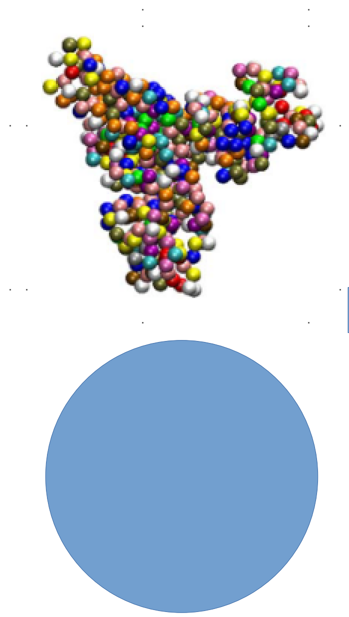
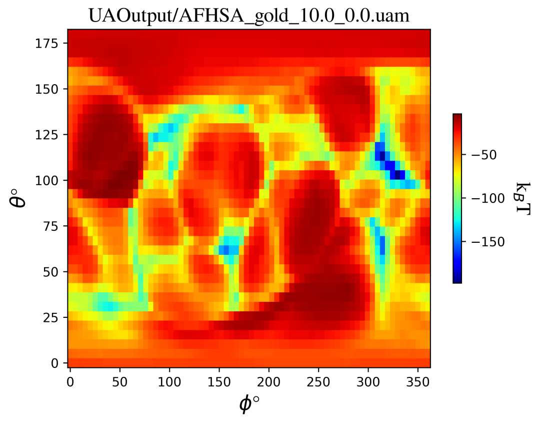
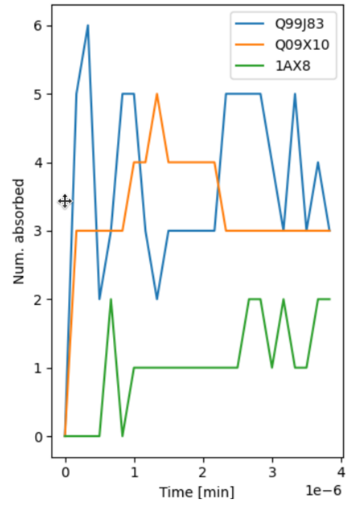
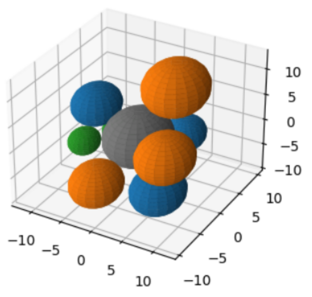
UnitedAtom/Corona KMC guide









Ian Rouse, 01/12/22

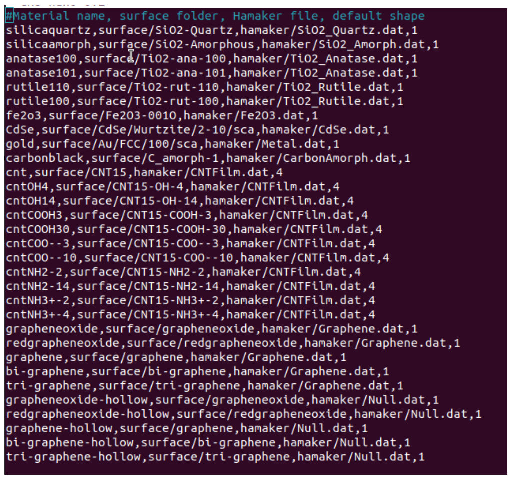
Installation

- Download the code from the bitbucket repository:
  - git clone https://softmattergroup@bitbucket.org/softmattergroup/unitedatom.git
- Compile UnitedAtom
  - make clean; make
  - UnitedAtom needs a C++ compiler and Boost to be installed
  - You might need to edit the make file if Boost is installed in a weird place to add -L/usr/local/lib/ or wherever your libraries are kept.
  - If you have any older version of boost check the readme for help on debugging the changes in file copying.
- Install any missing Python libraries – numpy, scipy, matplotlib, argparse should cover it.
-
- Before use make sure to run “git pull” to get the latest code and “make clean; make” to update UA
- Running “python3 PrepareKMCInput.py” runs a simple test case to make sure everything is working.

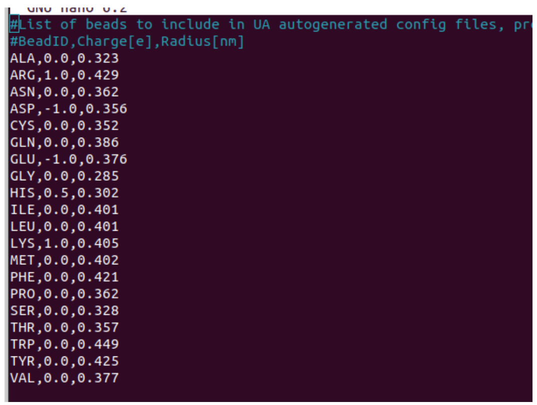
UnitedAtom package contents

- Source code (src folder) for building UnitedAtom
- Hamaker files (hamaker folder) for various materials
- Surface PMF files (surface folder) for various materials/phases/functionalisations from metadynamics
- Predicted PMF files (surface_pmfp) for even more materials based on machine learning
- Python scripts for running corona predictions (CoronaKMC-P3.py) and generating input for these (PrepareKMCInput.py, BuildCoronaParams-P3.py).
- A python script for quick-running UA (RunUA.py)
- Predefined materials MaterialSet.csv for common Hamaker +PMF pairs.
- Bead definition files (beadset folder)
- Miscalleneous tools (tools folder) for output processing
- NP storage folder (nps folder and subfolders)
- Protein storage folder (pdb and subfolders)

Corona prediction for simple NPs



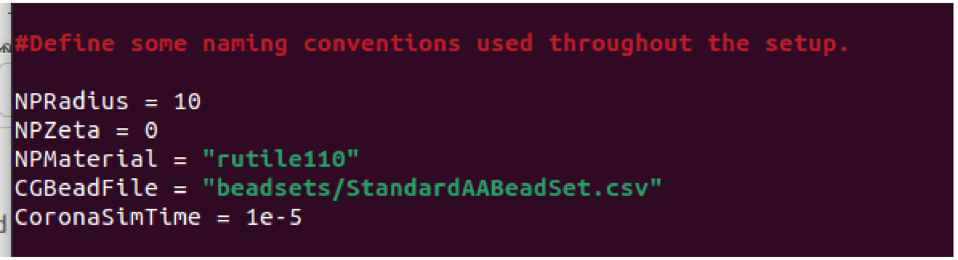
```
#Material name, surface folder, Hamaker file, default shape
silicaquartz,surface/SiO2-Quartz,hamaker/SiO2_Quartz.dat,1
silicaamorph,surface/SiO2-Amorphous,hamaker/SiO2_Amorph.dat,1
anatase100,surface/TiO2-ana-100,hamaker/TiO2_Anatase.dat,1
anatase101,surface/TiO2-ana-101,hamaker/TiO2_Anatase.dat,1
rutile110,surface/TiO2-rut-110,hamaker/TiO2_Rutile.dat,1
rutile100,surface/TiO2-rut-100,hamaker/TiO2_Rutile.dat,1
fe2o3,surface/Fe2O3-0010,hamaker/Fe2O3.dat,1
CdSe,surface/CdSe/Wurtzite/2-10/sca,hamaker/CdSe.dat,1
gold,surface/Au/FCC/100/sca,hamaker/Metal.dat,1
carbonblack,surface/C_amorph-1,hamaker/CarbonAmorph.dat,1
cnt,surface/CNT15,hamaker/CNTFilm.dat,4
cntOH4,surface/CNT15-OH-4,hamaker/CNTFilm.dat,4
cntOH14,surface/CNT15-OH-14,hamaker/CNTFilm.dat,4
cntCOOH3,surface/CNT15-COOH-3,hamaker/CNTFilm.dat,4
cntCOOH30,surface/CNT15-COOH-30,hamaker/CNTFilm.dat,4
cntCOO--3,surface/CNT15-COO--3,hamaker/CNTFilm.dat,4
cntCOO--10,surface/CNT15-COO--10,hamaker/CNTFilm.dat,4
cntNH2-2,surface/CNT15-NH2-2,hamaker/CNTFilm.dat,4
cntNH2-14,surface/CNT15-NH2-14,hamaker/CNTFilm.dat,4
cntNH3+-2,surface/CNT15-NH3+-2,hamaker/CNTFilm.dat,4
cntNH3+-4,surface/CNT15-NH3+-4,hamaker/CNTFilm.dat,4
grapheneoxide,surface/grapheneoxide,hamaker/Graphene.dat,1
redgrapheneoxide,surface/redgrapheneoxide,hamaker/Graphene.dat,1
graphene,surface/graphene,hamaker/Graphene.dat,1
bi-graphene,surface/bi-graphene,hamaker/Graphene.dat,1
tri-graphene,surface/tri-graphene,hamaker/Graphene.dat,1
grapheneoxide-hollow,surface/grapheneoxide,hamaker/Null.dat,1
redgrapheneoxide-hollow,surface/redgrapheneoxide,hamaker/Null.dat,1
graphene-hollow,surface/graphene,hamaker/Null.dat,1
bi-graphene-hollow,surface/bi-graphene,hamaker/Null.dat,1
tri-graphene-hollow,surface/tri-graphene,hamaker/Null.dat,1
```



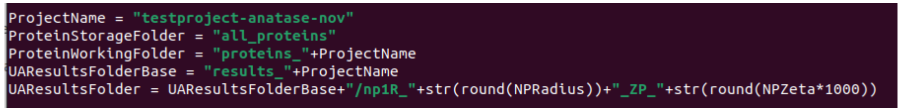
```
#List of beads to include in UA autogenerated config files, pr
#BeadID,Charge[e],Radius[nm]
ALA,0.0,0.323
ARG,1.0,0.429
ASN,0.0,0.362
ASP,-1.0,0.356
CYS,0.0,0.352
GLN,0.0,0.386
GLU,-1.0,0.376
GLY,0.0,0.285
HIS,0.5,0.302
ILE,0.0,0.401
LEU,0.0,0.401
LYS,1.0,0.405
MET,0.0,0.402
PHE,0.0,0.421
PRO,0.0,0.362
SER,0.0,0.328
THR,0.0,0.357
TRP,0.0,0.449
TYR,0.0,0.425
VAL,0.0,0.377
```

- Make sure your material is in MaterialSet.csv and all the CG beads you want to use are defined in a file in the beadsets folder
- MaterialSet.csv defines a material with a name, a surface folder, a Hamaker file and the default shape (1=sphere, 2=cylinder with planar PMF, 3= cube, 4= SWCNT, 5=MWCNT)
- The beadset files can be used to switch between different naming/radius/charge conventions so UA doesn't get confused if you want to do calculations for PMF sets without certain beads. It's suggested to copy the default file before making changes to stop git complaining when you update.

Corona prediction for simple NPs



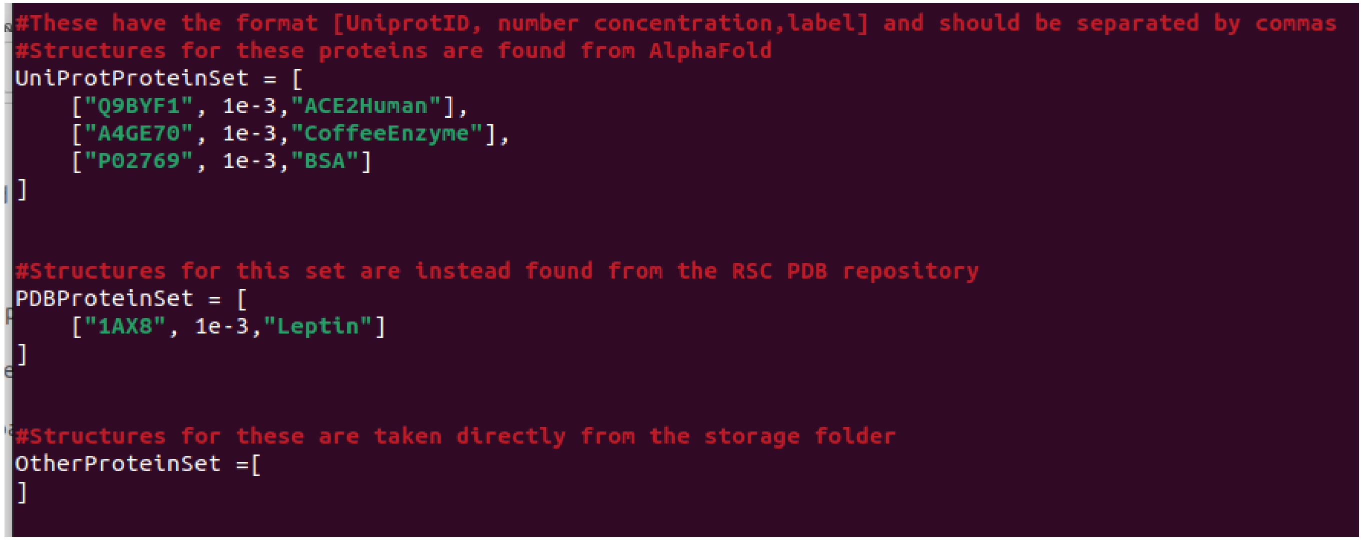
```
#Define some naming conventions used throughout the setup.

NPRadius = 10
NPZeta = 0
NPMaterial = "rutile110"
CGBeadFile = "beadsets/StandardAABeadSet.csv"
CoronaSimTime = 1e-5
```



```
ProjectName = "testproject-anatase-nov"
ProteinStorageFolder = "all_proteins"
ProteinWorkingFolder = "proteins_"+ProjectName
UAResultsFolderBase = "results_"+ProjectName
UAResultsFolder = UAResultsFolderBase+"/np1R_"+str(round(NPRadius))+"_ZP_"+str(round(NPZeta*1000))
```

- The script PrepareKMCInput.py is designed to automate most of the corona prediction.
- The first user input to set is setting the radius and zeta (actually surface) potential for the NP, choosing a material (from MaterialSet.csv), set of CG beads (from a file in beadsets), and defining the overall time for corona prediction.
- The project name needs to be set so it can locate specific files for a given NP and protein set. All these others can stay as their defaults.

Corona prediction: defining a protein set



```
#These have the format [UniprotID, number concentration,label] and should be separated by commas
#Structures for these proteins are found from AlphaFold
UniProtProteinSet = [
    ["Q9BYF1", 1e-3,"ACE2Human"],
    ["A4GE70", 1e-3,"CoffeeEnzyme"],
    ["P02769", 1e-3,"BSA"]
]


#Structures for this set are instead found from the RSC PDB repository
PDBProteinSet = [
    ["1AX8", 1e-3,"Leptin"]
]


#Structures for these are taken directly from the storage folder
OtherProteinSet =[
]
```

- Three classes defined by lists. Each list contains sets of three values: an ID code used to find the file, a number concentration for that protein/biomolecule, and a label.
- UniProtProteinSet loads from AlphaFold based on the UniProt ID unless the protein is already in storage/working
- PDBProteinSet loads from the PDB based on PDB ID unless the protein is already in storage/working
- OtherProteinSet just looks in the protein working directory and protein storage folder for something named ID.pdb
- If you don’t want any of a specific type, set the list to an empty list, e.g. UniProtProteinSet = []
- “Protein” can mean any biomolecule made of CG beads.

Corona prediction

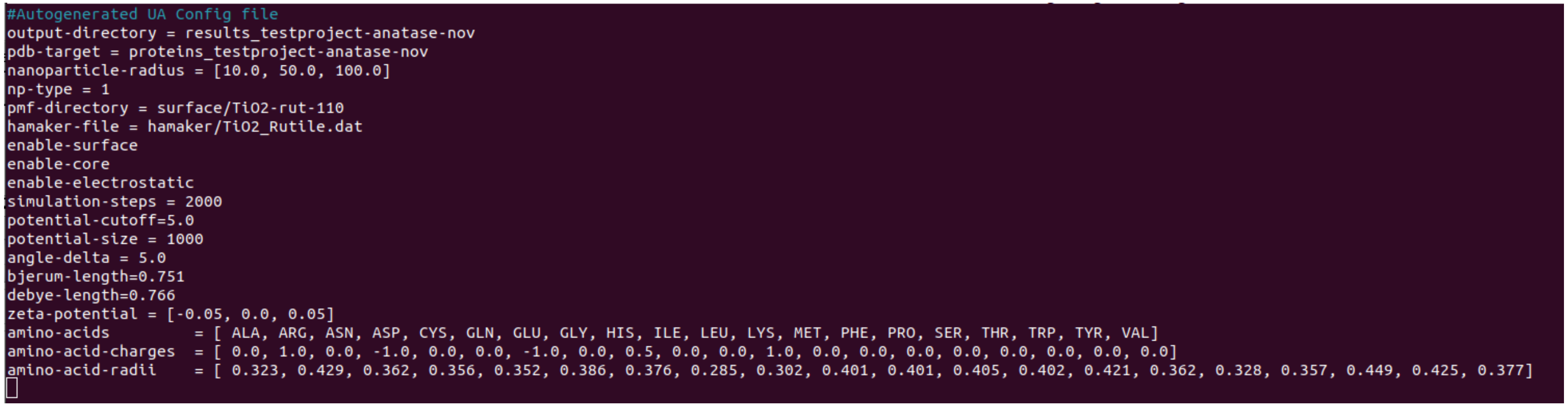
- Once PrepareKMCInput.py is set up, run “python3 PrepareKMCInput.py” and wait. It does the following:
- 1) Find protein structures, downloading if necessary.
- 2) Builds an appropriate UA config file and runs UA
  - RunUA.py
- 3) Converts UA output to KMC input
  - BuildCoronaParams-P3.py
- 4) Runs CoronaKMC
  - Python3 CoronaKMC-P3.py

- All of the above get run with automatically specified arguments.

UnitedAtom – scanning NPs

- More often you’ll want to run UA for a range of materials, radii and zeta potentials.
- This is done with manually editing config files or generating NP files for input, then running “./UnitedAtom –config-file=myconfigfile.config”

UA config file – one component



```
#Autogenerated UA Config file
output-directory = results_testproject-anatase-nov
pdb-target = proteins_testproject-anatase-nov
nanoparticle-radius = [10.0, 50.0, 100.0]
np-type = 1
pmf-directory = surface/TiO2-rut-110
hamaker-file = hamaker/TiO2_Rutile.dat
enable-surface
enable-core
enable-electrostatic
simulation-steps = 2000
potential-cutoff=5.0
potential-size = 1000
angle-delta = 5.0
bjerum-length=0.751
debye-length=0.766
zeta-potential = [-0.05, 0.0, 0.05]
amino-acids         = [ ALA, ARG, ASN, ASP, CYS, GLN, GLU, GLY, HIS, ILE, LEU, LYS, MET, PHE, PRO, SER, THR, TRP, TYR, VAL]
amino-acid-charges  = [ 0.0, 1.0, 0.0, -1.0, 0.0, 0.0, -1.0, 0.0, 0.5, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
amino-acid-radii    = [ 0.323, 0.429, 0.362, 0.356, 0.352, 0.386, 0.376, 0.285, 0.302, 0.401, 0.401, 0.405, 0.402, 0.421, 0.362, 0.328, 0.357, 0.449, 0.425, 0.377]
```

- For single-component NPs, edit a configuration file to have the material of interest and set the range of radius (in nm) and zeta potentials (in V). Copy in whatever bead types you need. UA builds all radius/ZP pairs. pdb-target can be a folder or single PDB file

- Don’t forget to change the output folder name!

UA config file – multicomponent NPs

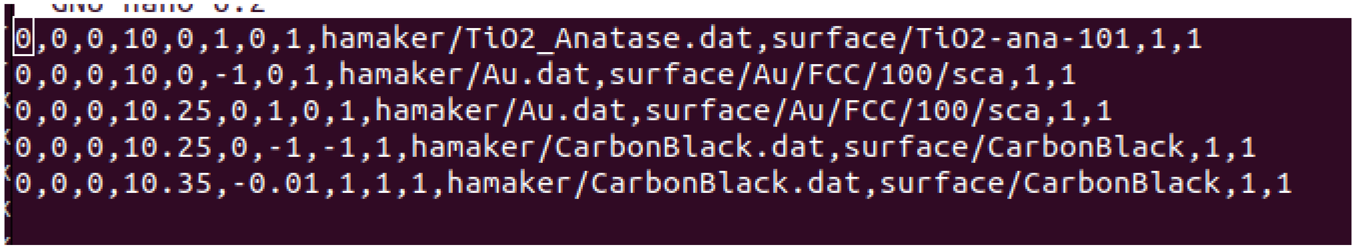
- For brushes, raspberry, core-shell NPs a set of NP files can be given as input by adding

```
np-target            = nps/somesubfolder
```

- This causes UA to ignore the radius/zeta/material settings (they still have to be present though) and read these from all np files in the subfolder.

NP file structure

- An NP file looks like this:



```
0,0,0,10,0,1,0,1,hamaker/TiO2_Anatase.dat,surface/TiO2-ana-101,1,1
0,0,0,10,0,-1,0,1,hamaker/Au.dat,surface/Au/FCC/100/sca,1,1
0,0,0,10.25,0,1,0,1,hamaker/Au.dat,surface/Au/FCC/100/sca,1,1
0,0,0,10.25,0,-1,-1,1,hamaker/CarbonBlack.dat,surface/CarbonBlack,1,1
0,0,0,10.35,-0.01,1,1,1,hamaker/CarbonBlack.dat,surface/CarbonBlack,1,1
```

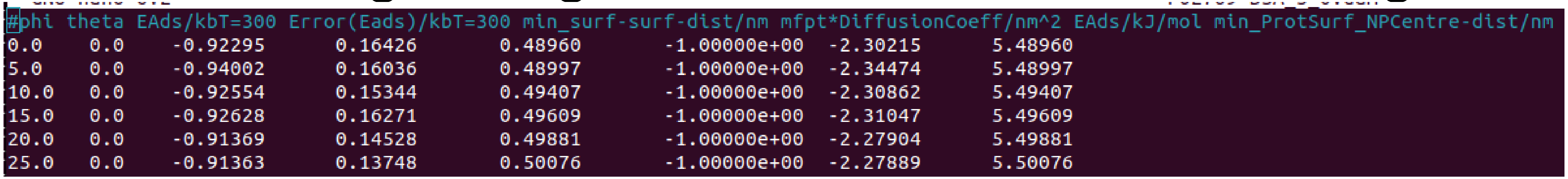
- Where each line in order gives x,y,z,radius,zeta,core scaling factor, surface scaling factor, shape (1=sphere), hamaker file, surface folder, PMF vdw cutoff (generally 1.2 or 1.0), correction type (usually 1).
- The surface and core scaling factors are multiplicative coefficients used to rescale PMF and Hamaker contributions respectively. Setting both to -1 generates an “anti NP” which can be used to cancel out part of a larger NP to generate a hollow shell. The above example uses these to make thin shells of gold and carbon around a solid anatase core.

Biomolecule input

- For historical reasons, the biomolecule input is called a protein and must be in .pdb file format.
- UA goes through the input PDB file line by line and keeps only lines starting with ATOM and with CA as the atom type.
- A three-letter tag is extracted for consistency with the PDB standard. Longer tags will get truncated to whatever three letters happen to be in the right place so don’t use them.
- The x/y/z co-ordinates are extracted, as is the occupancy (used as an overall scale for the potential of that bead to allow for multiple positions, averages over charge states, etc) and the bfactor (usually ignored except with certain config options enabled).

UA Output

- The direct UA output is a UA map (.uam) file containing energies for each orientation range.



```
#phi theta EAds/kbT=300 Error(Eads)/kbT=300 min_surf-surf-dist/nm mfpt*DiffusionCoeff/nm^2 EAds/kJ/mol min_ProtSurf_NPCentre-dist/nm
0.0    0.0    -0.92295      0.16426       0.48960       -1.00000e+00  -2.30215      5.48960
5.0    0.0    -0.94002      0.16036       0.48997       -1.00000e+00  -2.34474      5.48997
10.0   0.0    -0.92554      0.15344       0.49407       -1.00000e+00  -2.30862      5.49407
15.0   0.0    -0.92628      0.16271       0.49609       -1.00000e+00  -2.31047      5.49609
20.0   0.0    -0.91369      0.14528       0.49881       -1.00000e+00  -2.27904      5.49881
25.0   0.0    -0.91363      0.13748       0.50076       -1.00000e+00  -2.27889      5.50076
```

Phi, theta are the left hand edges, so the first line here is best thought of as phi = 2.5, theta = 2.5.
Eads/kbT and Error are the mean and standard deviation of energies in this interval.
min_surf-surf distance is the protein surface – NP surface separation at the energy minimum
MFPT is usually -1 and can be ignored.
Eads/kJ/mol is the adsorption energy in kJ mol to avoid confusion with different temperature outputs.
min_ProtSurf_NPCentre is the distance of the surface of the protein to the centre of the NP at the energy minimum. Usually, SSD + NP Radius for simple cases.

Generally plotting a heatmap is useful for visualisation (tools/plotmap) and these files are also used as input for CoronaKMC

UA Config options

- There are a lot of extra UA config options that might be needed!
- Check src/Config.h to see all current options
- Most useful are likely setting the np-shape to something different. 1 = sphere from planar PMF, 2 = cylinder from planar PMF, 3 = cube from planar PMF, 4 = SWCNT from 1.5nm diameter PMF, 5 = MWCNT from 1.5nm diameter PMF.
- The temperature and Debye length can be adjusted. Note that the PMFs may not be appropriate for different temperatures and salt concentrations.
- **The Bjerrum length usually does nothing.**
  - Unless you have recalculate-zp switched on, which you usually shouldn’t.
- Disorder strategies can be enabled for intrinsically disordered proteins. Disordered residues are recognised if their b-factor falls within certain limits (disorder-minbound and disorder-maxbound) specified in the config file and defaulting to AlphaFold definition (minbound = 0, maxbound = 50). disorder-strategy=0 is the default and treats this as normal. Strategy 1 moves these residues to the COM of the protein under the assumption they are freely moving. Strategy 2 switches them off under the assumption they move away to allow the core of the protein to bind. Generally, strategy 1 gives the strongest binding energy and strategy 2 the lowest, with the default inbetween. This is not always true, especially for a protein core surrounded by loops like AlphaFold tends to generate, in which case 1 and 2 produce more strongly binding than the default.

Non-automated corona prediction

- BuildCoronaParams-P3.py converts .uam files to a corona input file.
- This has a few arguments to make it work:
- parser.add_argument('-r','--radius',type=float,help="Radius of the NP",default=19)
- parser.add_argument('-z','--zeta',type=float,help="Zeta potential of the NP",default=0)
- parser.add_argument('-a','--average',type=float,help="average over orientations (does nothing right now)",default=0)
- parser.add_argument('-f','--folder',type=str,help="folder containing UA heatmaps",default="results_anatase_alltargets_sphere")
- parser.add_argument('-s','--shape',type=int,help="NP shape: 1 = sphere 2 = cylinder", default=1)
- parser.add_argument('-p','--proteins',type=str,help="protein definition file",default="ProteinConcs.csv")
- parser.add_argument('-c','--coordfolder',type=str,help="location of PDB files",default="pdbs/All")
- parser.add_argument('-v','--verbose',type=int,help="verbose",default=0)

Radius and zeta are used to find the uam file specified in the “folder” option, “--proteins” gives a concentration file (see the example), “--coordfolder” is the set of PDB structures used to run UA. The output is then a big file of orientations and rate constants.

Non-automated corona prediction

- Actual prediction is done using CoronaKMC-P3.py. Arguments are:
- parser.add_argument('-f','--fileid',help="String to append to the filename",default="0")
- parser.add_argument('-r','--radius',type=float,help="Radius of the NP",default=35.0)
- parser.add_argument('-p','--proteins',help="Protein file set",default="")
- parser.add_argument('-d','--diffuse',help="Enable surface diffusion",default=0)
- parser.add_argument('-c', '--coarse',help="Treat input as orientations of single protein, report only total numbers",default=0)
- parser.add_argument('-s','--shape',help="Shape of the NP, 1 = sphere, 2 = cylinder",default=1)
- parser.add_argument('-m','--meanfield',help="Enable mean field approximation",default=0,type=int)
- parser.add_argument('-t','--time',help="Number of hours of simulated time",default=1.0,type=float)
- parser.add_argument('-n','--numnp',help="Number of NPs to simulate simultaneously", default = 1, type=int)
- parser.add_argument('-x','--npconc',help="Concentration of NPs", default = 0, type=float)
- parser.add_argument('-H','--hardsphere',help="Enable true hard sphere modelling", default = 0, type=int)
- parser.add_argument('--demo',help="Enable the live demo mode", default = 0, type = int)
- parser.add_argument('--timedelta',help="Time step [s] between showing updates", default = 1e-5, type=float)
- parser.add_argument('-l','--loadfile',help="KMC file for previous run (precoating)", default="")
- parser.add_argument('-P','--projectname',help="Name for project", default="testproject")
-
- FileID can be anything, radius should match the radius specified elsewhere, --proteins is the output from BuildCoronaParams. Diffuse, coarse, meanfield, numnp, npconc, hardsphere can usually be ignored. Loadfile probably isn’t needed in most cases. Timedelta should be adjusted to make sure you see output and time adjusted so runs don’t take forever. Shape should be set to 1 for spheres and 2 for cylinders, or 3 for a truncated sphere (experimental and needs finetuning, but hopefully useable). Finally the projectname can be set to make it easy to find your output. Turning demo mode on slows things down but shows you the output graph and figure as it progresses.