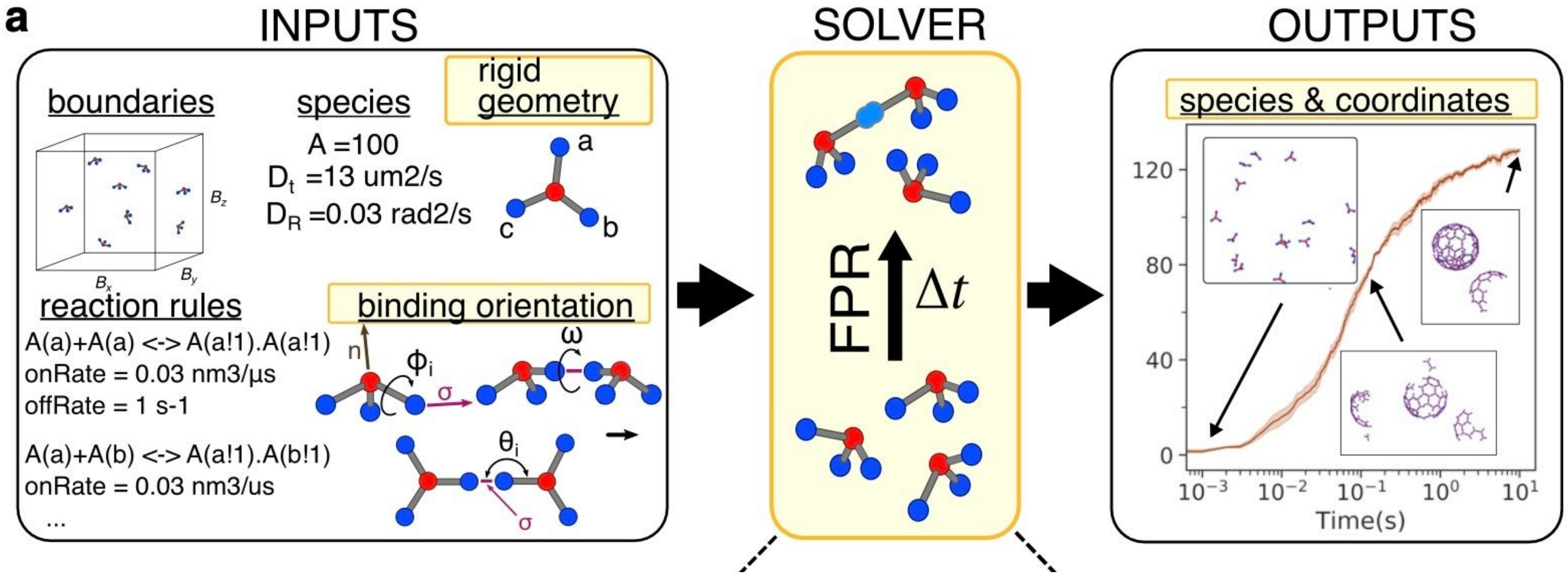
Simulating Macromolecular Self-assembly for Cell Biology: Hands-on Practice with NERDSS

Margaret Johnson and Samuel Foley

December 7
Biophysical Modeling of the Cell, ASCB 2025

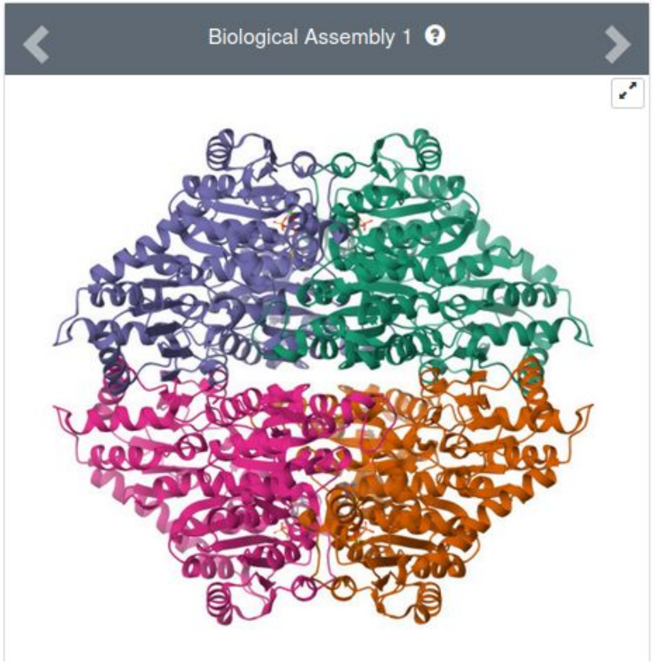
https://johnsonbiophysicslab.github.io/NERDSS/

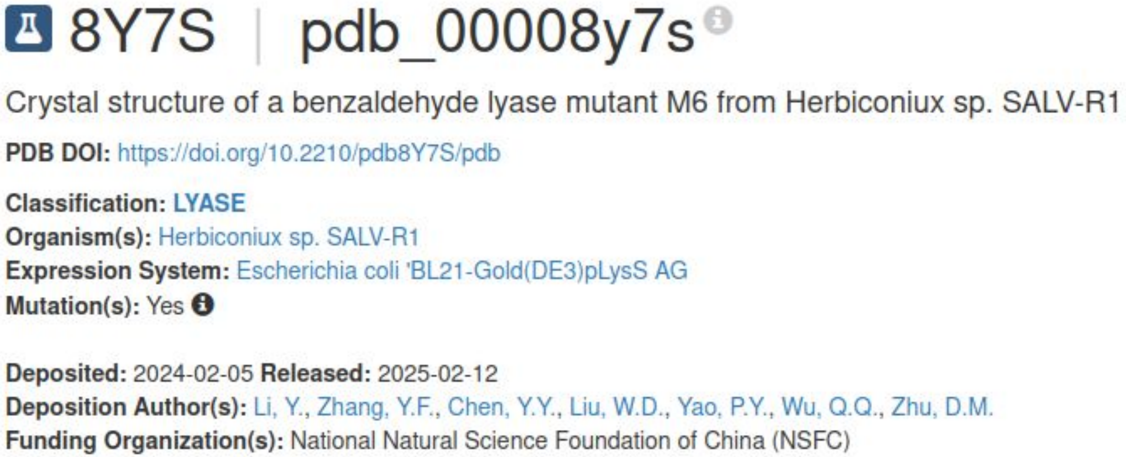
a



```
A =100
Dt =13 um2/s
DR =0.03 rad2/s

A(a)+A(a) <-> A(a!1).A(a!1)
onRate = 0.03 nm3/μs
offRate = 1 s-1

A(a)+A(b) <-> A(a!1).A(b!1)
onRate = 0.03 nm3/us
...
```

https://www.rcsb.org/structure/8Y7S





8Y7S | pdb_00008y7s

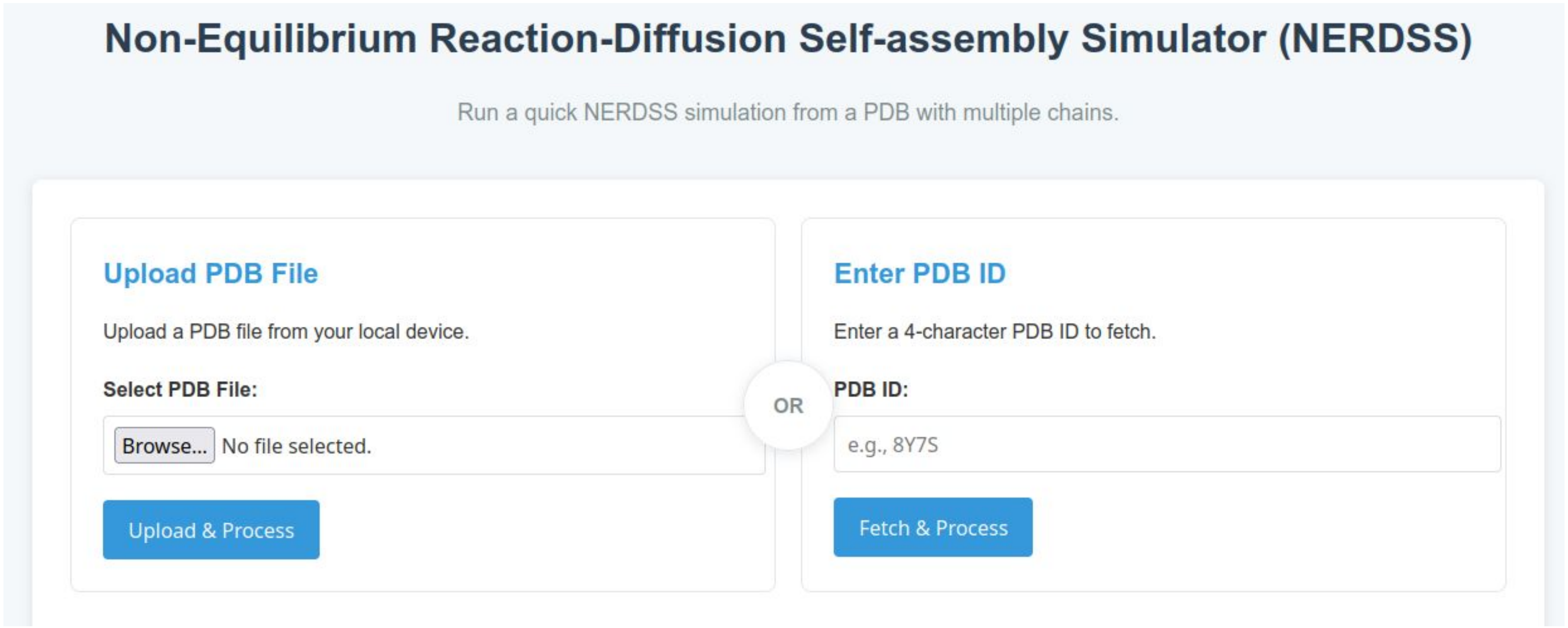
Crystal structure of a benzaldehyde lyase mutant M6 from Herbiconiux sp. SALV-R1

PDB DOI: https://doi.org/10.2210/pdb8Y7S/pdb

Classification: LYASE
Organism(s): Herbiconiux sp. SALV-R1
Expression System: Escherichia coli 'BL21-Gold(DE3)pLysS AG
Mutation(s): Yes

Deposited: 2024-02-05 **Released:** 2025-02-12
Deposition Author(s): Li, Y., Zhang, Y.F., Chen, Y.Y., Liu, W.D., Yao, P.Y., Wu, Q.Q., Zhu, D.M.
Funding Organization(s): National Natural Science Foundation of China (NSFC)

https://nerdssdemo.org/



Non-Equilibrium Reaction-Diffusion Self-assembly Simulator (NERDSS)

Run a quick NERDSS simulation from a PDB with multiple chains.

Upload PDB File

Upload a PDB file from your local device.

Select PDB File:

OR

Enter PDB ID

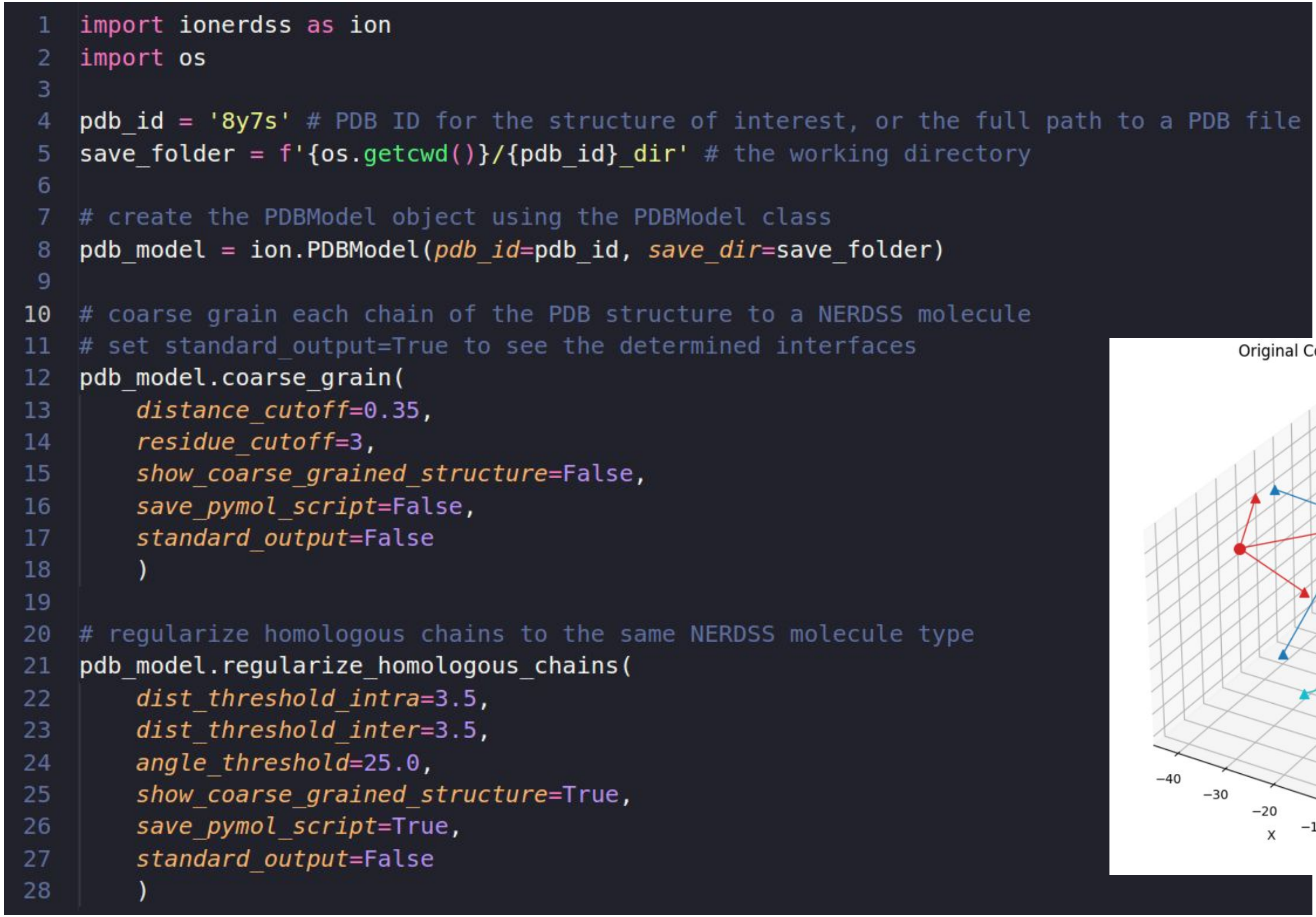
Enter a 4-character PDB ID to fetch.

PDB ID:

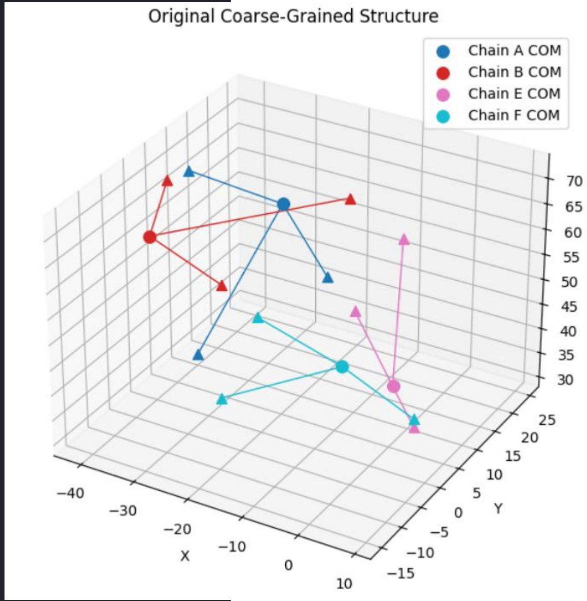
https://github.com/PhysFoley/Cell-Bio-2025-NERDSS-Demo

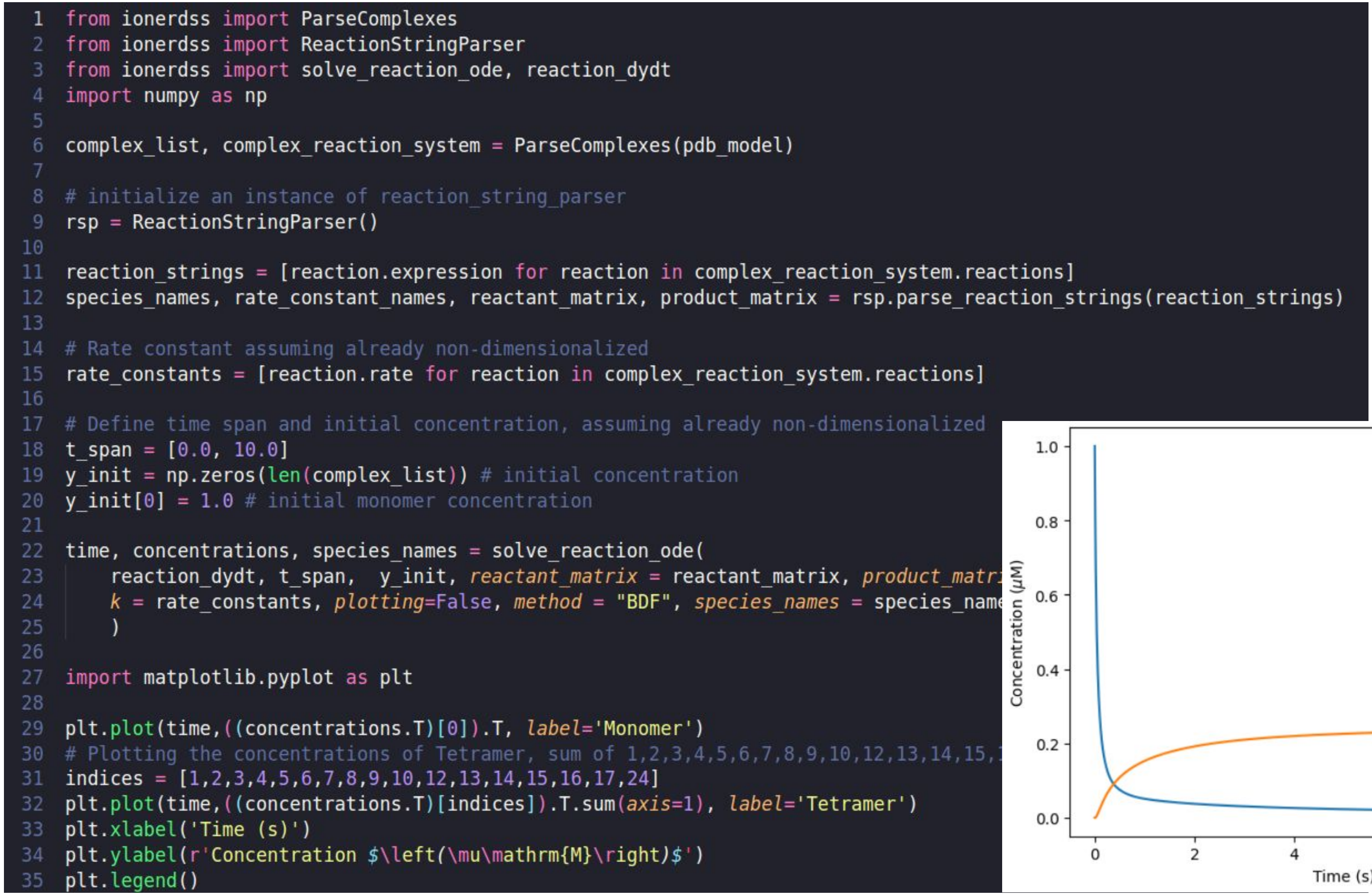
```
conda create -n nerdss numpy scipy matplotlib
conda activate nerdss
pip install ionerdss
```

ascb_demo.ipynb

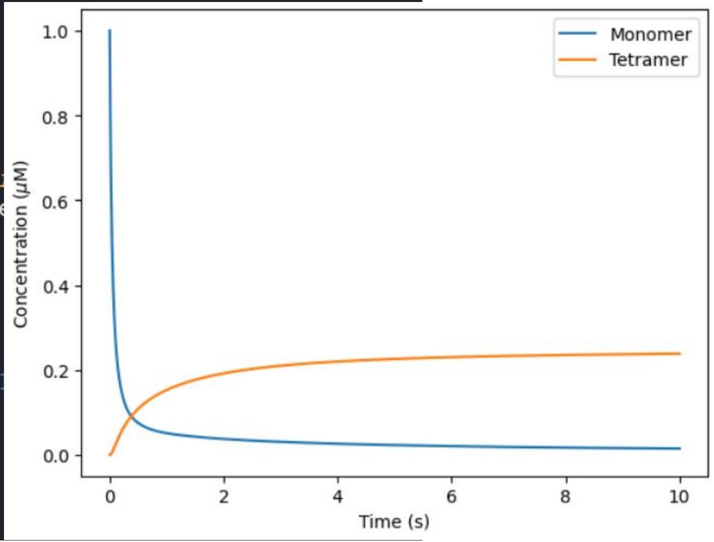


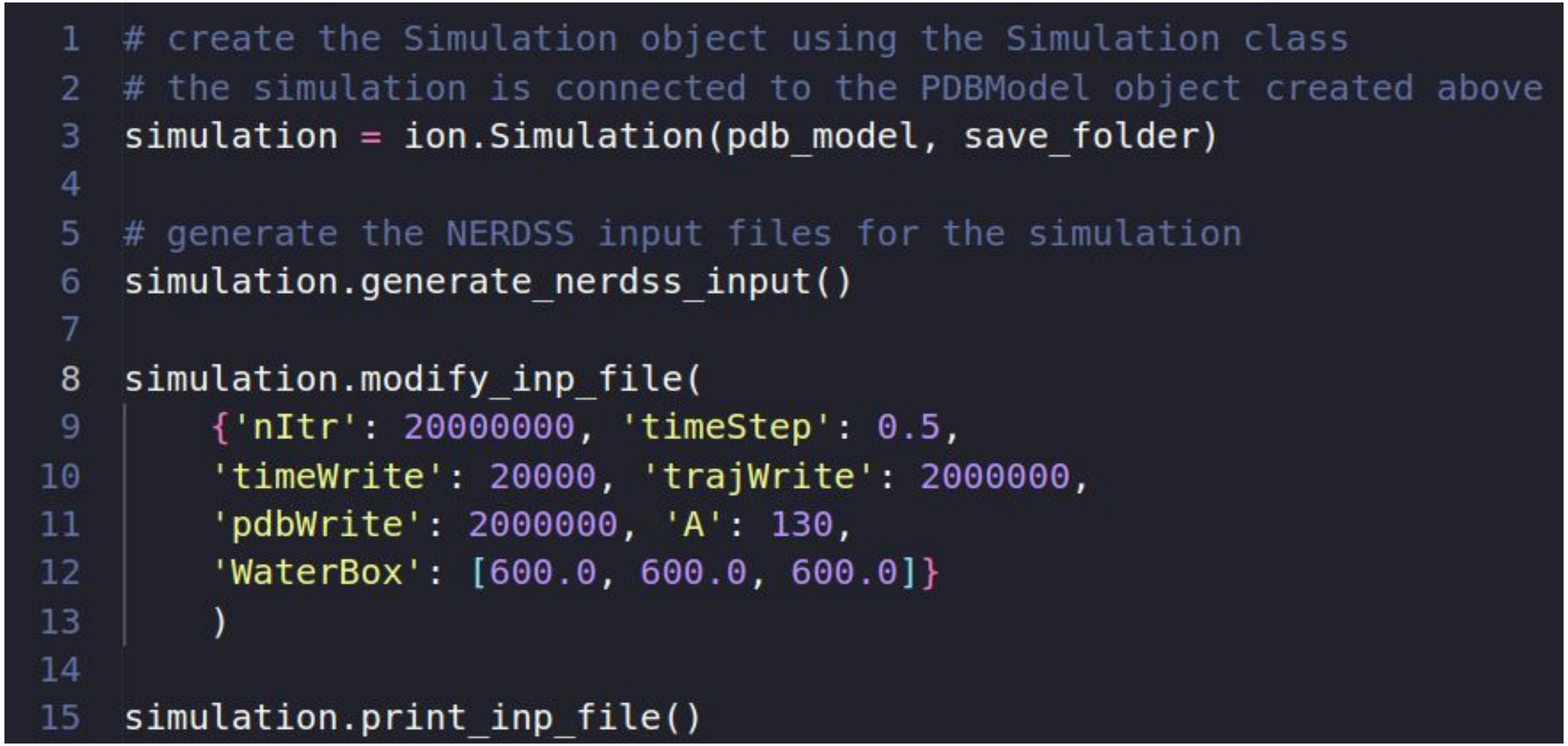
```python
import ionerdss as ion
import os

pdb_id = '8y7s' # PDB ID for the structure of interest, or the full path to a PDB file
save_folder = f'{os.getcwd()}/{pdb_id}_dir' # the working directory

# create the PDBModel object using the PDBModel class
pdb_model = ion.PDBModel(pdb_id=pdb_id, save_dir=save_folder)

# coarse grain each chain of the PDB structure to a NERDSS molecule
# set standard_output=True to see the determined interfaces
pdb_model.coarse_grain(
    distance_cutoff=0.35,
    residue_cutoff=3,
    show_coarse_grained_structure=False,
    save_pymol_script=False,
    standard_output=False
    )

# regularize homologous chains to the same NERDSS molecule type
pdb_model.regularize_homologous_chains(
    dist_threshold_intra=3.5,
    dist_threshold_inter=3.5,
    angle_threshold=25.0,
    show_coarse_grained_structure=True,
    save_pymol_script=True,
    standard_output=False
    )
```



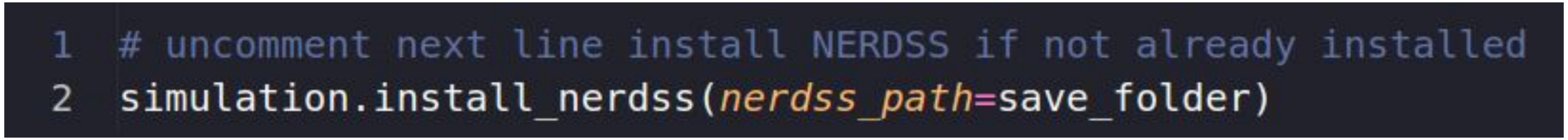


```python
from ionerdss import ParseComplexes
from ionerdss import ReactionStringParser
from ionerdss import solve_reaction_ode, reaction_dydt
import numpy as np

complex_list, complex_reaction_system = ParseComplexes(pdb_model)

# initialize an instance of reaction_string_parser
rsp = ReactionStringParser()

reaction_strings = [reaction.expression for reaction in complex_reaction_system.reactions]
species_names, rate_constant_names, reactant_matrix, product_matrix = rsp.parse_reaction_strings(reaction_strings)

# Rate constant assuming already non-dimensionalized
rate_constants = [reaction.rate for reaction in complex_reaction_system.reactions]

# Define time span and initial concentration, assuming already non-dimensionalized
t_span = [0.0, 10.0]
y_init = np.zeros(len(complex_list)) # initial concentration
y_init[0] = 1.0 # initial monomer concentration

time, concentrations, species_names = solve_reaction_ode(
    reaction_dydt, t_span,  y_init, reactant_matrix = reactant_matrix, product_matri
    k = rate_constants, plotting=False, method = "BDF", species_names = species_name
    )

import matplotlib.pyplot as plt

plt.plot(time,((concentrations.T)[0]).T, label='Monomer')
# Plotting the concentrations of Tetramer, sum of 1,2,3,4,5,6,7,8,9,10,12,13,14,15,1
indices = [1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,24]
plt.plot(time,((concentrations.T)[indices]).T.sum(axis=1), label='Tetramer')
plt.xlabel('Time (s)')
plt.ylabel(r'Concentration $\left(\mu\mathrm{M}\right)$')
plt.legend()
```



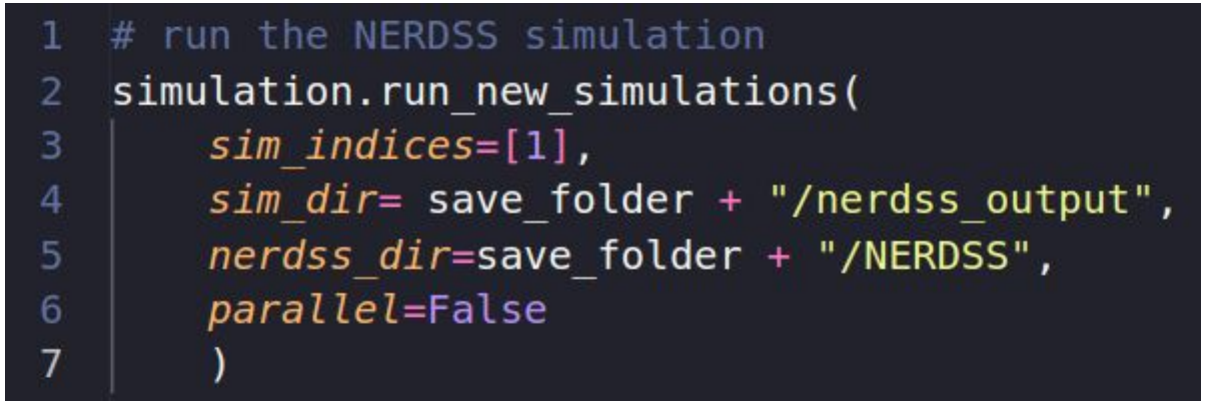


```python
# create the Simulation object using the Simulation class
# the simulation is connected to the PDBModel object created above
simulation = ion.Simulation(pdb_model, save_folder)

# generate the NERDSS input files for the simulation
simulation.generate_nerdss_input()

simulation.modify_inp_file(
    {'nItr': 20000000, 'timeStep': 0.5,
    'timeWrite': 20000, 'trajWrite': 2000000,
    'pdbWrite': 2000000, 'A': 130,
    'WaterBox': [600.0, 600.0, 600.0]}
    )

simulation.print_inp_file()
```

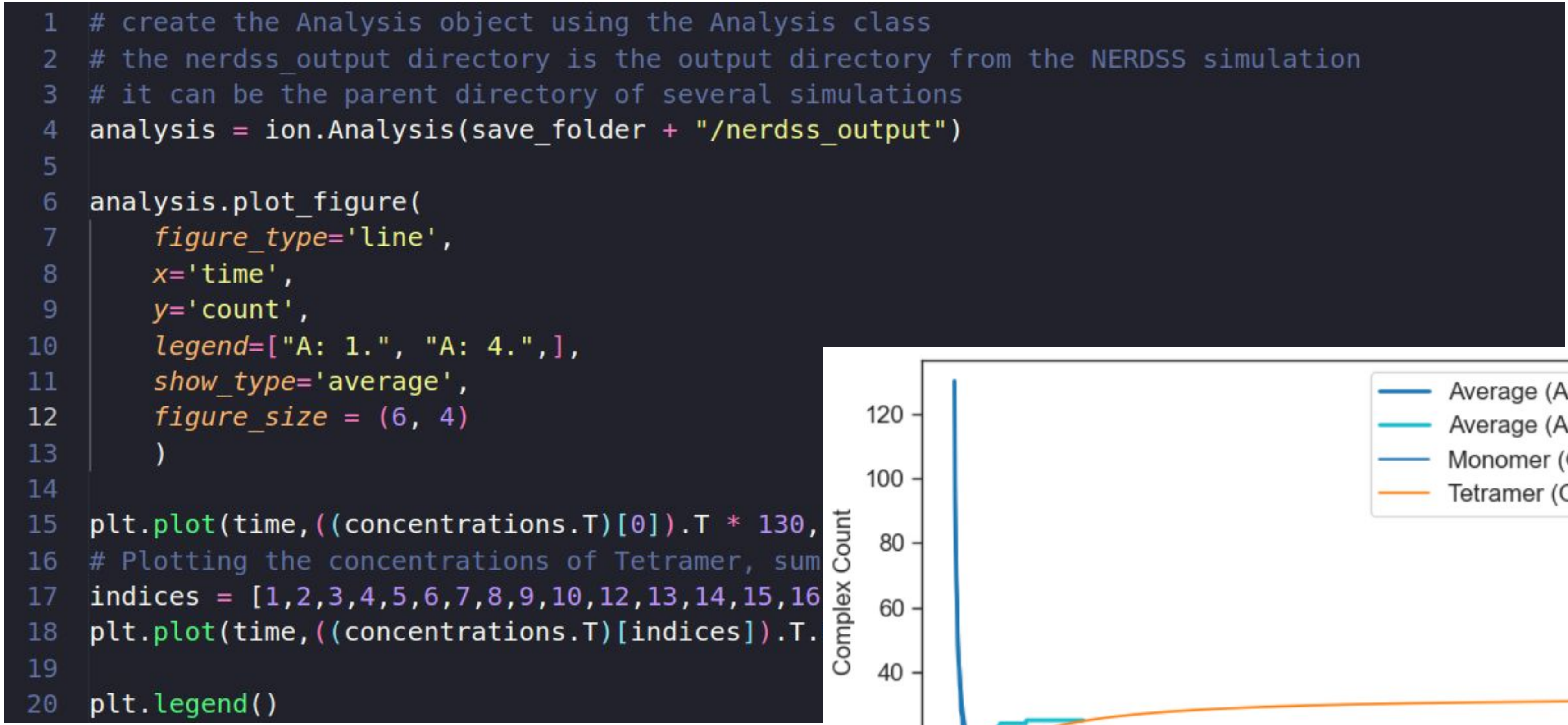


```python
# uncomment next line install NERDSS if not already installed
simulation.install_nerdss(nerdss_path=save_folder)
```

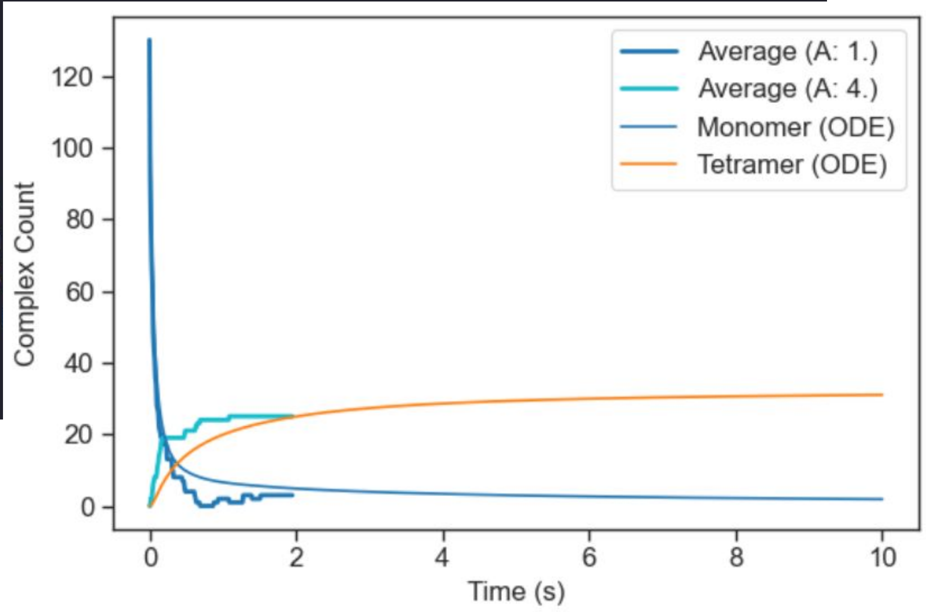
This will take a while…



```python
# run the NERDSS simulation
simulation.run_new_simulations(
    sim_indices=[1],
    sim_dir= save_folder + "/nerdss_output",
    nerdss_dir=save_folder + "/NERDSS",
    parallel=False
    )
```

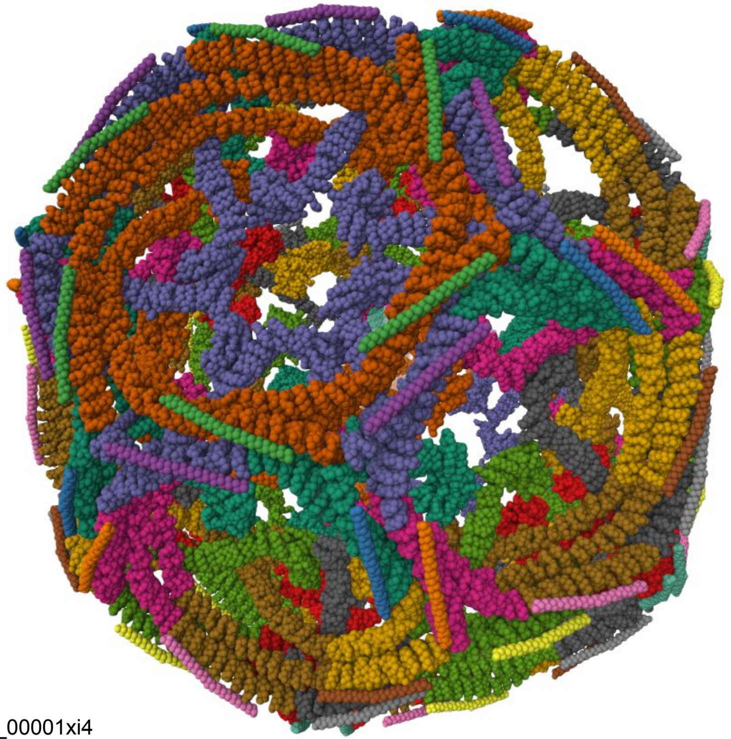


```
# create the Analysis object using the Analysis class
# the nerdss_output directory is the output directory from the NERDSS simulation
# it can be the parent directory of several simulations
analysis = ion.Analysis(save_folder + "/nerdss_output")

analysis.plot_figure(
    figure_type='line',
    x='time',
    y='count',
    legend=["A: 1.", "A: 4.",],
    show_type='average',
    figure_size = (6, 4)
    )

plt.plot(time,((concentrations.T)[0]).T * 130,
# Plotting the concentrations of Tetramer, sum
indices = [1,2,3,4,5,6,7,8,9,10,12,13,14,15,16
plt.plot(time,((concentrations.T)[indices]).T.

plt.legend()
```



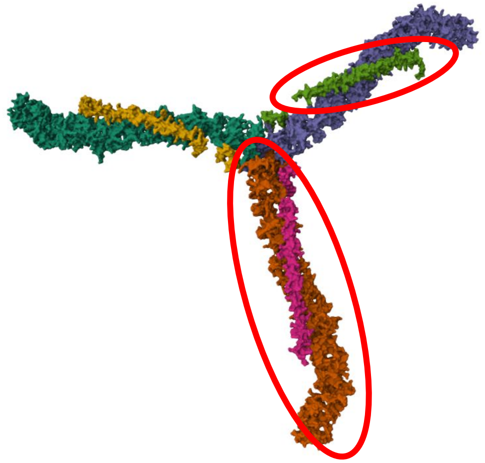
Setting Up and Simulating a Custom Model: Receptor-Coupled Self-Assembly

Downside to Auto-Generating Models from PDB Structures



pdb_00001xi4

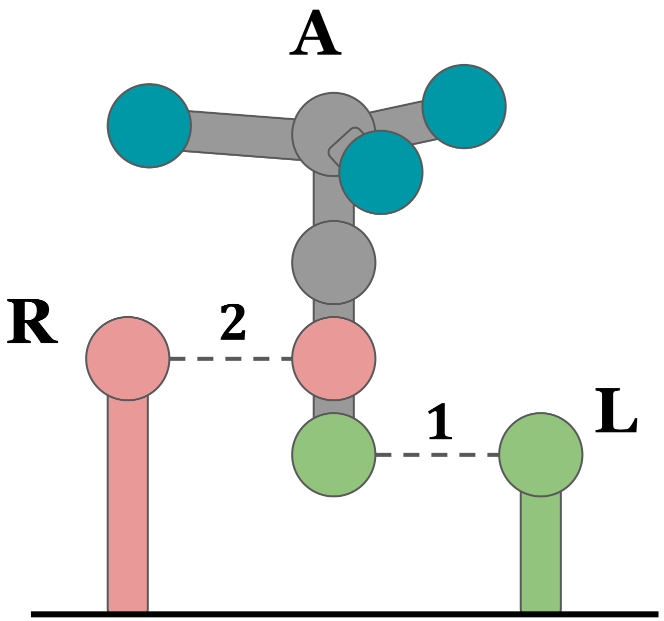
Fotin, A. *et al. Nature* 432, 573–579 (2004). doi:10.1038/nature03079

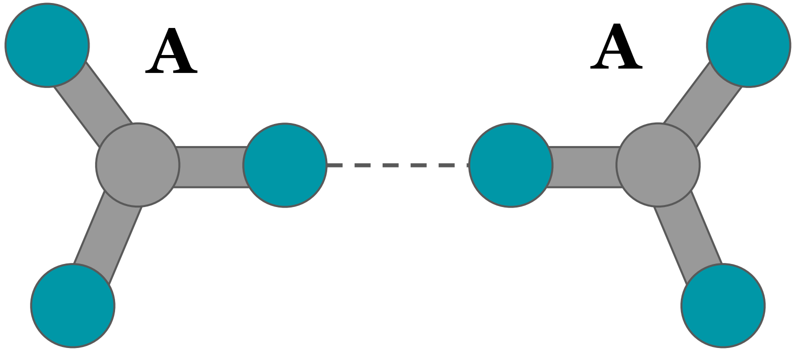


pdb_00003lvh

Wilbur, J. D. *et al. Dev. Cell* 18(5), 854-861 (2010). doi: 10.1016/j.devcel.2010.04.007

Simplified Coarse-Grained Self-Assembly Model

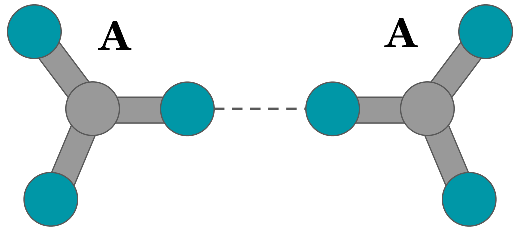


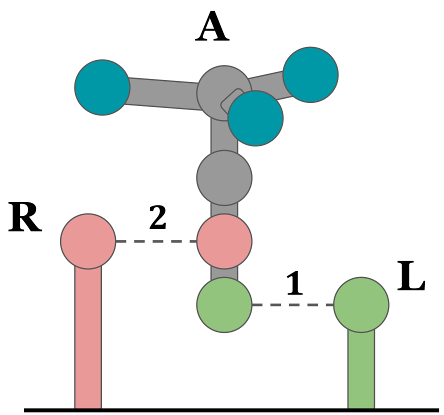


Simplified Coarse-Grained Self-Assembly Model

Custom CG model workflow:

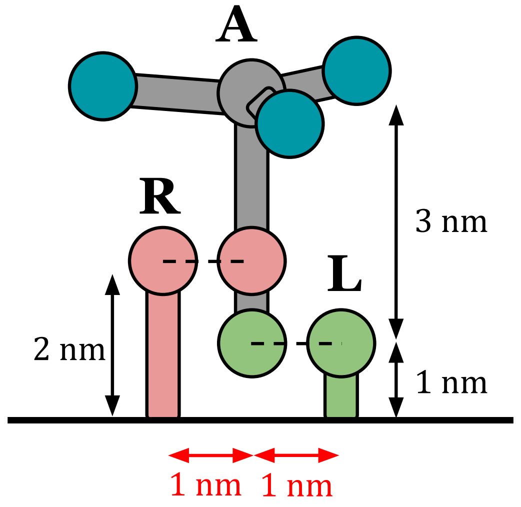
1. Enumerate model subunits and diffusion constants
2. Determine appropriate subunit geometries
3. Create molecule input files
4. Determine bond angles and parameters
5. Construct NERDSS parameter input file
6. Run simulation with NERDSS

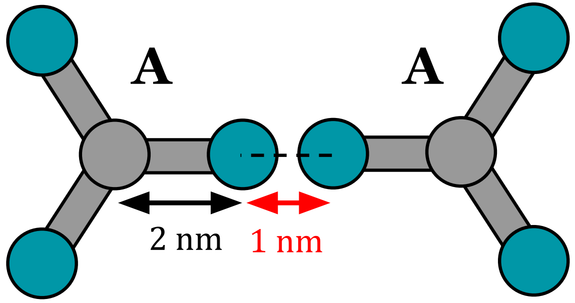




Simplified Coarse-Grained Self-Assembly Model

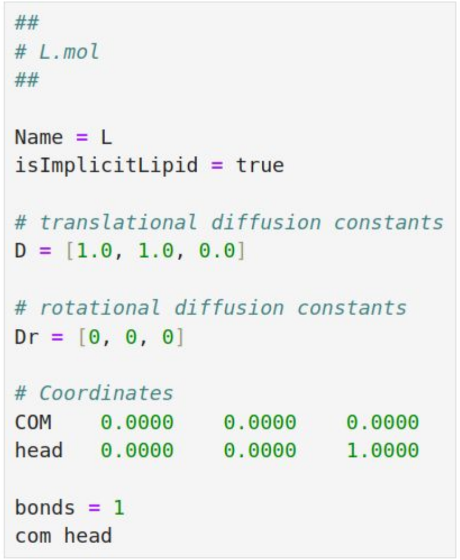
1. Enumerate model subunits and diffusion constants
   a. $D_A = 25\ \mu m^2/s$, $D_L = D_R = 1\ \mu m^2/s$
2. Determine appropriate subunit geometries:



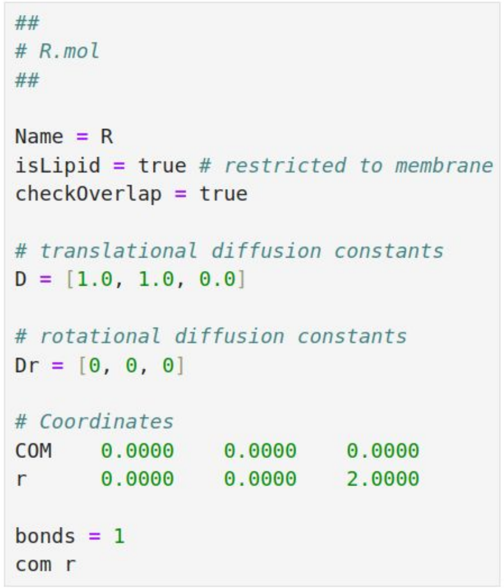


Simplified Coarse-Grained Self-Assembly Model

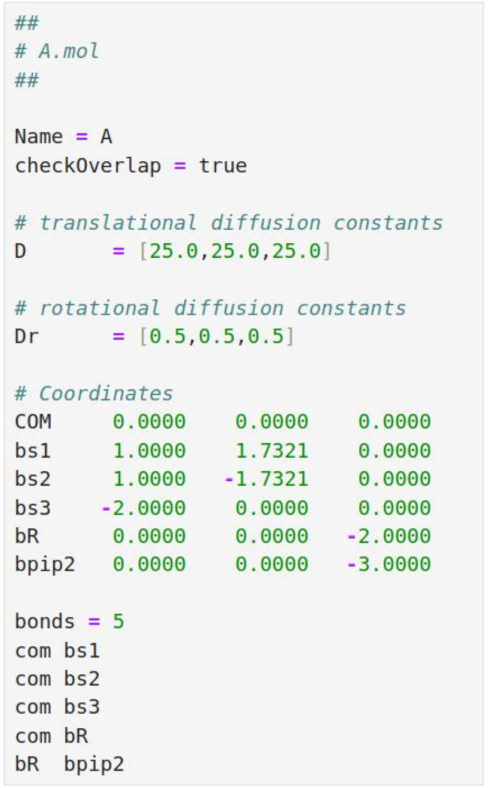
3. Create molecule input files



```
##
# L.mol
##

Name = L
isImplicitLipid = true

# translational diffusion constants
D = [1.0, 1.0, 0.0]

# rotational diffusion constants
Dr = [0, 0, 0]

# Coordinates
COM     0.0000     0.0000     0.0000
head    0.0000     0.0000     1.0000

bonds = 1
com head
```



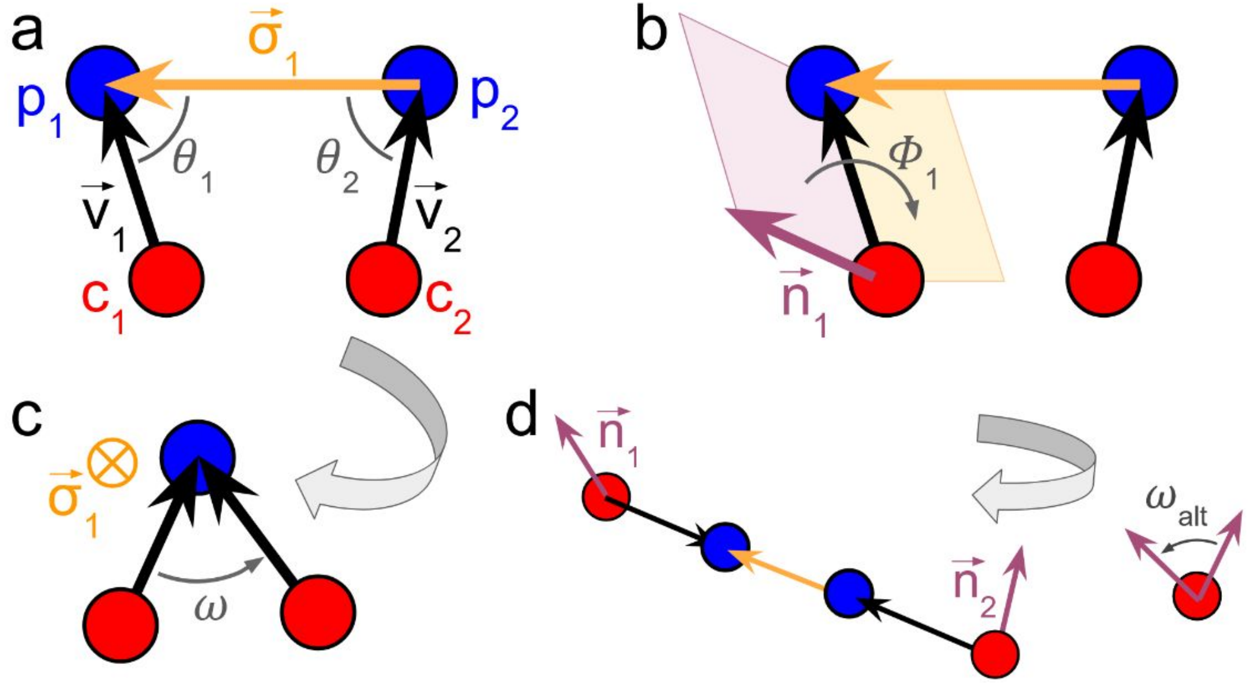
```
##
# R.mol
##

Name = R
isLipid = true # restricted to membrane
checkOverlap = true

# translational diffusion constants
D = [1.0, 1.0, 0.0]

# rotational diffusion constants
Dr = [0, 0, 0]

# Coordinates
COM     0.0000     0.0000     0.0000
r       0.0000     0.0000     2.0000

bonds = 1
com r
```



```
##
# A.mol
##

Name = A
checkOverlap = true

# translational diffusion constants
D        = [25.0,25.0,25.0]

# rotational diffusion constants
Dr       = [0.5,0.5,0.5]

# Coordinates
COM      0.0000     0.0000     0.0000
bs1      1.0000     1.7321     0.0000
bs2      1.0000    -1.7321     0.0000
bs3     -2.0000     0.0000     0.0000
bR       0.0000     0.0000    -2.0000
bpip2    0.0000     0.0000    -3.0000

bonds = 5
com bs1
com bs2
com bs3
com bR
bR  bpip2
```

Simplified Coarse-Grained Self-Assembly Model

4. Determine bond angles and parameters (k_{on}, k_{off}, σ, h)



Simplified Coarse-Grained Self-Assembly Model

5. Construct NERDSS parameter input file (`params.inp`)

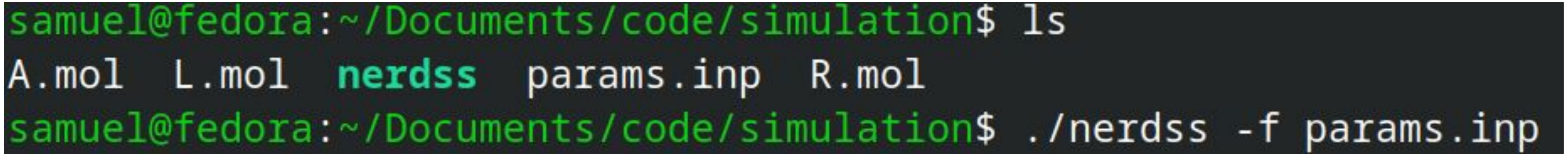
```
start parameters
    nItr = 360000000
    timeStep = 1 # us
    # See NERDSS user guide for more optional parameters
end parameters

start boundaries
    WaterBox = [1000,1000,1000] # nm
    xBCtype = reflect
    yBCtype = reflect
    zBCtype = reflect
end boundaries

start molecules
    L : 18000
    A : 120
    R : 360
end molecules
                          ⋮
```

```
                          ⋮
start reactions
    #### A - L ####
    A(bpip2) + L(head) <-> A(bpip2!1).L(head!1)
    onRate3DMacro = 0.3 # (uM)^-1 s^-1
    offRateMacro = 30.0 # s^-1
    sigma = 1.0 # nm
    norm1 = [1,0,0]
    norm2 = [0,0,1]
    assocAngles = [1.5708, 1.5708, M_PI, nan, M_PI]

    #### A - R ####
    A(bR,bpip2!*) + R(r) <-> A(bR!1,bpip2!*).R(r!1)
    onRate3DMacro = 1
    offRateMacro = 10
    sigma = 1.0
    norm1 = [1,0,0]
    norm2 = [0,0,1]
    assocAngles = [1.5708, 1.5708, 1.0472, nan, M_PI]
                          ⋮
```

```
                          ⋮
    #### A - A ####
    A(bs1) + A(bs2) <-> A(bs1!1).A(bs2!1)
    onRate3DMacro = 0.04 # (uM)^-1 s^-1
    offRateMacro = 10 # s^-1
    norm1 = [0,0,1]
    norm2 = [0,0,1]
    sigma = 1.0
    length3Dto2D = 10 # nm
    assocAngles = [M_PI,M_PI,nan,nan,0]

    # bs2+bs3 and bs1+bs3 reactions omitted for space

    A(bs1) + A(bs1) <-> A(bs1!1).A(bs1!1)
    onRate3DMacro = 0.02 # (uM)^-1 s^-1
    offRateMacro = 10 # s^-1
    norm1 = [0,0,1]
    norm2 = [0,0,1]
    sigma = 1.0
    length3Dto2D = 10
    assocAngles = [M_PI,M_PI,nan,nan,0]

    # bs2+bs2 and bs3+bs3 reactions omitted for space
end reactions
```

Simplified Coarse-Grained Self-Assembly Model

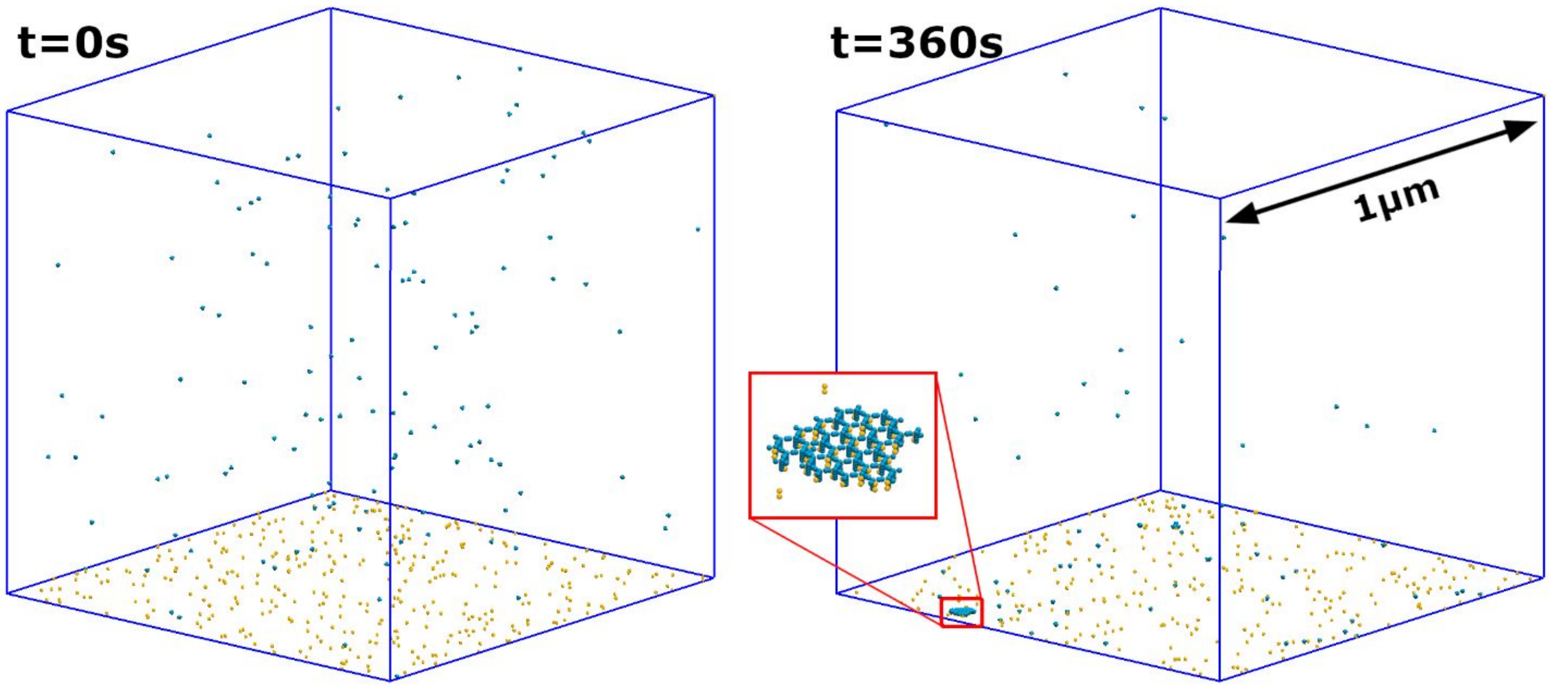
6. Run simulation with NERDSS



```
samuel@fedora:~/Documents/code/simulation$ ls
A.mol  L.mol  nerdss  params.inp  R.mol
samuel@fedora:~/Documents/code/simulation$ ./nerdss -f params.inp
```

Simplified Coarse-Grained Self-Assembly Model

6. Run simulation with NERDSS



Thank you for your attention!

Membrane-Associated Self-Assembly for Cellular Decision-Making

Foley, S. L. & Johnson, M. E., 10.48550/arXiv.2505.17290