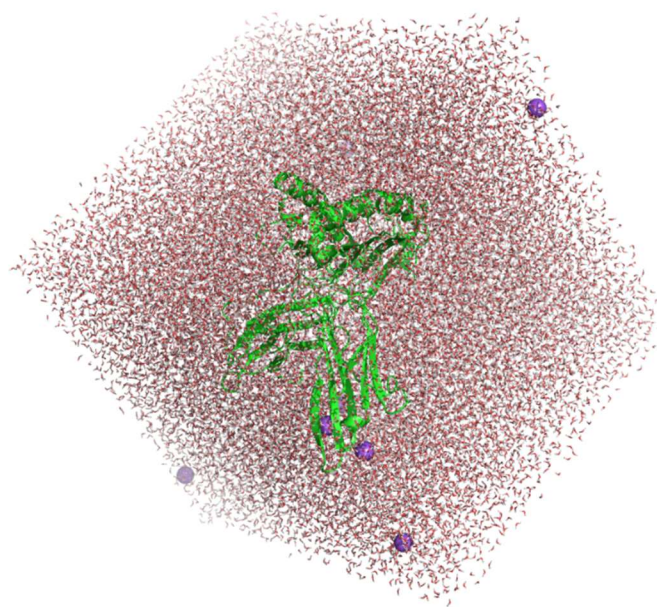


RAPID ANALYSIS SUITE USER MANUAL



1. Structure Analysis Folder:
 - SASA: Instructions for analyzing the solvent accessible surface area.
 - Secondary Structure: Guidance on analyzing the secondary structure of a molecule.
 - Residue Sequence Information: How to extract and interpret information about the residue sequence.
2. Rapid Analysis Folder Editing Bash Scripts:
 - Preliminary checks of GROMACS groups
 - Editing Scripts: Instructions on how to modify and customize the provided scripts.
3. Solvation Analysis Steps:
 - Indexing the Solvation Shell: Steps for identifying and indexing the solvation shell.
 - Tracking Water over MD Trajectory: How to analyze the movement of water molecules throughout the molecular dynamics (MD) trajectory.
4. Diffusion Analysis Steps:
 - Sequential Running of Bash Scripts: Instructions for running a series of bash scripts in a specific order.
 - Mean Square Displacement: How to calculate and interpret the mean square displacement of water.
5. Apparent Diffusion Calculation Steps:
 - Moving MSD Plots: Instructions on analyzing the mean square displacement plots over time.
 - Running Diffusion_C.py Script: How to utilize the provided script to calculate diffusion coefficients.
 - Block Averaging the Data: Guidelines for applying block averaging to improve the accuracy of the results.
6. Radial Distribution Function Steps:
 - Calculating Radial Distribution Function: Guidance on computing the radial distribution function of particles.
 - Integrating Peaks: Steps for integrating peaks in the radial distribution function.
7. Dipole Autocorrelation Function Steps:
 - Moving water index files
 - Explanation and Execution of Dipole analysis
 - Multiple Regression curve fitting of ACF
8. Hydrogen Bond Calculations
 - Moving and creating index files
 - Explanation and Execution of Hydrogen Bonding analysis

Getting Started

Download the [Rapid Analysis](#) and [Structure Analysis](#) folders to your working directory, each folder should contain the following files.

Rapid Analysis Folder	Structure Analysis Folder
<i>template_bashindex.sh</i> <i>template_blockaveraging.sh</i> <i>template_createdetails.sh</i> <i>template_details.sh</i> <i>template_diffusion.sh</i> <i>template_index.sh</i> <i>template_index_solvation_shell.sh</i> <i>template_move_calc_diffcoe.sh</i> <i>template_move_msd.sh</i> <i>template_movefiles.sh</i> <i>template_msd.sh</i> <i>template_rdf.sh</i> <i>template_rdf_integrate.sh</i> <i>template_residue_folders.sh</i> <i>template_solvation_shell.sh</i> <i>Script_Editor.py</i> <i>Diffusion_C.py</i> <i>Diffusion_Extraction.py</i>	<i>SASA.ipynb</i> <i>Secondary_Structure.ipynb</i> <i>Sequence.ipynb</i> <i>Residue_Info.ipynb</i> <i>Residue_Info(GUI).ipynb</i>

Structure Analysis Folder

Solvent Accessible Surface Area

Solvent Accessible Surface Area (SASA) is a measure of the surface area of a molecule that is accessible to solvent molecules. It provides insights into molecular interactions and can be useful in various areas of structural analysis.

The calculation of SASA typically involves using a probe or a set of probes to simulate the presence of solvent molecules around the molecule of interest. The probe is usually a small spherical or non-spherical entity that represents the size and shape of a solvent molecule.

Here is a simplified overview of how SASA is calculated using a probe:

1. Representation: The molecule of interest is typically represented as a set of atoms or molecular coordinates.
2. Probe Placement: The probe(s) are placed around each atom or molecular coordinate of the molecule. Multiple orientations and positions are tested to ensure thorough sampling.
3. Surface Determination: For each probe position, the surface points are determined based on the molecular shape and any overlap with other atoms or probes.
4. Surface Area Calculation: The surface points are used to calculate the area covered by the probe(s). This can be done using various algorithms, such as the rolling ball algorithm or the Connolly algorithm.
5. Solvent Exclusion: The surface area contributed by atoms of the molecule itself is subtracted to exclude internal molecular regions. This is done to obtain the solvent-accessible surface area.
6. Summation: The individual surface areas from all probe positions and orientations are summed up to obtain the total SASA value for the molecule.

The resulting SASA value provides an estimation of the exposed surface area, which is useful for studying interactions with solvents, ligand binding, protein folding, and other structural analyses.

It's important to note that there are different approaches and algorithms available for SASA calculations, and the specific details may vary depending on the software or method used. To calculate SASA we will use an online tool developed by The University of Texas Medical Branch at the following website <https://curie.utmb.edu/getarea.html>. Upload your PDB file, you can change the probe size but 1.4 is appropriate for water. You must enter an email address for the calculation to run, don't worry they won't bombard your inbox. For the desired level of output option choose 2, Area/energy per residue. The website will take you to another page with the calculated data.

The calculated SASA should look as follows (only the first 5 lines are shown)

Query "7192" for meheikes@shockers.wichita.edu submitted to webserver.							
Probe radius : 1.400							
Residue	Total	Apolar	Backbone	Sidechain	Ratio(%)	In/Out	
GLY 1	63.41	30.27	63.41	0.00	72.7	o	
SER 2	64.82	13.76	19.02	45.80	59.2	o	
HIS 3	27.93	21.66	5.65	22.28	14.4	i	

Using your mouse, highlight and copy the values into a .txt file named SASA.txt into your working directory. Now that we have solvent accessible surface area information we can group together residues for potential analysis using the Structure Analysis Suite. Double click on StructureAnalysisSuite.exe file and the application should run. The first option is SASA Analysis, click on the Run SASA Analysis button and load in your SASA.txt file.

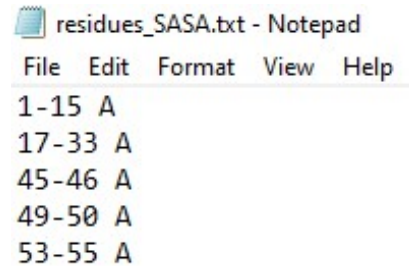


SASA Analysis

1. It reads the SASA data from the file and organizes it into groups based on the values.
2. Each group represents a consecutive range of residues with similar SASA values.
3. If a SASA value is less than 0.5, it starts a new group. If it's greater than or equal to 0.5, it adds the residue to the current group.
4. It assigns a chain ID (a letter) to each group based on the order of the residues.
5. Once all the data is processed, it formats the groups in a specific way, using a range notation if a group has more than one residue.
6. It saves the formatted groups to a new file.
7. Finally, it prints a message confirming that the groups with SASA values of 0.5 or higher have been written to the file.

Note: Although there are many programs to analyze SASA, this python script is coded in such a way that it will only be read from the website listed above.

The output after running the program should look like the example below.



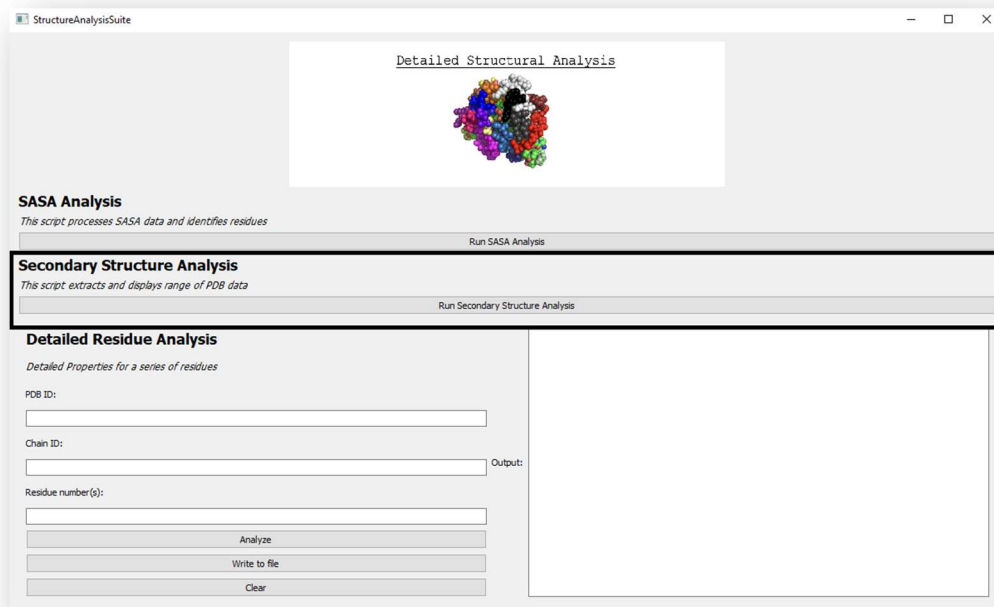
```
File Edit Format View Help
1-15 A
17-33 A
45-46 A
49-50 A
53-55 A
```

Secondary Structure Analysis

The secondary structure of a protein refers to the regular, recurring patterns of local folding within the protein chain, primarily involving alpha helices and beta sheets. Understanding the secondary structure is crucial for many reasons:

1. **Functional implications:** The secondary structure plays a significant role in determining a protein's function. Different secondary structure elements contribute to specific functions such as enzyme catalysis, protein-protein interactions, DNA binding, and membrane insertion. The arrangement of secondary structure elements influences the overall shape and molecular interactions of the protein, thereby dictating its functional capabilities.
2. **Stability and folding:** The secondary structure helps stabilize the three-dimensional structure of a protein. The folding and stability of proteins depend on the correct formation and maintenance of secondary structure elements. They provide structural integrity and can act as nucleation points for protein folding. Disruptions or alterations in secondary structure can lead to misfolding, protein aggregation, and loss of function.
3. **Protein-protein interactions:** Secondary structure elements often participate in protein-protein interactions by forming binding sites or interfaces. Recognition and binding between proteins can rely on specific secondary structure motifs. Understanding the secondary structure is essential for predicting protein-protein interactions and designing therapeutics that target specific binding sites.
4. **Structural characterization:** Determining the secondary structure of a protein is a fundamental step in its structural characterization. Experimental techniques such as X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy rely on secondary structure analysis to interpret and validate the obtained structural data. Secondary structure predictions also aid in the initial modeling and analysis of protein structures.
5. **Evolutionary conservation:** Secondary structure elements can be conserved throughout evolution, indicating their functional importance. By comparing the secondary structure of homologous proteins across species, scientists can gain insights into the conservation of function and identify critical regions or residues that contribute to the protein's activity.

Overall, understanding the secondary structure of a protein provides valuable insights into its function, stability, interactions, and evolutionary relationships. It plays a pivotal role in various fields such as biochemistry, molecular biology, structural biology, drug discovery, and protein engineering. Most likely you are already familiar with the important secondary features of your protein, the Secondary_Structure.ipynb script writes the secondary structures to an easily readable text file. Having the solvent accessible surface area and secondary structure broken up into easily comparable text files will make parsing out residues for analysis much quicker. Download a PDB file of your protein from www.rcsb.org into your working directory. Click the Run Secondary Structure Analysis and load the PDB file you just downloaded.



Note: The Secondary_Structure.ipynb script only works if there is sequence data in your PDB file. If there is not, you can download a PDB file for your protein from rscb.org and it will work.

The Secondary_Structure.txt file will inform you about secondary structure based on residue sequence. You can use this information together with SASA information to determine how to group residues together for analysis.

One more useful tool in the Structure Analysis Suite is the Detailed Residue Analysis which looks at properties of a residue sequence. Follow the prompts for PDB ID, Chain ID, and Residue numbers and the information will be populated to the right. Multiple residues can be analyzed, and the data will continue populating to the right in the output window. You can also save the information to a file for later use. To restart analysis just press clear and the data in the output window will be reset. Below is an example of the output data.

StructureAnalysisSuite

Detailed Structural Analysis

SASA Analysis
This script processes SASA data and identifies residues
Run SASA Analysis

Secondary Structure Analysis
This script extracts and displays range of PDB data
Run Secondary Structure Analysis

Detailed Residue Analysis
Detailed Properties for a series of residues

PDB ID:
2GTZ

Chain ID:
B

Residue number(s):
1-12

Analyze

Write to file

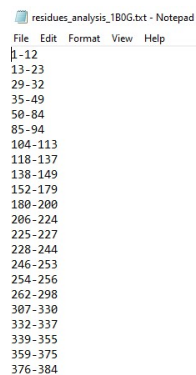
Clear

Output:

Residue 1: Isoleucine (ILE), Charge: 0, Hydrophobic, Cannot form H-bonds
Residue 2: Glutamine (GLN), Charge: 0, Hydrophilic, Can form H-bonds
Residue 3: Arginine (ARG), Charge: 1, Hydrophilic, Can form H-bonds
Residue 4: Threonine (THR), Charge: 0, Hydrophilic, Can form H-bonds
Residue 5: Proline (PRO), Charge: 0, Hydrophobic, Cannot form H-bonds
Residue 6: Lysine (LYS), Charge: 1, Hydrophilic, Can form H-bonds
Residue 7: Isoleucine (ILE), Charge: 0, Hydrophobic, Cannot form H-bonds
Residue 8: Glutamine (GLN), Charge: 0, Hydrophilic, Can form H-bonds
Residue 9: Valine (VAL), Charge: 0, Hydrophobic, Cannot form H-bonds
Residue 10: Tyrosine (TYR), Charge: 0, Hydrophilic, Can form H-bonds
Residue 11: Serine (SER), Charge: 0, Hydrophilic, Can form H-bonds
Residue 12: Arginine (ARG), Charge: 1, Hydrophilic, Can form H-bonds

Total Charge: 3
% Hydrophilic Residues: 66.66666666666666%
% of Residues that can form H-Bonds: 66.66666666666666%

Once you have decided how to break up residues for analysis create a .txt file listing residues to analyze. If your protein contains multiple chains, do not start over sequentially at each chain, just continue counting forward so that you don't have 2 different groups of the same values as this will confuse GROMACS. If your protein has 2 chains with chain A being 100 residues and chain B being 50 residues, just count residues sequentially, 1-150. This file will be used later to create all bash script files for analysis, the file should be formatted like the example below.



```
residues_analysis_1B0G.txt - Notepad
File Edit Format View Help
1-12
13-23
29-32
35-49
50-84
85-94
104-113
118-137
138-149
152-179
180-200
206-224
225-227
228-244
246-253
254-256
262-298
307-330
332-337
339-355
359-375
376-384
```

Rapid Analysis Folder

Preliminary checks of GROMACS groups

Now that you have sectioned off your protein based on the residues you wish to analyze its time to create the bash scripts you will be using for all analysis. First we need to double check the groups that GROMACS has assigned and potentially edit a few of the template files. The template files *index_solvation_shell.sh*, *index_solvation_shell_rdf.sh*, *msd.sh*, *rdf.sh*, and *solvation_shell_COM.sh*, *dipole.sh*, *hbonding.sh* all call specific groups in GROMACS, so we need to make sure that they chosen correctly based on your simulation as each is unique and may affect the grouping.

In terminal, move to the working directory where your MD files are located type the following command.

gmx make_ndx -f file_name.gro

where “file_name” is the name of your specific .gro file. You are not creating index files here; you just want to see how GROMACS has set up the groups. If your groups are set up as follows you do not need to make any changes, do not worry about the number of atoms.

0	System	:	88443	atoms
1	Protein	:	6136	atoms
2	Protein-H	:	3160	atoms
3	C-alpha	:	384	atoms
4	Backbone	:	1152	atoms
5	MainChain	:	1539	atoms
6	MainChain+Cb	:	1899	atoms
7	MainChain+H	:	1912	atoms
8	SideChain	:	4224	atoms
9	SideChain-H	:	1621	atoms
10	Prot-Masses	:	6136	atoms
11	non-Protein	:	82307	atoms
12	Water	:	82299	atoms
13	SOL	:	82299	atoms
14	non-Water	:	6144	atoms
15	Ion	:	8	atoms
16	NA	:	8	atoms
17	Water_and_ions	:	82307	atoms

If they are in a different order than this or you have more groups, follow these steps to edit the template files listed above. When you save updated files DO NOT CHANGE THE FILENAME.

Filenames: template_index_solvation_shell.sh & template_index_solvation_shell_rdf.sh

Below is what the files should look like, you are potentially making two changes. The number 8 should represent the group “SideChain”, and the number 18 should 1 group more than the number of groups that were shown. If you look at my example grouping above there are only 17 groups listed, this script creates new groups so it will create the next group sequentially, group 18. Change the two numbers if needed and save.

```
#!/bin/bash

mkdir resi
cd resi

gmx make_ndx -f ../../../../grofile <<EOF
ri resi & 8
name 18 resi-sidechain
q
EOF

cd ..
```

Filename: template_msd.sh

Below is what the file should look like, there are quite a few edits you can make to this script but right now only change the group number, the rest will be explained in more detail later. The line `group="18"` may need editing. The number 18 represents the group you created, and the group number should be 1 greater than the groups initially listed.

```
#!/bin/bash

i=1 j=1
for k in `seq 0 200 100000`
do
    First_frame="$k"
    Last_frame="$((k+200))"

    Index_file="index$((i++)).ndx"
    Output_file="msd$((j++)).xvg"

    group="18"

    gmx msd -f ../../xtcfile -s ../../grofile -o $Output_file -b $First_frame -e $Last_frame -n $Index_file -beginfit 10 -endfit 40 <<EOF
$group
EOF

rm -rf *#

done
```

Filename: template_rdf.sh

Below is what the file should look like, there are potentially 2 changes that need to be made. The numbers 18 and 13. 18 should reflect the group you created and 13 should be water or the solvent of interest.

```
#!/bin/bash

cd resi
gmx rdf -f ../../xtcfile -s ../../grofile -seltype res_com -n index.ndx -bin 0.05 <<EOF
18
13
EOF
cd ..
```

Filename: template_solvation_shell_COM.sh

Below is what the file should look like, there is only one potential change regarding group selection. If water is the molecule of interest there are no changes that need to be made, if it is any other solvent change the group name “SOL” to match the group of your molecule.

```
#!/bin/bash

cd resi

gmx select -f ../../../xtcfile -s ../../../grofile -select '"Water" group SOL and within 0.6 of group "resi-sidechain"' -seltype res_com -n index.ndx -dt 200 -oi <<EOF
EOF

cd ..
```

Filename: dipole.sh

Below is what the file should look like, there is only one potential change regarding group selection. The line “group=18” where 18 should reflect the index group you created. If the group you created is numbered differently change 18 to the correct number.

```
#!/bin/bash

i=1 j=1
for k in `seq 0 200 100000`
do
First_frame="$k"
Last_frame="$((k+200))"

Index_file="index$((i++)).ndx"
Output_file="dipcorr$((j++)).xvg"

group="18"

    gmx dipoles -f ../../../xtcfile -s ../../../tpzfile -c $Output_file -corr mol -P 1 -b $First_frame -e $Last_frame -n $Index_file <<EOF
$group
EOF

rm -rf *#

done
```

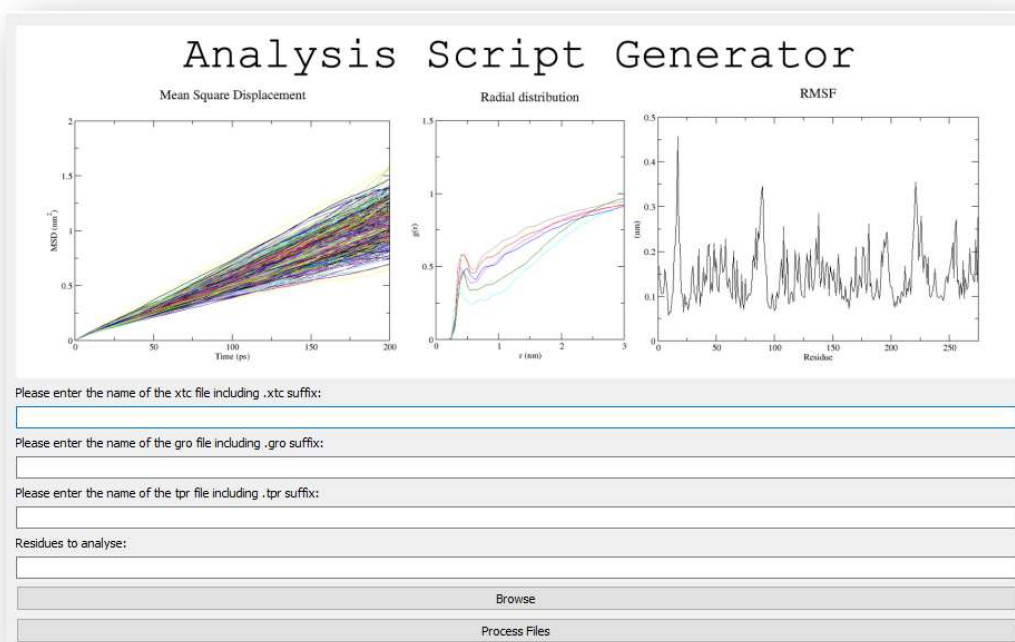
Filename: hbonding.sh

Below is what the file should look like, there are 2 potential changes regarding group selection. Groups 18 and 37 are being selected, where 18 should reflect the index group you created and group 37 should represent equal $((\text{your group number} * 2) + 1)$. For example, if the group you created is numbered 20, the groups below should read 20 and 41. If the group you created is numbered differently change 18 and 37 to the correct numbers.

```
#!/bin/bash
i=1 j=1 l=1
for k in `seq 0 200 100000`
do
    First_frame="$k"
    Last_frame="$((k+200))"
    Index_file="res_water$((i++)).ndx"
    Output_file1="numHbonds$((j++)).xvg"
    Output_file2="lifeH$((l++)).xvg"
    gmx hbond -f ../../xtcfile -s ../../tprfile -num $Output_file1 -life $Output_file2 -b $First_frame -e $Last_frame -n $Index_file -ac <<EOF
18
37
EOF
rm -rf *#
done
```

Creating Bash Scripts

Once your GROMACS groups have been set correctly you are now ready to create all bash scripts you will use to conduct your analysis. The program BashScriptEditor.exe will create all of the scripts needed to run analysis and organize them in their corresponding folder. In the folder containing the BashScriptEditor.exe file should be all template files and the text file listing residues you would like to analyze. You can also use terminal to run the BashScriptEditor.py script or use Jupyter Notebook to run the ipynb file with the same name. When ran you should see a pop-up screen that looks as follows:



The first box is the name of the trajectory file from your simulation, the second box is the name of the structure file, and the third is the name of your tpr topology file. If you have converted from .gro to .pdb post simulation you can use the .pdb filename in the second box. The fourth box is the text file in the current working directory listing the residues you would like to analyze. After all names are entered, and you have selected the residues to analyze text file click the Process Files button. In your current directory you should now see 10 new folders: Diffusion, Dipole, Dipole_Plots, HBond, HBond_Lifetime, MSD, RDF, Residue_Details, Solvation, and Templates. Each of these folders contains the bash scripts needed for each of these types of analysis, the Templates file stores the template bash scripts if you would like to re-run the program.

Solvation Analysis Steps

In your working directory where your simulation files are located (specifically .gro or .pdb and .xtc files) create a new folder named analysis. Go into the newly created analysis folder and move you Solvation folder you created here. Go into the Solvation folder and you will see 2 files: index_solvation_shell.sh and solvation_shell_COM.sh. The first file we will run is the index_solvation_shell.sh script which will create index files based on the residues you wish to analyze. Execute the script by entering the following command in terminal:

bash index_solvation_shell.sh

In your solvation folder you should now have a list of folders for each section of residues you wish to analyze and within each folder is an index file for GROMACS to call to. Next you will run the solvation_shell_COM.sh script. This script will create a file named index.dat in each folder. The index.dat file lists all waters within 6 Angstroms of your index group in 200ps increments. If you are working on a cluster and have the option of slurm scripting it is recommended here as this step takes substantially more time than just indexing groups. Run the script using the bash or sbatch (job scheduling/slurm option) command:

bash or sbatch solvation_shell_COM.sh

These are the only 2 steps for characterization of the solvent around your protein. The first script created residues of interest, and the second tracks water over the course of your MD trajectory. Within each folder you should have an index.ndx file and an index.dat file.

Diffusion Analysis Steps

Move the Diffusion folder you created into the analysis folder, move into the Diffusion folder. There should be 7 scripts: bashindex.sh, createdetails.sh, details.sh, diffusion.sh, index.sh, movefile.sh, msd.sh. First you need to copy the index.dat file and manipulate it slightly. The index.dat file has 3 columns: the first column is the time in picoseconds, the second column lists how many waters (or solvent of interest) are present, and the third column lists all molecules present. We need to get rid of the first two columns only leaving the molecules of interest listed. This can be done by running the createdetails.sh bash script which copies the index.dat file from the Solvation folder. Next run the details.sh script, this will remove the first 2 columns of the file details.txt leaving only the list of your molecules of interest. Below is a before and after of what the file should look like:

index.dat

0.000	93	1741	2601	2707	2718	2723	2735	2832	2867	2930	4083	7444	8142	8163	8175	8186	8235	8244	824
200.000	96	541	1711	2797	2852	2871	3903	4388	7360	7404	7445	8142	8169	8207	8235	8244	8248	8249	826
400.000	108	541	1711	1925	2769	2811	3016	3279	3853	4409	5176	6514	7445	7740	7968	8149	8235	8244	824

details.txt

1741	2601	2707	2718	2723	2735	2832	2867	2930	4083	7444	8142	8163	8175	8186	8235	8244	824
541	1711	2797	2852	2871	3903	4388	7360	7404	7445	8142	8169	8207	8235	8244	8248	8249	826
541	1711	1925	2769	2811	3016	3279	3853	4409	5176	6514	7445	7740	7968	8149	8235	8244	824

The index.sh and msd.sh files need to be moved into each residue folder, to do this run the movefile.sh script. Now that the files have been moved run the bashindex.sh script, this script will execute the index.sh script in each folder. It is recommended to use slurm scripting for this if available. The index.sh script reads your details.txt file and creates an index.ndx file with all solvent molecules within 6 Angstroms of your residues every 200 picoseconds for the entire trajectory. If your trajectory is 100 nanoseconds, in each residue folder you will now have 500 index files, each listing the solvent molecules present at that snapshot in time.

After the bashindex.sh script has finished running, and index files for the entire trajectory are present in every folder you are now ready to run the diffusion.sh script. The diffusion script calls to the msd.sh script you placed in each folder with movefile.sh script. Let's look at what exactly the msd.sh script is doing.

```
#!/bin/bash

i=1 j=1
for k in `seq 0 200 100000`
do

First_frame="$k"
Last_frame="$((k+200))"

Index_file="index$((i++)).ndx"
Output_file="msd$((j++)).xvg"

group="18"

    gmx msd -f ../../1B0G_noPBC.xtc -s ../../nvt_md.gro -o $Output_file -b $First_frame -e $Last_frame -n $Index_file -beginfit 10 -endfit 40 <<EOF
$group
EOF

rm -rf *#

done
```

1. The script defines two variables 'i' and 'j' and sets their initial value to 1.
2. The loop variable 'k' represents the frames of the molecular dynamics simulation, with each frame representing a snapshot of the system at a particular point in time. The loop starts from 0 and increases in steps of 200ps up to 100000ps.
3. Within the loop, 'First_frame' is set to the current 'k' value, and 'Last_frame' is set to 'k + 200'. This sets a window of 200 frames over which the MSD is calculated.
4. 'Index_file' is the file where the output of the 'gmx msd' command will be written. The filename changes with each iteration of the loop (incremented by 'j').
5. 'Output_file' is the file where the output of the 'gmx msd' command will be written. The filename changes with each iteration of the loop (incremented by 'j').
6. The 'group' variable is set to '18' which is the index group you created listing the solvent of interest within 6 Angstroms of the solvent exposed residue sequence.
7. The script then runs the 'gmx msd' command with a set of options:
 - '-f' specifies the trajectory file, which contains the position of the atoms over time.
 - '-s' specifies the structure file, which contains the initial configuration of the atoms.
 - '-o \$Output_file' specifies where to write the output (within the same directory). The output files are written as msd1.xvg, msd2.xvg, etc.

- ‘-b \$First_frame -e \$Last_frame’ specifies the beginning and end of the time period over which to calculate the MSD.
 - ‘-n \$Index_file’ specifies the index file to use.
 - ‘-beginfit 10 -endfit 40’ are parameters related to a linear fit of the MSD as a function of time.
8. A Here Document (‘<<EOF ... EOF’) is used to supply the group number to the ‘**gmx msd**’ command. The number 18 is piped into the command as its input.
 9. The script then removes all files whose names end in ‘#’. The ‘-rf’ options to the ‘**rm**’ command indicate the removal should be recursive (including directories) and forceful (without asking confirmation).
 10. This loop will continue until all frames (0 to 100000 in increments of 200) have been processed.

If you have the option to use a computing cluster, it is highly recommended for this step. If running on a local computer, it will take a over a week to complete. Once your diffusion.sh script has finished running check each of your folders to ensure that the MSD was calculated over the entire trajectory, and you have the appropriate number of msd plots.

Summary of bash script order for diffusion

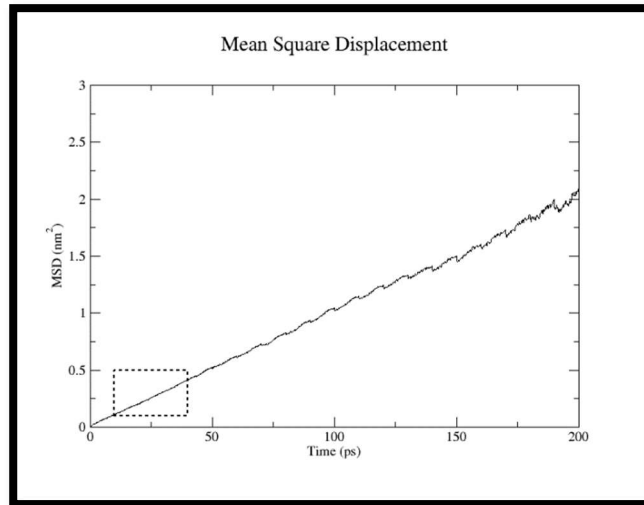
1. createdetails.sh
2. details.sh
3. movefile.sh
4. bashindex.sh
5. diffusion.sh

Diffusion Coefficient Calculation

After the diffusion.sh script has finished running and you have msd.xvg plots for the entire trajectory, you are now ready to calculate the diffusion coefficient. This method of calculating Diffusion Coefficients uses the Einstein relation for Brownian motion in a fluid. The relationship between MSD and diffusion coefficients is given by the following equation.

$$MSD = 2 * n * D * t$$

Where MSD is mean squared displacement, n is the number of dimensions (n=1 for 1D, n=2 for 2D, n=3 for 3D), D is the diffusion coefficient, and t is time. We have a 3D system, and the slope is given by 2nD, we can calculate the diffusion coefficient by dividing the slope by 2n or 6 in our 3D system. To begin analysis first be sure that the MSD folder is in the analysis folder. Within the MSD folder run the script move_msd.sh, this will make residue folders and move all plots to this folder. This consolidates all plots and makes the total file size smaller and easier to zip or transfer. After all plots have been moved, run the move_calc_regression.sh bash script. This will move the Linear_Regression.py file into each folder and execute the python script. The python script loops over all msd files and calculates the diffusion coefficient for each one, converts to the appropriate units of cm²/s, and saves the coefficients into a file named Diff_Linear_Regression.txt. The linear regression is performed over the most linear portion of the plot, between 10 and 40 ps.



After the diffusion coefficient has been calculated for each plot, you can now run the `blockaveraging.sh` script. Because we are analyzing time data it is good practice to do block averaging, it is easier to get a better understanding of how diffusion fluctuates over time. The block averaging script separates the diffusion coefficient data into 20 blocks, which each represent a 5 ns snapshot of diffusion. Block 1 is the average diffusion coefficient from 0-5 ns, Block 2 is 5-10 ns etc. After you have run this script you should see an output file named `Block_Averaging_20Blocks.txt` in each residue folder which should look as follows.

```
Block Averaging Results
-----
Block 1 Average: 8.570262431060667e-06
Block 2 Average: 8.215110320020543e-06
Block 3 Average: 8.435211798329361e-06
Block 4 Average: 8.698136854172948e-06
Block 5 Average: 8.275459246889028e-06
Block 6 Average: 7.801786401265462e-06
Block 7 Average: 8.241160629962586e-06
Block 8 Average: 8.185852638748079e-06
Block 9 Average: 7.595155517596794e-06
Block 10 Average: 7.568303458798201e-06
Block 11 Average: 7.812647070849406e-06
Block 12 Average: 8.097269464819006e-06
Block 13 Average: 8.529891105151009e-06
Block 14 Average: 8.09853897908559e-06
Block 15 Average: 7.76324236975265e-06
Block 16 Average: 7.694144171960712e-06
Block 17 Average: 7.911009875336239e-06
Block 18 Average: 7.540136322525774e-06
Block 19 Average: 7.86034875279433e-06
Block 20 Average: 7.569216178392897e-06

Average of Block-Averaged Values: 8.023144179375564e-06
Standard Deviation of Block-Averaged Values: 3.5389405237961765e-07
Variance of Block-Averaged Values: 1.2524100030966756e-13
```

Once you have the averages calculated you can run `get_diffusion_averages.sh` to consolidate them in order in the main MSD directory, the file will be named `Diffusion_Averages.txt`.

Summary of bash script order for Diffusion Coefficient Calculation

1. `move_msd.sh`
2. `move_calc_regression.sh`
3. `blockaveraging.sh`
4. `get_diffusion_averages.sh`

Radial Distribution (RDF)

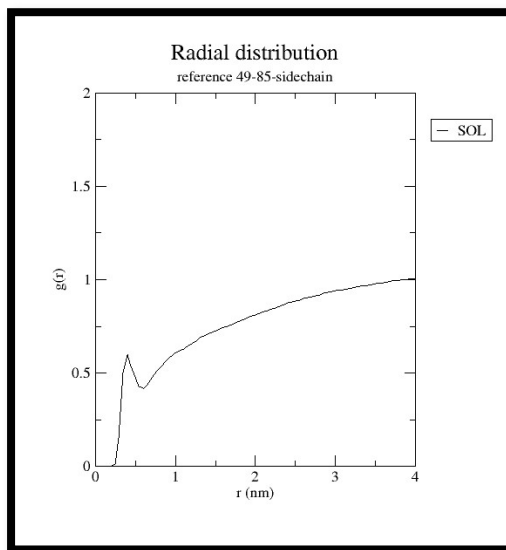
The Radial Distribution Function (RDF), also known as pair distribution function, $g(r)$, in a molecular system is a measure of the probability of finding a particle at a distance r from another particle, compared to that expected for a completely random distribution at the same density. It provides insights into the structural organization of systems at the atomic or molecular level. You will use GROMACS to calculate the RDF using a bash script made earlier. It is important to note that you need to specify the groups of atoms for which you want to calculate the RDF when running in GROMACS. Internally, GROMACS calculates the RDF by creating a histogram of particle distances for each frame for the trajectory and then normalizes the histogram by the volume of the spherical shell corresponding to each distance bin and by the number density of the particles, so that in a completely homogeneous and isotropic system, the RDF approaches 1 at large distances. Again, you will utilize bash scripting to calculate the RDF for each set of exposed residues. Move the RDF folder you created into the analysis folder, then go into the RDF folder. Open the `rdf.sh` file in a text editor and let's look at what the `rdf.sh` script is doing.

```
#!/bin/bash

cd resi
gmx rdf -f ../../../xtcfile -s ../../../grofile -seltype res_com -n index.ndx -bin 0.05 <<EOF
18
13
EOF
cd ..
```

- ‘**-f**’ specifies the trajectory file (.xtc). This file contains the trajectory of the simulation: the positions of all particles at every time step.
- ‘**-s**’ specifies the structure file to be used containing starting geometries.
- ‘**-n**’ makes use of the index file created with the residues you wish to analyze.
- ‘**-bin**’ specifies the width of the bins used for histogramming the distances, has to do with how smooth the output plot is.
- ‘**18**’ is the index group included with the ‘**-n**’ option.
- ‘**13**’ is the solvent of interest.

Running the `rdf.sh` script will produce an `rdf.xvg` plot in each residue folder. The RDF calculation is computationally expensive, so it is recommended to use `sbatch` or run on a cluster if available. After you have run the `rdf.sh` script, check in each folder to ensure that the RDF was calculated, and you have an `rdf.xvg` plot that should look like the plot below.



Next, we need to integrate our RDF plot out to 6 Angstroms. The RDF of pure water shows a distinct solvent shell out to 6 Angstroms, check your RDF plot to see where the appropriate cutoff is for your integration. By integrating the RDF, we can obtain the cumulative number of neighboring atoms within a radius r from the reference atom(s). This provides us with information on the average number of nearest neighbors and allows us to identify the most probable distances between particles. For your liquid simulation, the first peak of the RDF corresponds to the first coordination shell, where the neighboring atoms are most densely packed. The integral of the RDF up to the first minimum gives the average number of atoms in the first coordination shell. To integrate all your RDF plots, run the script `rdf_integrate.sh`. A simple bash command will work fine for this script and should not take long to complete. If on a computing cluster, use the `sbatch` command so that an output file will be generated. You can then use the `grep` command to extract the line with the Integral value in it, for example you could input the following line in terminal.

```
grep 'Integral' slurm-687335.out > Integrals.txt
```

The name `slurm-687335.out` is an example name of the output file from `sbatch rdf_integrate.sh`.

Dipole Autocorrelation Function

In molecular systems, the dipole moment is a measure of the overall polarity of the molecule, which depends on the spatial distribution of charges in the molecule. It is a vector quantity, meaning it has both a magnitude and a direction. The dipole moment can fluctuate over time, depending on the motion and interaction of the particles in the system. The autocorrelation function is a measure of how a quantity (in this case, the dipole moment) correlates with itself as a function of time. It provides a measure of the time scale over which the dipole moment "forgets" its initial orientation, which is related to the rotational relaxation of the molecule. To calculate the dipole autocorrelation, you first need to move the folder titled Dipole to the analysis folder. Within the Dipole folder run the `move_index_diffusion_dipole.sh` bash script to copy and move the `index.ndx` files created in the diffusion folder to the dipole folder. These are the index files that list water within 6 Angstroms of the protein every 200 ps. Next you will run the `move_dipolescript.sh` script which will move the `dipole.sh` script into each residue folder. Now you are ready to run the `dipole_calc.sh` script if you are working on a computing cluster sbatch the dipoles slurm script to run the `dipole_calc.sh` bash script. Let's look at what the `dipole.sh` script is doing.

```
#!/bin/bash

i=1 j=1
for k in `seq 0 200 100000`
do

First_frame="$k"
Last_frame=$((($k+200))"

Index_file="index$((i++)).ndx"
Output_file="dipcorr$((j++)).xvg"

group="18"

    gmx dipoles -f ../../../xtcfile -s ../../../tprfile -c $Output_file -corr mol -P 1 -b $First_frame -e $Last_frame -n $Index_file <<EOF
$group
EOF

rm -rf *#

done
```

1. The script sets initial values for the variables **i** and **j** to 1.
2. A loop is then started that runs from 0 to 100000 in steps of 200. The loop variable **k** represents the frames of the molecular dynamics simulation, with each frame representing a snapshot of the system at a certain point in time.
3. Within each iteration of the loop:
 - **First_frame** and **Last_frame** are defined as **k** and **k + 200** respectively, indicating a window of 200 frames over which the dipole moment is calculated.
 - **Index_file** is defined as a file containing indices of the molecules for which the dipole moment should be calculated. Its name changes with each iteration of the loop.
 - **Output_file** is defined as the file where the output of the **gmx dipoles** command will be written. Its name also changes with each iteration of the loop.

- **group** is set to "18", indicating that the 18th group defined in the index file will be analyzed. This is the group containing a snapshot of all waters within 6 Angstroms of the residues every 200 ps.
4. The **gmxdipoles** command is then executed with several options:
 - **-f ../../xtcfile** specifies the input trajectory file.
 - **-s ../../tprfile** specifies the input structure file.
 - **-c \$Output_file** specifies the output file for the total and/or average cosine of the angle with the z-axis.
 - **-corr mol** specifies that the correlation function should be calculated per molecule.
 - **-P 1** means that only the first principal component of the inertia tensor is used.
 - **-b \$First_frame -e \$Last_frame** specifies the start and end time for analysis in the trajectory file.
 - **-n \$Index_file** specifies the index file to be used.
 5. The **gmxdipoles** command is run for the group of molecules specified in the **Index_file** within the given **First_frame** and **Last_frame** time window.
 6. This process is repeated for every 200-frame window in the specified frame range.

After the script has finished running you should see 500 dipcorr.xvg files in each of the residue folders. Now you are ready to do curve fitting for the ACF function. Move the folder named Dipole_Plots into the analysis folder and run the script move_dipole_plots.sh, this will move all desired plots to this folder. After all plots have been moved, run the script dipole_averaging.sh, this bash script calls to the dipole_averaging python script which averages all plots and generates a new plot named average_dipcorr.xvg. Next is to fit the curve and get the optimized coefficients for the fitted equation. To do this run the script curve_fitting.sh, this script moves the curve_fitting python script into each folder and executes it, producing a plot and optimized parameters in each folder. After you have done the curve fitting, running the bash script get_parameters.sh will generate a file named Optimized_Coefficients.txt with all coefficients listed.

1.4.8 HBond Folder

The HBond folder contains 8 scripts which are used to move and create index files and calculate the hydrogen bond autocorrelation function. First the user must move index.ndx files from both the solvation and diffusion folders, the index file from solvation creates the group defining the atoms of the residue of interest. The index files from diffusion create a unique group that represents all waters within 6 Angstroms every 200 picoseconds. All index files can be moved by running both move_index_diffusion_hbond.sh and move_index_solvation_hbond.sh. Now that the 500 index files from diffusion and the index file from solvation are in each residue folder, several new index file which combines the two needs to be made. To create new index files, the user must first run move_ndx_hbond.sh to move all necessary bash scripts into each folder. After appropriate scripts have been moved, the user can run the script build_index_files.sh which will consolidate the residues of

interest and the waters of interest into a series of index files. Below is an example of the script that accomplishes this with a brief description of the code.

```
#!/bin/bash

i=1 j=1
for k in `seq 0 200 100000`
do
    First_frame="$k"
    Last_frame="$((k+200))"

    Index_file="index$((i++)).ndx"
    Output_file="res_water$((j++)).ndx"

    gmX make_ndx -f ../../1I7T_md3.gro -o $Output_file -n index.ndx $Index_file <<EOF
q
EOF

rm -rf *#

done
```

1. **i=1 j=1**: This line initializes two variables **i** and **j** to 1. These variables will be used to number the output index files.
2. **for k in seq 0 200 100000`**: This begins a loop that iterates over a sequence of numbers from 0 to 100000 in steps of 200. Each number in the sequence is assigned to the variable **k**.
3. **First_frame="\$k"** and **Last_frame="\$((k+200))"**: These lines assign the current value of **k** and **k+200** to the variables **First_frame** and **Last_frame**, respectively. These variables represent the range of frames to be analyzed.
4. **Index_file="index\$((i++)).ndx"** and **Output_file="res_water\$((j++)).ndx"**: These lines create variables **Index_file** and **Output_file** that contain the names of the input and output index files, respectively. The names are numbered according to the current iteration of the loop.
5. **gmX make_ndx -f ../../1I7T_md3.gro -o \$Output_file -n index.ndx \$Index_file <<EOF**: This command runs the GROMACS **gmX make_ndx** command. The **-f** flag specifies the input **.gro** file, **-o** flag specifies the output index file's name, and **-n** flag indicates an existing index file to add the group to.
6. **q**: The **q** command, within the EOF (end-of-file) markers, quits the **gmX make_ndx** command.
7. **rm -rf *#**: This line removes any files whose names end with "#". These are typically temporary files created during the execution of the script and are not needed after the script has finished running.
8. **done**: This line ends the loop.

1.4.9 HBond_Lifetime

The HBond_Lifetime folder contains 4 scripts which will be utilized to calculate hydrogen bond lifetime by doing a multi exponential curve fitting of the hydrogen bond ACF plots previously generated. By running the script `move_hb_data.sh` all autocorrelation plots and the output "screen.txt" will be moved to the HBond_Lifetime folder. After all plots have been moved the user will run the script

hbond_averaging.sh which will average all plots over the entire trajectory producing a new plot named average_HBond.xvg. The next step is to perform a multi exponential curve fitting, this is done by running the script curve_fitting.sh.