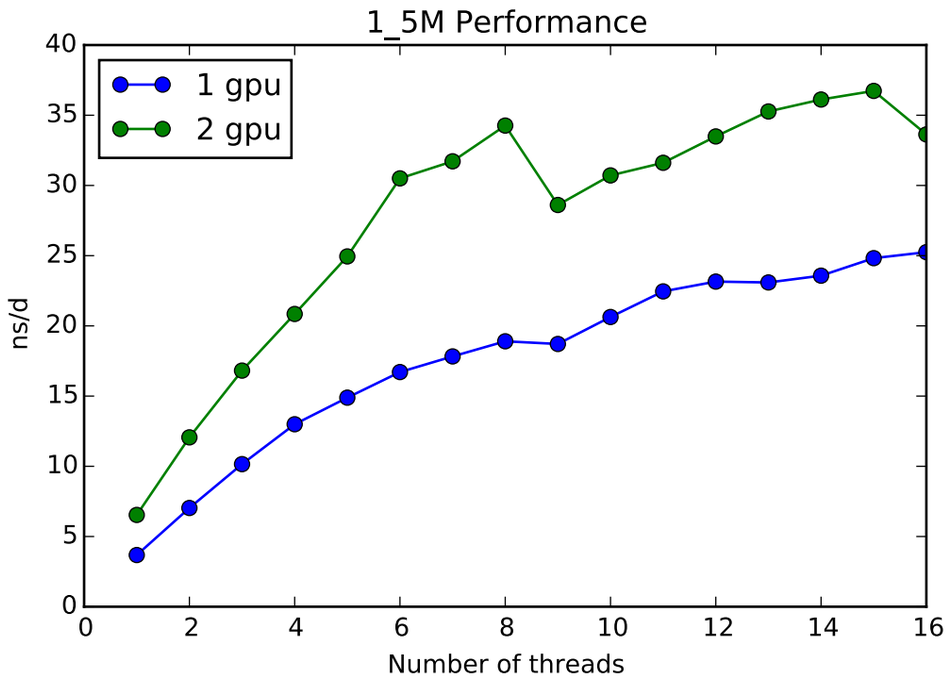
ASU COLLEGE of LIBERAL ARTS & SCIENCES
ARIZONA STATE UNIVERSITY



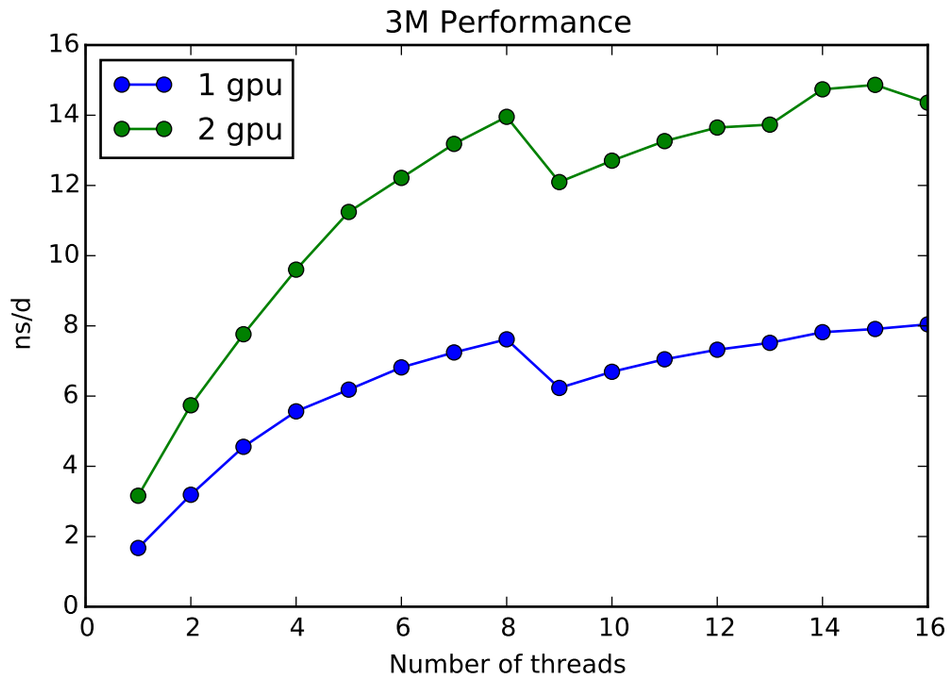
Appendix

Benchmark: Gromacs Performance on Desktop

3R30.0 (1.7 M particles)

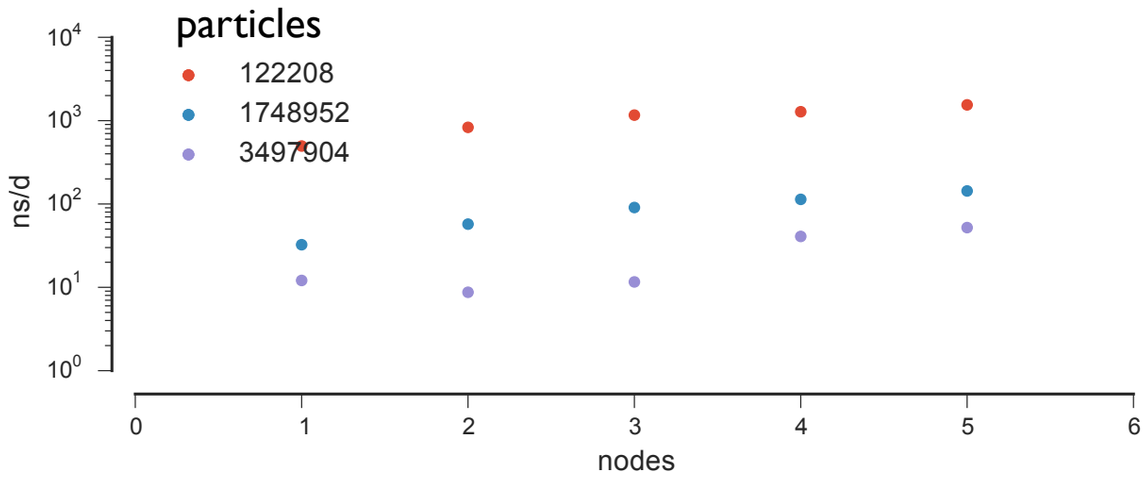


6R30.0 (3.4 M particles)

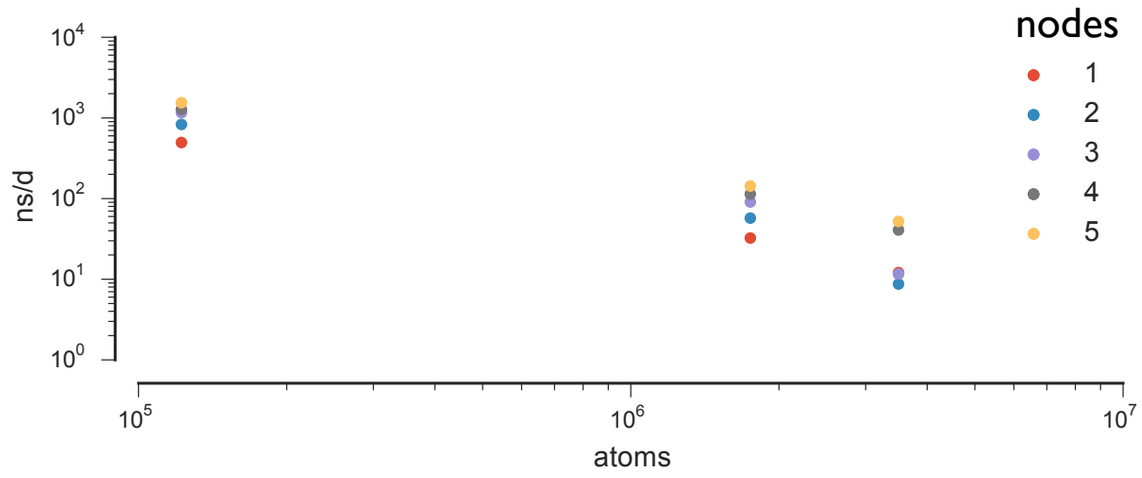


- simulation with Gromacs 5.0.5 on local workstation (16 core Intel Sandy Bridge 2.6 GHz, NVIDIA GTX 690)

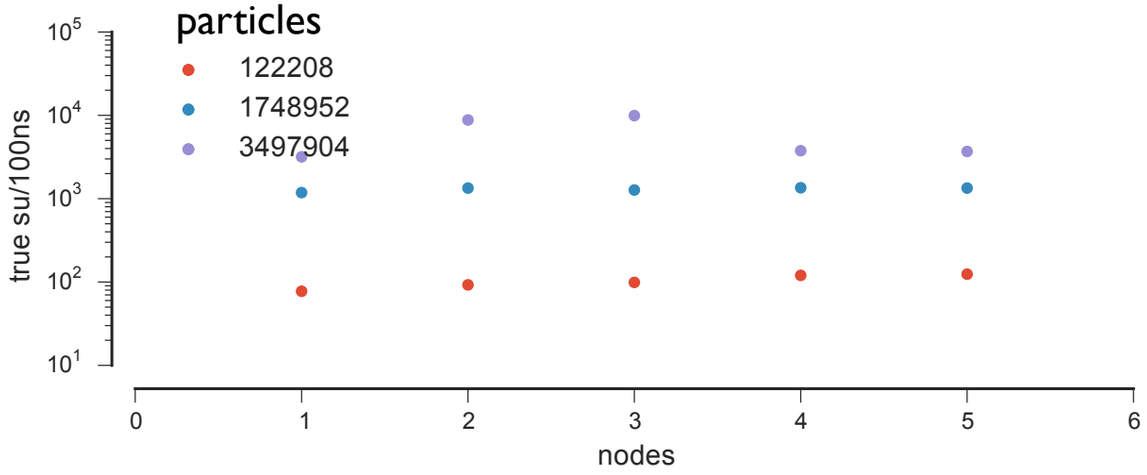
Benchmark: Gromacs Performance on *stampede*



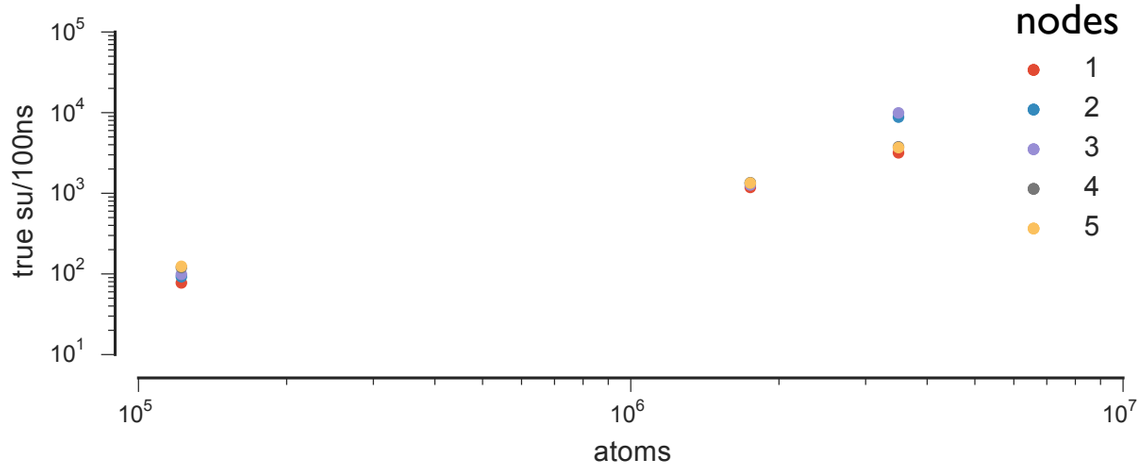
(a) Performance for varying node numbers.



(b) Performance for varying particle numbers



(c) SU cost to run a 100 ns MD simulation for varying node numbers



(d) SU cost to run a 100 ns MD simulation for varying particle numbers

- 16 cores per node, one NVIDIA K20 GPU per node