Report on: Molecular dynamics simulations of the stabilization of pea diamine oxydase structure with trehalose

Pau Domínguez-Sorribas[1]*   Jordi Villà-Freixa[1,2]

[1]Research Group on Bioinformatics and Bioimaging, Department of Biosciences,
Universitat de Vic - Universitat Central de Catalunya (UVic-UCC), Vic, Spain
{pau.dominguez, jordi.villa}@uvic.cat
[2]Institut de Recerca i Innovació de la Catalunya Central (IRIS-CC), Vic, Spain.

August 27, 2024

Abstract

This report delves into the molecular dynamics (MD) simulations of pea diamine oxidase (DAO) in the presence and absence of trehalose molecules, using with the AMBER MD software suite. The primary aim is to investigate the stabilizing effects of trehalose on DAO, an enzyme crucial for the metabolism of biogenic amines. DAO simulations are conducted in a truncated octahedral box with periodic boundary conditions to emulate an infinite system, enhancing the realism of the molecular environment.

Keywords: Diamine oxydase, protein stability, trehalose, molecular dynamics, AMBER

Contents

---

*Author One developed his work during his BSc final project

1   Introduction

It is well recognized that saccharides have exceptional properties in helping biomolecules such as proteins preserve their native structures under harsh conditions, e.g., high or low temperatures, and dehydration[1–3]. Among naturally available saccharides, trehalose, a nonreducing homo-disaccharide in which two D-glucopyranose units are linked together by an α-1,1-glycosidic linkage (Figure 1 (a)) is probably the best biomolecule stabilizer[1, 4–7]. As a result, trehalose is widely used as additive for long-term preservation of therapeutic proteins, foods, and cosmetics industries[1, 8–10].

1.1   Diamine oxidase

In this project, we focus on a probiotic developed by a partner biotech company. This probiotic, among other constituents, contains diamine oxidase (DAO), a copper amine oxidase (CAO) which, due to its instability, gets rapidly degraded in the final product. A representation of DAO structure, as obtained from PDB code 1KSI[11], is shown in Figure 1.

Figure

Copper amine oxidases (CAOs) have been extensively studied across various domains of life[12], revealing a common architectural motif despite differences in origin. Structures from bacteria (Escherichia coli; Arthrobacter globiformis), yeast (Hansenula polymorpha; Pichia pastoris), plants (Pisum sativum), and mammals (Homo sapiens; Bos taurus) share homodimeric arrangements[13], each subunit typically comprising around 700 amino acid residues (Figure 3). The pioneering E. coli enzyme's structure[14] resembled a mushroom, with the stalk formed by D1 domains and the cap by D2, D3, and D4 domains of each subunit.

1.1.1   Cofactors and postranslational modifications

A part from Cu^{+2} and Mn^{+2} ions, this protein has two important molecules attached to its structure (Figure:2). First one is the modified residue topaquinone (TPQ) and the second one would be NAG (N-Acetyl-D-Glucosamyine) that is presented in two forms, as a ligand where only one molecule of NAG is linked to the protein by an asparagine, and as polymer where two molecules of NAG joined together are linked to an asparagine in the protein.

TPQ or topaquinone is the modified residue we find in our enzyme DAO. It consists on a redox cofactor derived from tyrosine residue where two additional oxygen atoms have been attached. Its biogenesis proceeds by a sel-processing pathway which requires only the protein, copper and molecular oxygen. Not only is found in plants like in our case but in bacteria, mammals or fungi. As a result of previous studies, specifically those that helped out to crystallographically establish DAO structure, we know that TPQ presents a conformational flexibility which is the capacity of side chains and polypeptide backbone fragments to take on various conformations while preserving the original protein structure [11].

NAG generally is refered to 2-acetoamido-2deoxy-beta-D-glucopyranose or N-Acetyl-D-Glucosamine which is a ligand attached to the protein. It is also a N-Glycan since is covalently attached to the protein at asparagine (Asn) rsidues by an N-glycosidic bond. N-Glycans affect to the protein properties, including their conformation, solubility, activity... Which are factors to must have into account to study a protein. It derives from a glucose monosaccharide forming an amide between glucosamine and acetic acid. In our case we have six different NAGs in four different positions: 2 of them are
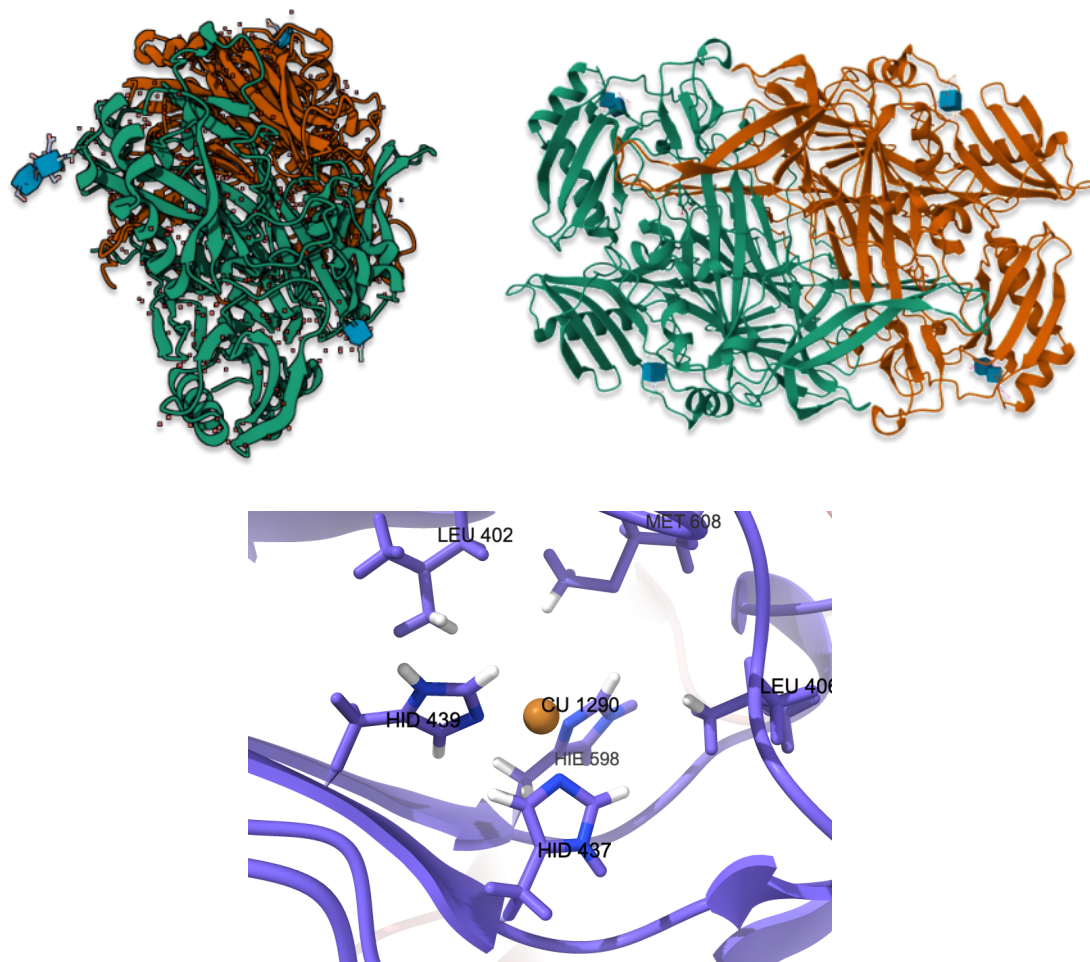
Figure 1: a) Quaternary structure of DAO, indicating α helices (yellow), β strands (green) and loops (no color). Notice the position of TPQ (in red). b) Detail of the active site, with the coordination of the Cu^{+2} ion in chain A. Note that the numbering corresponds to that in the AMBER calculation (5 residues are msising in the N-terminal for each of the two units of the biologically active DAO. Obtained with ChimeraX; PDB code: 1KSI.

simple N-Glcyans covalently attached to the protein at asparagine, but only one NAG linked to that asparagine; the other four consist in two carbohydrate polymers also attached to an asparagine: Each polymer consists in two NAGs linked between them and then attached to an asparagine[15].

Comparing various studies which has N-linked glycans, we can asume that our NAGs exercise the same function related with proper folding of the protein and their structure stabilization.

Notably, the two active sites, situated in the D4 domains, house copper ions and the characteristic trihydroxyphenylalanine quinone (TPQ) cofactor. TPQ, crucial for enzyme function, forms post-translationally from a tyrosine residue in the presence of copper ions and molecular oxygen.

The enzymatic mechanism involves a ping-pong reaction, commencing with a reductive half where the substrate amine reacts with TPQ to form a Schiff base. This step is followed by an oxidative half, resulting in the release of the aldehyde product and reoxidation of the enzyme facilitated by molecular oxygen. While structures from other organisms mirror the E. coli CAO, they typically lack the D1 or stalk domain. Despite these variations, the fundamental architecture and catalytic mechanism remain conserved, highlighting the evolutionary significance and functional conservation
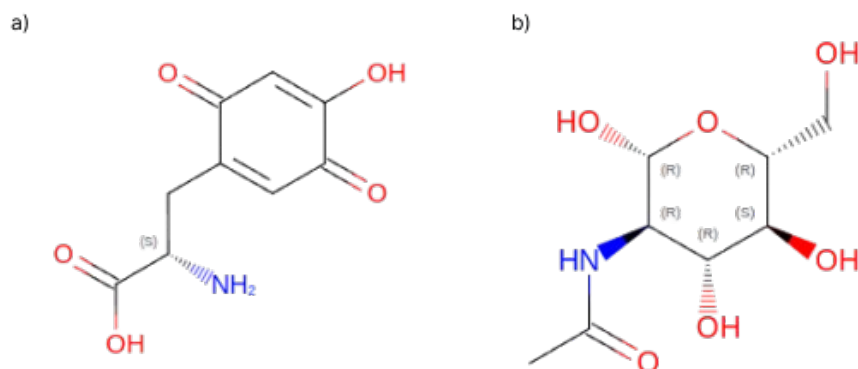
Figure 2: Chemical structure of diamine oxidase cofactors: TPQ (a) and NAG (b)configuration image obtained by PDBe code= 1KSI: NAG, TPQ

chains A,B	6	VQHPLDPLT KEEFLAVQT I VQNK YP I SNNR LAFHY I GL DDPE KDH
chains A,B	51	VLRYET HPTL VS I PRK I FVVA I I NSQTHE I L I NLR I RS I VSDN I H
chains A,B	96	NGYGFP I LS VDEQSLA I KLPLK Y PPF I DSVKK RGLN LSE I VCSSF
chains A,B	141	TMGWFGEEKNVR TVRLDCFM KESTVN I YVRP I TG I T I VAD LDLMK
chains A,B	186	I VEYHD RD I EAVPTAENTEYQ VSK QSPPFGPKQHSLTSHQPQGPG
chains A,B	231	FQ I NGH SVSWANWKFH I GFD VRAG I V I SL AS I YD LEKH KSRRVLY
chains A,B	276	KGY I SELF VPYQDPTEEFYFKTFF DSGEF GFGLSTVSL I PNRDCP
chains A,B	321	PHAQF I DT YVHSANGT P I LL KNA I CVFEQYGN I MWRHTENG I PNE
chains A,B	366	S I EESRTEVNL I VRT I VTVGN X DNV I DWEFKASGS I KPS I ALSG I
chains A,B	411	LE I KGTN I KH KDE I KEDLHGKLVSANS I G I YHDHFY I YYLDFD I D
chains A,B	456	GTHN SFEKTSLKTVR I KDGSSKRKS YWTTETQT AKT ESDA K I T I G
chains A,B	501	LAP AELVVVNPN I KTAVGN EVGYRL I PA I PAHPLLTEDD YPQ I RG
chains A,B	546	AFTN YN VWVTA YNRTEKWAGGLYVDHSRGDDT LAVWTKQ NRE I VN
chains A,B	591	KD I VMWHVVG I HH VPA QED FP I MPL LSTSF ELRP TNFFERNPVLK
chains A,B	636	T LSPRDVAWPGC

Figure 3: Primary structure of DAO, indicating α helices (yellow), β strands (green) and loops (no color). Notice the position of TPQ (in red). Obtained from ChimeraX; PDB code: 1KSI.

of CAOs across diverse biological systems

The diamine oxidase used in this probiotic comes from a plant which is an enzyme involved in the metabolism, oxidation, and inactivation of histamine and other polyamines; being the main reason why it is used in this product because intolerance to histamine or its non-regulation can lead to gut problems.

Referring to histamine, the low level concentration of intestinal histaminase (whose the main function is to oxidize histamine) in the human body, together with some food intake like chocolate, beer, red wine or fish can eventually cause histaminosis, which is a dysfunction associated with pseudo allergy phenomenon[16] where an excess of histamine in blood is present.

1.2 Trehalose as a protein stabilizer

Mycose, tremalose or better known as trehalose (α,α-trehalose) is a common disaccharide derived from glucose. It is formed by a 1,1-glycosidic bond between two α-glucose (Figure 4).

Thanks to its chemical and biological properties it's expected to be a good stabilizer for our protein
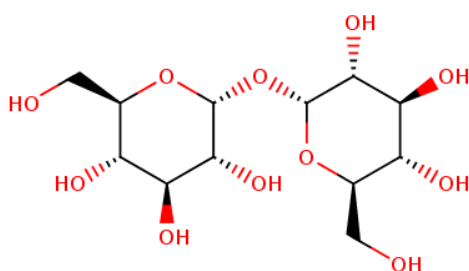
Figure 4: Chemical structure of trehalose

in the further simulations. It's been proved that some carbohydrates have the ability to stabilize proteins in several extreme conditions because of its resistance to hydrolysis, high temperatures or acidic conditions[1].

Through hydrogen bonding, trehalose directly interacts with and influences the interactions between the other species by: Intermolecular hydrogen bonding interaction causing trehalose molecules to cluster; formation of hydrogen bonds between water and trehalose; influencing the strength of the water-water hydrogen bonding network without affecting protein-water hydrogen bonding and forming hydrogen bonds towards proteins while also replacing surrounding water molecules, reducing the possibility of hydrogen bonding from water to protein in accordance with the "water replacement" scenario.

1.3 Hypothesis and objectives

Main hypothesis is that, in agreement with previous studies, trehalose follows a water replacement process around the protein structure to estabilize the latter. This in connected with a change in entropy in the water solvation, similar to what occurs in hydrophobic processes, where water is displaced from an interaction. This may mean that the effect of trehalose is not enthapically driven but entropically, and the presence of trehalose molecules around the protein limits the flexibility of the protein and makes it, thus, more reluctant to denaturalization processes. The fact that trehalose is the best saccaride to create such environment should be tested with other saccarides, and thus the inner features of trehalose that makes it especiffically relevant to enhance the protein structure may be revealed.

Objectives:

1. Generate metrics for the stability of DAO surreounded by different solvent mixtures (pure water, water/trehalose mixtures, and water/other dissaccarides mixtures)

2. Find ways to evaluate the enthalpy and entropy of the water desolvation processes in the mixtures simulations, and apply them to dissect the free energy of solvation in each case.

2 Methodology

All simulations have been done with Amber following the workflow in Figure S1. Protein structures where visualized with ChimeraX [17] and trajectories analyzed with VMD[18] and cpptraj[19], as well as xmgrace[20] for plotting.

On AMBER calculations:
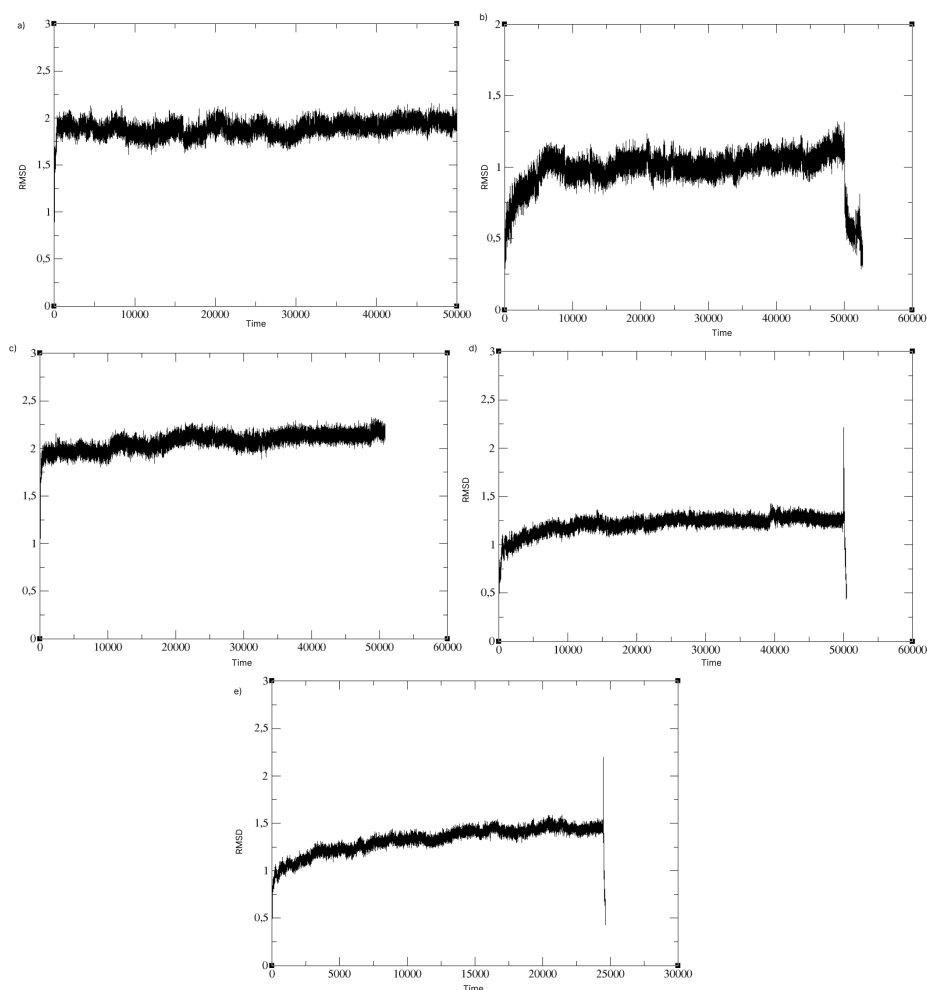
- GAFF2 force field[21] for TPQ

Figure 5: Graphical representations of RMSD results from each of the trehalose simulations: a) 1 trehalose molecule, b) 2 trehalose molecules, c) 5 trehalose molecules, d) 10 trehalose molecules and e) 25 trehalose molecules.

3 Results

3.1 Trehalose simulaltions results

In order to firstly verify that trehalose behaves as a stable molecule in a water system with same parameterization as the final simulation of DAO with NAG, ions and trehalose; 5 different simulations of different concentration of trehalose molecules were run roughly 100 ns each. The results given from production process that are more important for us to analyze are the trajectory ones (`*.nc`) and the initial product topology file from each tleap. To visualize that trajectory with an RMSD (Root Mean Square Deviation) the tool cpptraj was used.

Figure 5 presents the Root Mean Square Deviation (RMSD) plots for five molecular dynamics (MD) simulations of trehalose molecules. Subfigures a, b, c, d, and e depict the RMSD trajectories for systems with 1, 2, 5, 10, and 25 trehalose molecules, respectively. All simulations were conducted for 100 ns, with the RMSD values stabilizing after an initial equilibration period. The plots indicate that each system maintained structural stability throughout the simulation duration, showcasing the consistent behavior of trehalose molecules in varying quantities under the simulated conditions. This stability is crucial for validating the reliability of the MD setup and a first indication of the appropriateness of the force field used to represent the disaccaride interactions.

3.2 DAO without NAG, ions or trehalose in water system results

After collecting all the data once the simulation was runned during 500 ns, it has to be prepared to be analyzed. Considering that each simulation result takes up a lot of space in the hard disk the the process must be carried out in an orderly and correct manner. After trehalose results next results to be collected were those from DAO without NAG or trehalose simulation. Since the calculations took longer to be performed, the more runs were needed and therefore the size of each file. Due to this files size and the local computer used for this project which does not have a high storage capacity, the graphs created to visualize the results are from only some of the files (Figure 6). As it can be seen, even though the graphs are not made from all the files we can observe by the RMSD that the trehalose started to become unsettled.
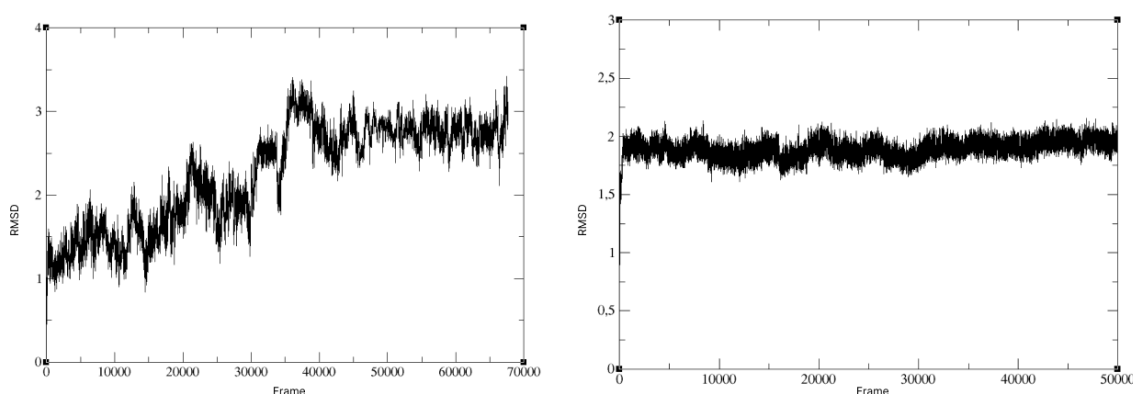

Figure 6: Two RMSD graphs from the first four runs of the simulation (left image) and the first run of the simulation (right image) showing the measure of the average distance between the atoms

3.3 DAO with NAG and trehalose results

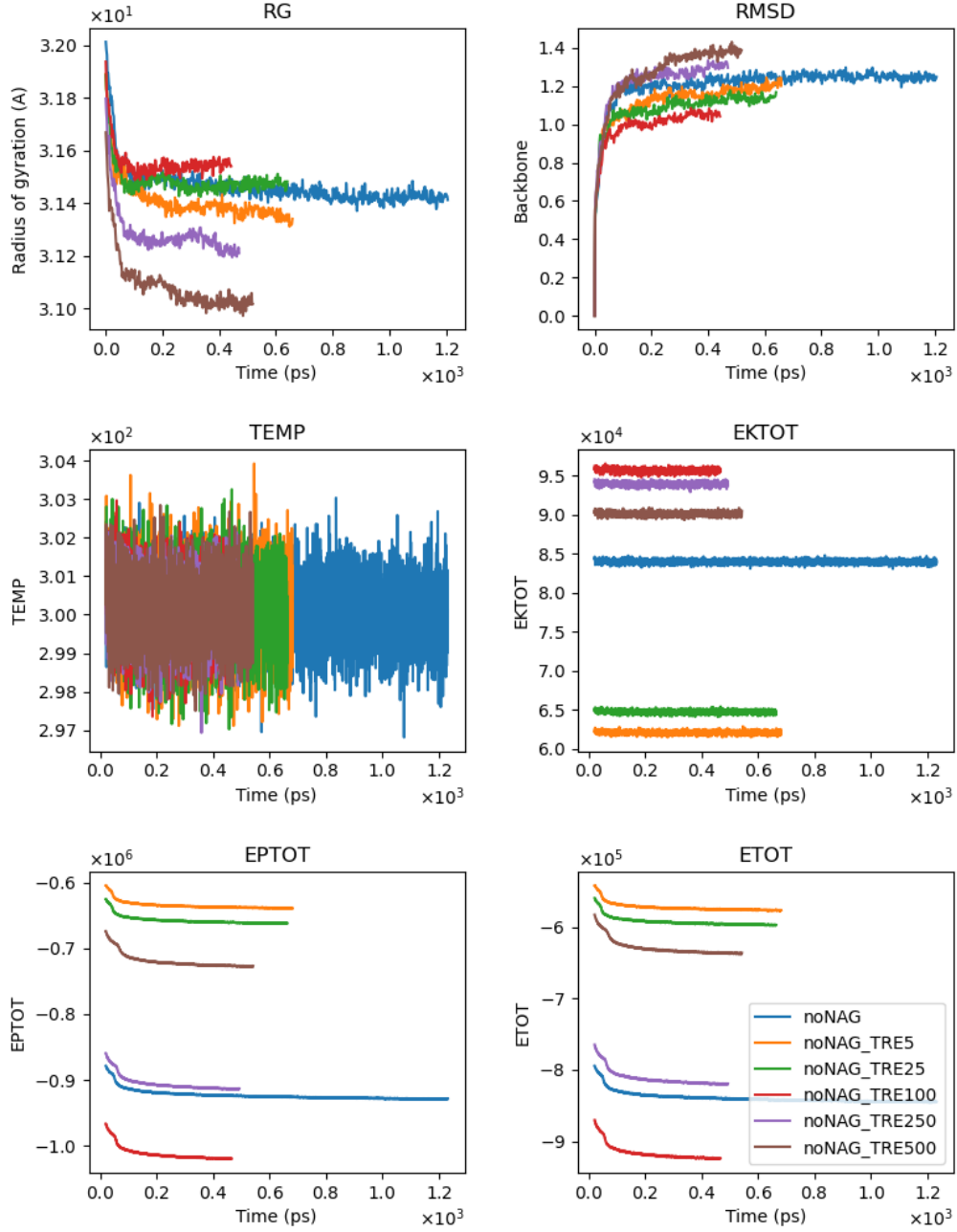Summary of results is shown in Figure 7

Figure 7: summary of all calculations

4 References

References

[1] Qiang Shao, Jinan Wang, and Weiliang Zhu. Trehalose Stabilizing Protein in a Water Replacement Scenario: Insights from Molecular Dynamics Simulation, December 2019. Pages: 2019.12.27.889063 Section: New Results.

[2] Lois M Crowe. Lessons from nature: the role of sugars in anhydrobiosis. *Comparative Biochemistry and Physiology Part A: Molecular & Integrative Physiology*, 131(3):505–513, March 2002.

[3] X. M. Cao, X. Yang, J. Y. Shi, Y. W. Liu, and C. X. Wang. The effect of glucose on bovine serum albumin denatured aggregation kinetics at high concentration: The master plots method study by DSC. *J Therm Anal Calorim*, 93(2):451–458, August 2008.

[4] Alain Hédoux, Jean-François Willart, Laurent Paccou, Yannick Guinet, Frédéric Affouard, Adrien Lerbret, and Marc Descamps. Thermostabilization Mechanism of Bovine Serum Albumin by Trehalose. *J. Phys. Chem. B*, 113(17):6119–6126, April 2009.

[5] Nishant K. Jain and Ipsita Roy. Role of trehalose in moisture-induced aggregation of bovine serum albumin. *European Journal of Pharmaceutics and Biopharmaceutics*, 69(3):824–834, August 2008.

[6] Nishant Kumar Jain and Ipsita Roy. Effect of trehalose on protein structure. *Protein Sci*, 18(1):24–36, January 2009.

[7] Jai K. Kaushik and Rajiv Bhat. Why is trehalose an exceptional protein stabilizer? An analysis of the thermal stability of proteins in the presence of the compatible osmolyte trehalose. *J Biol Chem*, 278(29):26458–26465, July 2003.

[8] John F. Carpenter, Michael J. Pikal, Byeong S. Chang, and Theodore W. Randolph. Rational Design of Stable Lyophilized Protein Formulations: Some Practical Advice. *Pharmaceutical Research*, 14(8):969–975, 1997.

[9] Takanobu Higashiyama. Novel functions and applications of trehalose. *Pure and Applied Chemistry*, 74(7):1263–1269, January 2002.

[10] Tsutomu Arakawa, Steven J. Prestrelski, William C. Kenney, and John F. Carpenter. Factors affecting short-term and long-term stabilities of proteins. *Advanced Drug Delivery Reviews*, 46(1):307–326, March 2001.

[11] V.Kumar, D.M. Dooley, H.C. Freeman, J.M. Guss, I. Harvey, M.A. McGuirl, M.C. Wilce, and V.M. Zubak. RCSB PDB - 1KSI: CRYSTAL STRUCTURE OF A EUKARYOTIC (PEA SEEDLING) COPPER-CONTAINING AMINE OXIDASE AT 2.2A RESOLUTION, 1996.

[12] Aaron P McGrath, Kimberly M Hilmer, Charles A Collyer, Eric M Shepard, Bradley O. Elmore, Doreen E Brown, David M Dooley, and J Mitchell Guss. The structure and inhibition of human diamine oxidase,. *Biochemistry*, 48(41):9810–9822, October 2009.

[13] Giovanni Floris and Bruno Mondovì, editors. *Copper amine oxidases: structures, catalytic mechanisms, and role in pathophysiology*. CRC Press, Boca Raton, 2009. OCLC: ocn253187967.

[14] M. R. Parsons, M. A. Convery, C. M. Wilmot, K. D. Yadav, V. Blakeley, A. S. Corner, S. E. Phillips, M. J. McPherson, and P. F. Knowles. Crystal structure of a quinoenzyme: copper amine oxidase of Escherichia coli at 2 A resolution. *Structure*, 3(11):1171–1184, November 1995.

[15] E. Deniz Tekin. Investigation of the effects of *N*-Acetylglucosamine on the stability of the spike

protein in SARS-CoV-2 by molecular dynamics simulations. *Computational and Theoretical Chemistry*, 1222:114049, April 2023.

[16] Catherine Jumarie, Marilyne Séïde, Lucia Marcocci, Paola Pietrangeli, and Mircea Alexandru Mateescu. Diamine oxidase from white pea (lathyrus sativus) combined with catalase protects the human intestinal caco-2 cell line from histamine damage. *Applied Biochemistry and Biotechnology*, 182:1171–1181, 7 2017.

[17] Meng EC, Goddard TD, Pettersen EF, Couch GS, Pearson ZJ, Morris JH, and Ferrin TE. Ucsf chimerax: Tools for structure building and analysis. *Protein Sci.*, 32:e4792, 2023.

[18] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

[19] Daniel R. Roe and Thomas E. III Cheatham. PTRAJ and CPPTRAJ: Software for Processing and Analysis of Molecular Dynamics Trajectory Data. *J. Chem. Theory Comput.*, 9(7):3084–3095, July 2013. Publisher: American Chemical Society.

[20] PJ Turner. XMGRACE, 2005.

[21] Xibing He, Viet H. Man, Wei Yang, Tai-Sung Lee, and Junmei Wang. A fast and high-quality charge model for the next generation general AMBER force field. *The Journal of Chemical Physics*, 153(11):114502, September 2020.

[22] D.A. Case, H.M. Aktulga, K. Belfon, I.Y. Ben-Shalom, J.T. Berryman, S.R. Brozell, D.S. Cerutti, T.E. Cheatham III, G.A. Cisneros, V.W.D. Cruzeiro, T.A. Darden, N. Forouzesh, G. Giamba, T. Giese, M.K. Gilson, H. Gohlke, A.W. Goetz, J. Harris, S. Izadi, S.A. Izmailov, K. Kasavajhala, M.C. Kaymak, E. King, A. Kovalenko, T. Kurtzman, T.S. Lee, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, M. Machado, V. Man, M. Manathunga, K.M. Merz, Y. Miao, O. Mikhailovskii, G. Monard, H. Nguyen, K.A. O'Hearn, A. Onufriev, F. Pan, S. Pantano, R. Qi, A. Rahnamoun, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, A. Shajan, J. Shen, C.L. Simmerling, N.R. Skrynnikov, J. Smith, J. Swails, R.C. Walker, J. Wang, J. Wang, H. Wei, X. Wu, Y. Wu, Y. Xiong, Y. Xue, D.M. York, S. Zhao, Q. Zhu, , and P.A. Kollman. Amber 2023, 2023.

A Supplementary material

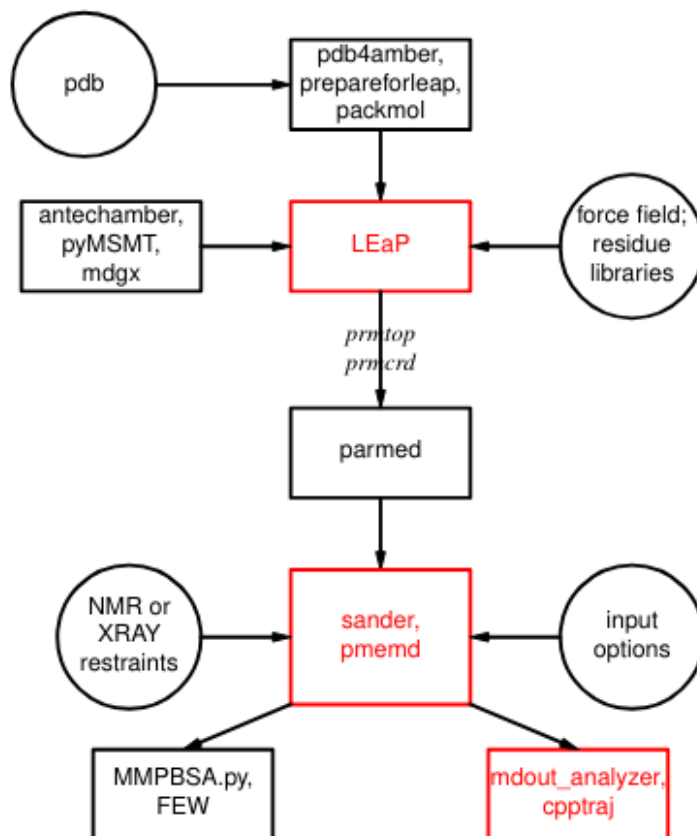All details on the calculations are compiled in the GitHub repository https://github.com/JordiVillaFreixa/Pau_TFG_DAO.

Figure S1: Basic workflow of a molecular dynamics simulation done with Amber (extracted from Amber 2023 Reference Manual)

A.1 Preprocessing files

The "linkofinterest" is where it goes the copied link that refers to the files. Download command has to be runned multiple times, one for each file is precised. After the download, is recommended to check and verify that we have the exact ones and the file is correctly structured. Analyzing 1KSI PDB file, the following message can be found: "NAG C 1 HAS WRONG CHIRALITY AT ATOM C1 NAG D 1 HAS WRONG CHIRALITY AT ATOM C2"

Until the protein is analyzed it cannot be assured that it may affect in the future simulations, although it has been previously stated that N-glycans affect to the protein in several ways, so it is likely to have an effect in some way.

A.2 Preparation of files

Once all the needed files are downloaded in their respective structure, they have to be prepared for the use in Amber, because if left as they are, Amber is not going to recognize their structure and will not be useful. To start it will be prepared the main objective file of this study which is DAO. By running the simply command below, the file will be processed:

Command 1: Preparing a pdb file to be used in `Amber`

```
    pdb4amber -i 1KSI.pdb -o DAO.pdb --reduce --dry
```

The command itself, what it does is to restructure the input PDB file into a new one that Amber can recognize. In addition, the "reduce" and "dry" words remove the crystallographic waters the file contains and adds hydrogen atoms in their optimal positions [22].

Being the protein PDB file prepared, the following thing to do is prepare the cofactor files remaining. First file of the two to be prepared after the protein one is TPQ.cif, the modified residue in DAO. As a modified residue is not considered a standard residue that the program can recognize, so a new entry on the Amber library will be needed for Amber to recognize and process once the file is loaded for the final simulations. The first thing TPQ.cif file needs, is to be processed by Antechamber, which is a set of auxiliary programs for molecular mechanic studies that solves the following problems during the MM calculations:

1. Recognition of the atom type

2. Recognition of bond type

3. Judging the atomic equivalence

4. Generation of residue topology file

5. Finding of missing force field parameters

To obtain an output file with atom types and partial charges an Antechamber command is going to be used. The command is as follows:

Command 2: Antechamber for TPQ
```
    antechamber -fi ccif -i TPQ.cif -bk \
             TPQ -fo ac -o tpq.ac -c bcc -at amber
```

The usage of Antechamber may vary depending on the input file format, the output file format, charge method... So it depends on the parameterization one wants to do. This command let the input file TPQ.cif be transformed into a new file containing now atom types and partial charges. At the same time, while checking the document we find a NT atom, that has to be changed into N by hand in the document using the following command:

Command 3: Changing values by sed
```
    sed 's/NT/ N/' tpq.ac >tpq2.ac
```

The sed command lets change the value from the file that matches with the one put in the code for another one, and create a new file from that change. Having created the new file, it will be used as the new input. The next step is to prepare the library and force field parameters for use with leap. Leap is referred to the generic name is given to the programs teLeap and xaLeap which are used to prepare input for Amber molecular mechanics programs. In this project, only teLeap is used because xaLeap is an enhanced version of teLeap that is not required[22].

The TPQ.cif document downloaded consists in a complete molecule with all of its hydrogen atoms included which is what antechamber command needs to compute the partial charges, but is necessary to strip off some atoms in order to make an aminoacid-like residue that is ready to be connected to other amino acids. In order to do this, is required to create a main chain file for TPQ, that specifies which atoms will be removed and which ones are part of the main chain:

```
HEAD_NAME N
TAIL_NAME C
```

```
MAIN_CHAIN CA
OMIT_NAME H2
OMIT_NAME OXT
OMIT_NAME HXT
PRE_HEAD_TYPE C
POST_TAIL_TYPE N
CHARGE 0.0
```

In this file there are different contents. At first the HEADNAME and the TAILNAME that identifies which atoms will connect to the previous and the following atoms. The MAINCHAIN identifies the atoms along the chain that connect the head and tail atoms. OMITNAMES list the atoms in TPQ residue that should be removed from final structure as they won't be present in the protein. The PREHEAD and POSTTYPE let prepgen know what atoms types in the surrounding protein will be used for covalent connection. At the end there is the CHARGE line which for each residue is put the total charge of it.

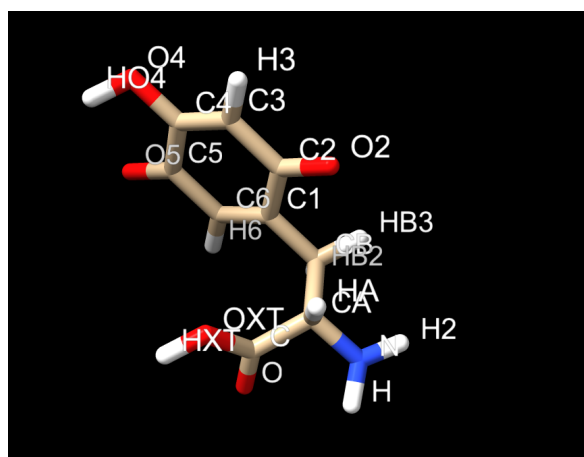Now with the main chain file called tpq.mc, it has to be used with the following command:

Figure S2: Topology and atom names of the TPQ residue used in this calculation.

Command 4: Prepgen command for TPQ

```
prepgen -i tpq2.ac -o tpq.prepin -m tpq.mc -rn TPQ
```

By running that command a file containing the definition of the TPQ modified residue is generated into a prepi file. For now we have the residue library with the charges for the atoms in our modified amino acid; but the covalent parameters like bonds, angles and dihedrals are still needed.

To obtain those parameters the following command has to be run:

Command 5: Parmchk2 command

```
parmchk2 -i tpq.prepin -f prepi -o frcmod.tpq -s gaff2
```

By reading an ac file or a prep file it gives as an input an additional force field file (frcmod file) with the missing parameters. Every atom type can have one or more corresponding atom types for which force field parameters are changeable. Having completed the TPQ parameterization, a similar process will be followed to parameterize NAG. Initially the same first command used for TPQ with a change in the format output file (fo) is runned:

Command 6: Antechamber for NAG

```
antechamber -fi ccif -i NAG.cif -bk \
           NAG -fo prepi -o nag.prepin -c bcc -at amber
```

By this change in the command the output file directly is a prepi file. That change in this case provides an optimization because we are creating two less files since NAG is not a cofactor like TPQ that derives from a tyrosine that has been modified. As it is a carbohydrate, is not needed to create a mc file to transform it in an "amino-acid" like residue by the prepgen file. It is true, that all NAGs are linked to the protein by asparagine but the correctly way to address this linking to amber understand it, is by changing in our protein PDB file called DAO.pdb, look for the ASN (asparagine) that link NAGs to the protein and change that ASN for a NLN (N-linkage to ASN). It can be done manually one by one, or openning the file in Visual Studio Code, look for each ASN that has a N-linkage by simply doing a Control + F and replace them in the little window that appears for NLN. In this case the ASN to change are:

```
ASN A 126
ASN A 553
ASN B 768
ASN B 1195
```

Continuing with NAG parameterization only lacks the frcmod file, that is obtained by using the prepi file as an input and the following command:

Command 7: Parmchk2 command

```
parmchk2 -i nag.prepin -f prepi -o frcmod.nag -s gaff2
```

Now the two files needed from TPQ and NAG are prepared.

After parameterizing each of the two non-standard residues of the protein, is needed to parameterize the carbohydrate, trehalose. This step was a little more complex because the CIF or PDB file of trehalose couldn't be found anywhere, then to obtain the file, the use of Openbabel was required. First of all a file from drugbank in 3D mol format was downloaded. Then, that file was processed and transformed by Openbabel into a 3D coordinates PDB file. At the end we achieved a file that amber software could process by running the command pdb4amber.

To start with trehalose parameterization as in the previous cases first is used Antechamber on the input file:

Command 8: Antechamber for trehalose

```
antechamber -fi pdb -i tre.pdb -fo ac -o tre.ac -c bcc -at amber
```

Having the ac file, as well as previous parameterizations, still two files are required to be loaded in our tleap file, the prepi file and frcmod file. To get the prepi one, first thing to create is a main chain file for later use prepgen command. Considering that trehalose will be isolated, the main chain file only will have a line referring to the charge:

```
CHARGE 0.0
```

With the main chain file created for the isolated trehalose, it is the time to use it as input the prepgen command

Command 9: Trehalose prepgen

```
prepgen -i tre.ac -o tre.prepin -m tre.mc -rn TRE
```

With that command now the prepi file generated is ready to be loaded in the tleap file. For the obtention of frcmod file, the prepi file is needed as an input file, so the command to use:

Table 1: Amber simulations

Simulations to perform
Simulation of DAO in water system without ions, NAG ligand and trehalose
Simulation of DAO in water system with NAG, water, ions and trehalose (0,5M; 1M; 1,5M)
Simulation of 1 trehalose molecule in water system
Simulation of 2 trehalose molecules in water system
Simulation of 5 trehalose molecules in water system
Simulation of 10 trehalose molecules in water system
Simulation of 25 trehalose molecules in water system

Command 10: Trehalose parmchk2

```
parmchk2 -i tre.prepin -f prepi -o frcmod.tre -a Y \
         -p $AMBERHOME/dat/leap/parm/parm10.dat
```

Since one of the objectives is to run five simulations with different number of trehalose molecules, five different files are needed, which for each one of them a different set of number of trehalose molecules where added by rotating and translocating them randomly with a Python program that was created based on Euler angles and randomization of the reorganization of N molecules in space, with a separation between them at a minimum 3Å of distance.

A.3   Tleap files

Having the protein file and the parameterization files ready, they are need to be loaded into a tleap file along with other parameters. Tleap file, once it is run, it gives as an output the topology and coordinates files to use for simulations of the system created. Since different simulations will be performed, varying tleap files will be needed. Each variation of tleap file will be explained with each simulation in the following section.

A.4   Amber simulations

Typically every molecular simulation follows the same step by step workflow, probably adding, removing or changing steps depending on the simulation requirements. In this project the workflow consisted in first of all obtaining the necessary PDB and CIF files for the protein, the components of it and the dissacharyde. After that we prepared the protein file and parameterize the components and the trehalose. Next to do was to generate different tleap files for each simulation. This procedure was performed in a very similar manner between each simulation(Table:1)

A.4.1   DAO without NAG, ions or trehalose

Before anything else, to extract NAG from DAO.pdb because is not going to be used in this simulation, running an easy command in the terminal, deletes every part of the document that contains NAG:

Command 11: Trehalose prepgen

```
grep -v NAG dao.pdb > daonag.pdb
```

For this simulation it is only needed to load the parameters of TPQ since we deleted NAG for this simulation. Ions were not added in this case. The tleap file for this simulation called tleap.in looked like this:

```
source leaprc.protein.ff19SB
source leaprc.water.opc
loadAmberPrep tpq.prepin
```

```
loadAmberParams frcmod.tpq
dao=loadpdb daonag.pdb
bond dao.132.SG dao.153.SG
bond dao.314.SG dao.340.SG
bond dao.774.SG dao.795.SG
bond dao.956.SG dao.982.SG
solvateOct dao OPCBOX 10.0
savepdb dao daowat.pdb
saveAmberParm dao dao.parm7 dao.rst7
quit
```

The first two lines load the force fields we need for the simulation, both of the two following ones are the parameters files we created for TPQ. Next line loads our protein file of interest as a variable and gives it a name, in this occasion dao. Now we have reached the lines that by our explicit needs we specify that between those atoms there is a disulfide bond. This disulfide bonds are between CYX residues (a cysteine involved in a disulfide bridge). Once every file needed was loaded it is time to solvate the system; that's what the solvateOct line is for. In this case we solvate the system in a truncated octahedral water box. Having the system completed we need to save a new PDB file of the system and save its topology (parm7) and coordinates(rst7) in two separate files. At the end of the file we must put simply quit to tell amber to shut off tleap. With the tleap file we have to run this command to execute it:

Command 12: Trehalose prepgen

```
tleap -f tleap.in
```

As a result of the run of that command we achieved three new files that would be used later for running the simulation. The new PDB file contains the system information of our protein inside a truncated octahedral water box (Figure S3) and the other two the initial coordinates and the topology of the system.
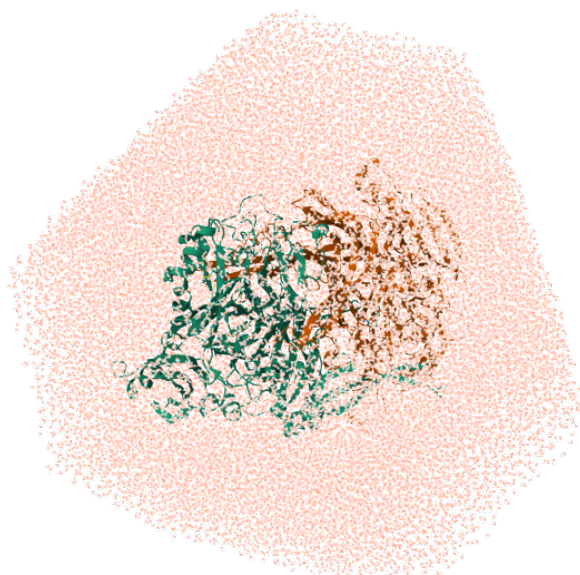

Figure S3: Starting coordinates and structure from the system of DAO without NAG into a truncated octahedral water box as a reference for simulation process (image obtained by 3D PDB viewer)

Given that the simulation is runned in 3 steps: minimization, heating and production, we are going to create three different files to be used as inputs by amber. The first file would be minimization file (called min.in) that must have certain parameters established as well as the other two. Minimization is a numerical process that begins with a higher energy initial structure and finds a minimum on the potential energy surface. In this particular minimization process of this simulation was like that:

```
  simple generalized Born minimization script
&cntrl
   imin=1, ntb=1, maxcyc=100, ntpr=10, cut=8.0, ntmin=1,
 /
```

Once the file is prepared it has to be executed from the terminal, in this case:

Command 13: Trehalose prepgen

```
pmemd -O -i min.in -p dao.parm7 -c dao.rst7 -o min1.out -r min1.rst7
```

Running this command and using the min.in file as an input together with the coordinates and topology file from tleap, we get an output file that register the process of minimization and a restart file with the coordinates and velocities from the last step of minimization. After running minimization comes heating, as it name suggests consists of the progressive increase of temperature of the system very slowly until it reaches the desired temperature. Heating file called heat.in had this structure:

```
Explicit
 &cntrl
   imin=0, irest=0, ntx=1,
   ntpr=1000, ntwx=1000, nstlim=100000,
   dt=0.002, ntt=3, tempi=0,
   temp0=300, gamma_ln=5.0, ig=-1,
   ntp=0, ntc=2, ntf=2, cut=8.0,
   ntb=1, igb=0, ioutfm=1, nmropt=1,
 /
 &wt
   TYPE='TEMP0', ISTEP1=1, ISTEP2=100000,
   VALUE1=10.0, VALUE2=300.0,
 /
 &wt TYPE='END'
 /
```

As in minimization process, the file it has to be executed by a command from the terminal:

Command 14: Heating run

```
pmemd -O -i heat.in -p dao.parm7 -c min1.rst7 -o heat.mdout \
      -x heat.nc -r heat.rst7
```

Using the heat.in as an input file along with the coordinates file from minimization and the topology file, the command returns just as in the previous step an output file with the heating register and a restart file with coordinates and velocities from heating; but in addition it gives an extra file, the heat.nc file which is the one that saves the trajectory of the system during the heating process. Lastly we have to run the largest process in molecular dynamics simulation which is the production. As in the previous processes, we have to create a new file to run this process, for this production the file was named md.in:

```
Explicit md
```

17

```
 &cntrl
   imin=0, irest=1, ntx=5, ntt=3,
   ntpr=1000, ntwx=1000, ntwr=1000, nstlim=500000000,
   dt=0.002, ntt=3, tempi=300,
   temp0=300, gamma_ln=2.0, ig=-1,
   ntp=1, ntc=2, ntf=2, cut=8.0,
   ntb=2, igb=0, ioutfm=1,
 /
```

Repeating the cycle like in the previous two steps of simulation, to use the md.in file with the heating restart file, a code to run in the terminal is needed:

Command 15: Heating run

```
    pmemd -O -i md.in -p dao.parm7 -c heat.rst7 -o md1.mdout \
      -x md1.nc -r md1.rst7
```

Given that this command would take a lot of time to be completed since uses local CPU, the GPU from a supercomputer was required. Logging into a supercomputer by ssh connection, all the files were copied there to work on; in addition a new bash file to run in the supercomputer was created (md.batch) to run the process:

```
#!/bin/bash
#SBATCH -J mddao
#SBATCH -e %J.%j.err
#SBATCH -o %J.%j.out
#SBATCH -p gpu
#SBATCH -n 1
#SBATCH -t 0-24:00
module load apps/amber/22
##
#  Modify the input and output files!
cp -r ${SLURM_SUBMIT_DIR}/{*.in,*.parm7,*.rst7} $SCRATCH
cd $SCRATCH
echo $(SCRATCH)
echo $(SLURM_SUBMIT_DIR)
echo $SCRATCH
ls
pwd
pmemd.cuda -O -i md.in -p dao.parm7 -c heat.rst7 -o \
md1.mdout -x md1.nc -r md1.rst7
cp ./*.out ${SLURM_SUBMIT_DIR}
```

To run this new file in the supercomputer, from the terminal logged by ssh (secure shell) connection we have to run the following command:

Command 16: Submitting job into supercomputer

```
    sbatch md.batch
```

The outputs of this command are the same that in heating step, a registration, a trajectory and a restart output files. Due to the large time it takes to run a production process of a simulation, normally it takes more than one run; in this production 15 runs were needed because the calculations took 24 hours each to be completed. Since the same production process was divided in 15 runs, each time a run finished, its restart output file was used in the following run as an input coordinates file so the process wouldn't be lost.

A.4.2 Trehalose simulations

Before preparing a simulation containing the protein and trehalose in a water system, a first verification of the stability of trehalose will be done. For the stability testing, five different simulations with a different number of trehalose molecules in each of them.
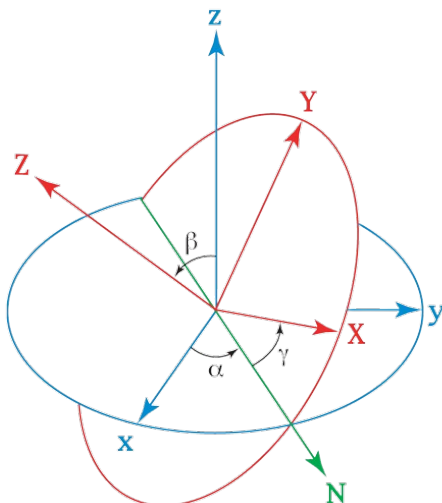

Figure S4: Two orthogonal coordinates system where Euler angles are shown

As mentioned earlier, to generate five different files to be used in each simulation, a Python program was created. The program called "addtobox" was based in different parameters. First of all, it used as an input file the trehalose ac file obtained in the preparation file process which includes the partial charges. The orientation of one typically mobile orthogonal axis reference system with respect to another normally fixed orthogonal axis reference system is specified using a set of three angular coordinates known as Euler angles (Figure S4), so an array of randomized Euler rotation angles is created to use later. In parallel a translation vector is generated. The next thing was to read the ac file coordinates and extract them into an array which then for every molecule number desired would be add in different chains into a new PDB file. Having a file with the coordinates copied according to the number of desired molecules we apply the translation vector and the Euler array generating new rotated and translated coordinates from the original.

The Python program was runned five times generating five different files with different number of molecules (Figure S5).

Once the files were generated they were processed like the protein PDB file. Since we already have parameterization files for trehalose it is not necessary to create new ones, the only difference for each simulation is the tleap file as the other ones can be shared. The difference resides in the PDB file that is load in the tleap, as in the example below, the loadpdb would change for every tleap:

```
source leaprc.protein.ff19SB
source leaprc.water.opc
source leaprc.GLYCAM_06j-1
set default PBRadii mbondi3
loadAmberPrep tre.prepin
loadAmberParams frcmod.tre
tre=loadpdb tre-copy.pdb
solvateBox tre OPCBOX 10.0
savepdb tre tre1hwat.pdb
saveAmberParm tre tre1.parm7 tre1.rst7
quit
```
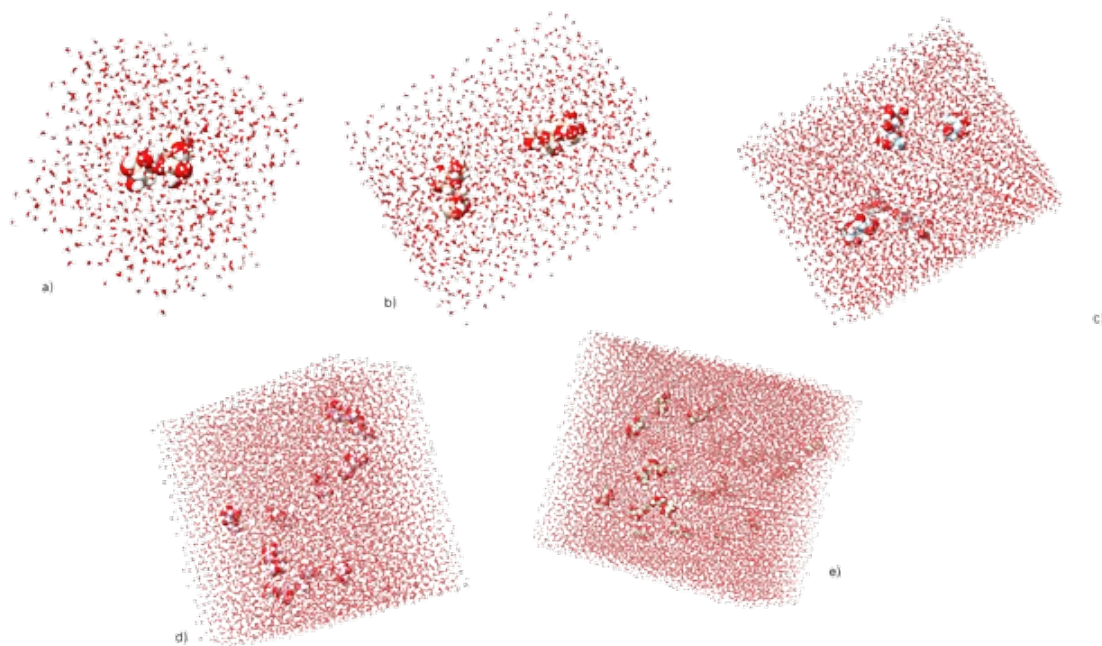
Figure S5: Different cubic water box systems with different number of trehalose molecules inside: a) 1 trehalose molecule, b) 2 trehalose molecules, c) 5 trehalose molecules d) 10 trehalose molecules, e) 25 trehalose molecules. Images got with ChimeraX

For the shared files we have the minimization, heating and production files. Although these files are shared, we have to change in the command it executes the different topology and coordinates files for each one of the simulation.

The minimization file (min.in) used to minimize the system was:

```
     simple generalized Born minimization script
 &cntrl
   imin=1, ntb=1, maxcyc=100, ntpr=10, cut=8.0, ntmin=1,
 /
```

As can be seen, each of the files differ one of each other in some of the parameters, the value of those, the quantity... This is because each step of the simulation requires different parameters to be assigned. Not only the parameters between different steps of simulation are different, but for every simulation a new system of files is needed. The structure of this heating file is this way:

```
    Explicit
 &cntrl
   imin=0, irest=0, ntx=1,
   ntpr=1, ntwx=1000, nstlim=2000,
   dt=0.002, ntt=3, tempi=0,
   temp0=300, gamma_ln=5.0, ig=-1,
   ntp=0, ntc=2, ntf=2, cut=8,
   ntb=1, igb=0, ioutfm=1, nmropt=1,
 /
 &wt
   TYPE='TEMP0', ISTEP1=1, ISTEP2=100000,
   VALUE1=10.0, VALUE2=300.0,
```

```
 /
 &wt TYPE='END'
 /
```

And the production file to run in supercomputer:

```
    #!/bin/bash
#SBATCH -J md1tre
#SBATCH -e %J.%j.err
#SBATCH -o %J.%j.out
#SBATCH -p gpu
#SBATCH -n 1
#SBATCH -t 0-12:00
module load apps/amber/22
##
#  Modify the input and output files!
cp -r ${SLURM_SUBMIT_DIR}/{*.in,*.parm7,*.rst7} $SCRATCH
cd $SCRATCH
echo $(SCRATCH)
echo $(SLURM_SUBMIT_DIR)
echo $SCRATCH
ls
pwd
pmemd.cuda -O -i md.in -p tre1.parm7 -c \
tre1heat.rst7 -o tre1md.mdout -x tre1md.nc -r tre1md.rst7
cp ./*.out ${SLURM_SUBMIT_DIR}
```

A.4.3   DAO with NAG and trehalose simulation

The last simulation to be prepared is the one that contains the protein, all of its cofactors, the carbo-hydrate, ions and the cubic water box completing the system.

It follows basically the same preparation that was used for the first simulation with some changes in it. The first thing that is needed is the PDB file of DAO (code 1KSI from RCSB) prepared for Amber by running the pdb4amber command, but in this case the lines containing NAG are needed so they will not be erased. This simulation, unlike the one consisting of DAO without NAG, requires the addition of new command lines in its tleap file:

```
source leaprc.protein.ff14SB
source leaprc.water.opc
source leaprc.GLYCAM_06j-1
loadAmberPrep tpq.prepin
loadAmberPrep nag.prepin
loadAmberParams frcmod.tpq
loadAmberParams frcmod.nag
dao= loadpdb dao.pdb
bond dao.132.SG dao.153.SG
bond dao.314.SG dao.340.SG
bond dao.774.SG dao.795.SG
bond dao.956.SG dao.982.SG
bond dao.1287.C1 dao.768.ND2
bond dao.1287.O4 dao.1288.C1
bond dao.1292.C1 dao.1195.ND2
bond dao.126.ND2 dao.1285.C1
```

```
bond dao.1285.O4 dao.1286.C1
bond dao.553.ND2 dao.1289.C1
addions dao NA 0
solvateBox dao OPCBOX 10.0
savepdb dao daowat.pdb
saveAmberParm dao dao.parm7 dao.rst7
quit
```

As it can be seen in the tleap file, a new force field has been added (GLYCAM06-j) which is the one for carbohydrates as NAGs and trehalose. Also new bonds have been specified that represent the bonds between NAG-ASN or NAG-NAG. The last two differences comparing to the first simulation is the addition of Sodium (Na) ions to address a charge of 0 in the global system, and at last the solvation of the system into a water cubic box.

Running the file several errors have been shown. The first error shown was that some NAGs did not have an atom type, which should not be the case because the parameterization files were loaded. Looking at the error carefully shows that only the NAGs are linked between them and to an ASN (ASN-NAG-NAG) are the ones presenting an error.

In a first attempt to correct the error, the assembly of a new residue was tried. It consisted of hooking up the sequences of the ASN and NAGs that were linked between them into a new file and process it as it was a PDB or CIF file downloaded from the web. This try was to no avail.

Another attempt to solve the error consisted in changing the residue name from each of those NAGs into 0YB or 4YB (as GLYCAM section in the Amber manual it is shown), being the 0YB the NAG connected to ASN and the 4YB the NAG linked to the other NAG. At first it seemed that the error was corrected, but it was not and another error appeared.

At the end it was decided to erase those NAG and solvate the system anyway and after that, trehalose was added in a concentration of 0.5M. The addition of trehalose was done into the output file from solvation and considering the volume of it to know how many molecules of trehalose were needed to add. To calculate the number of trehalose molecules needed, first is necessary to know the total volume of the system, that was 1611694.133 Å³. From there the calculations are:

$$V = 1,611,694.133\,\text{Å}^3 \cdot \frac{10^{-27}\,\text{litres}}{\text{Å}^3} = 1.611,694,133 \cdot 10^{-21}\,\text{litres}$$

$$0.5\,\text{M} \cdot 1.611,694,133 \cdot 10^{-21}\,\text{litres} = 8.058,470,665 \cdot 10^{-22}\,\text{moles}$$

$$8.058,470,665 \cdot 10^{-22}\,\text{moles} \cdot 6.022 \cdot 10^{23} = 485\,\text{molecules}$$

After this simple calculations and given as a result the needed number of trehalose molecules to add, we use the command called AddToBox as it is shown:

Command 17: AddToBox command

```
$AMBERHOME/bin/AddToBox -c daowat.pdb -a tre.pdb -na \
485 -o daotrewat.pdb -P 20712 -RP 2.0 -RW 3.0 -G 0.2
```

The result of this command is a system with the protein, almost all of its cofactors except for the NAG polymers, the carbohydrate, ions and water (Figure S6).
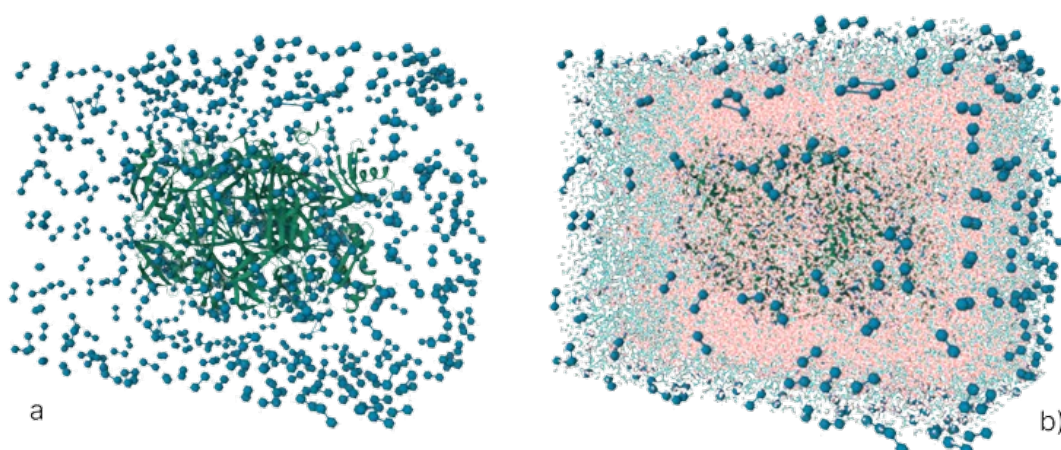
Figure S6: The system composed by the protein, its cofactors, water, carbohydrate and ions: **a)** water molecules not visible, only trehalose (blue molecules) and the protein; **b)** all the system visible

A.5 Results visualization

As the results we transfer from the supercomputer to our local machine are data files with numbers on it, the processing of those files is essential. From the *.mdout files, we get a lot of information, but it cannot be interpreted or plotted. In order to solve this issue, a Python program called dataframes.py was created to perform the function of structuring the data. At the same time another program called graphs.py was created to generate plots as desired like choosing more than one variable to compare or simply the values from only one variable against another one. The *.mdout files are not the only type of file that we get as an output from a simulation, but also trajectory files. These are more complicated to graph because they format are binary, so other program is needed. To initialize the process of graphing these results, first is needed to call cpptraj (once installed) from the terminal in the computer by writing cpptraj on it. For each result the same process was followed, so to take as an example, only one will be explained. Once cpptraj is runned it will appear:

```
CPPTRAJ: Trajectory Analysis. V6.18.1 (AmberTools)

    ___  ___  ___  ___
   | \/ | \/ | \/ |
  _|_/\_|_/\_|_/\_|_

| Date/time: 06/02/24 19:32:04
| Available memory: 397.255 MB


Loading previous history from log 'cpptraj.log'
```

Next thing to do was to load topology and trajectory files:

Command 18: Loading topology file

```
   parm file
```

Command 19: Loading trajectory file

```
   trajin file
```

With the two or more needed files are loaded by a specific command that can be found in AmberTutorials:

Command 20: RMSD command

23

```
   rms ToFirst :1-13&!@H= first out rmsdtre1.agr mass
```

With the cpptraj prepared, the only thing left is to run it by typing the word "run". Once is processed the only thing to do is to visualize the results with xmgrace or other visualization tool:

Command 21: Visualization

```
   xmgrace rmsdtre1.agr
```

This process is repeated for each of the simulations with the amount of trajectory files one wants.
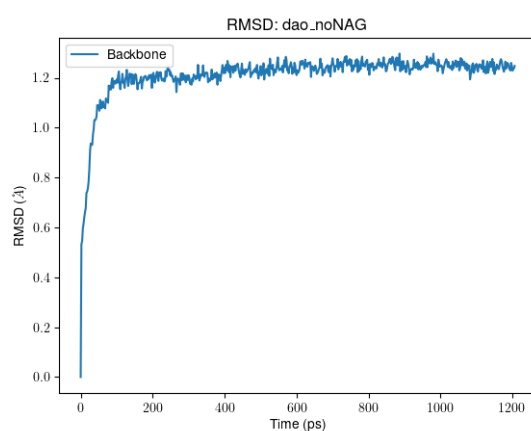
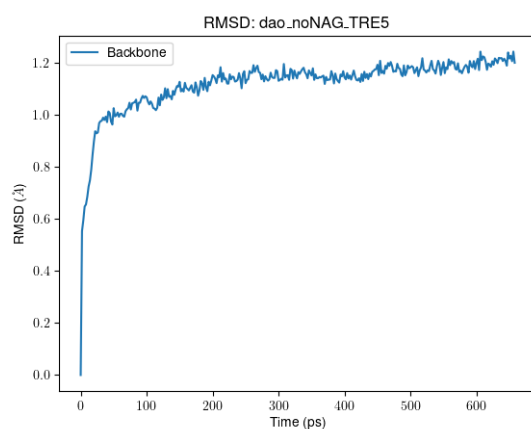## A.6   Raw plots

Figure S7: RMSD for DAO without NAG in water

Figure S8: RMSD for DAO without NAG with 5 trehalose molecules in water
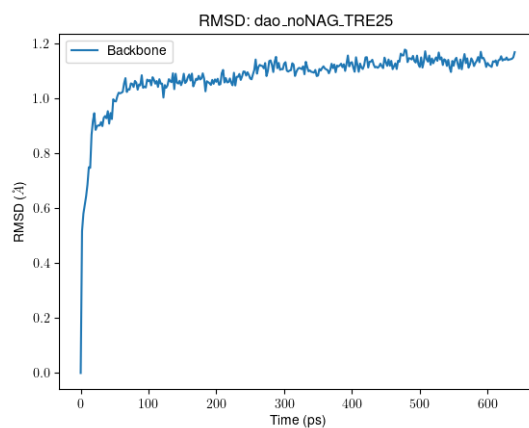
Figure S9: RMSD for DAO without NAG with 25 trehalose molecules in water
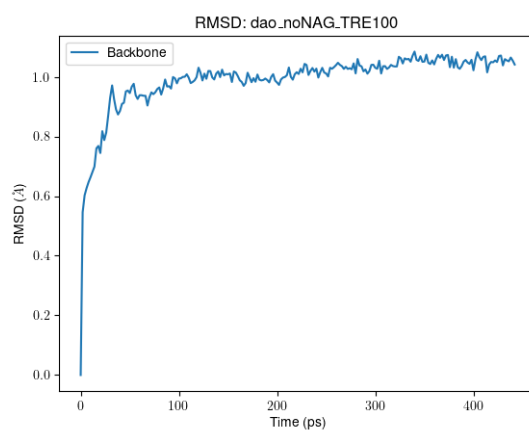

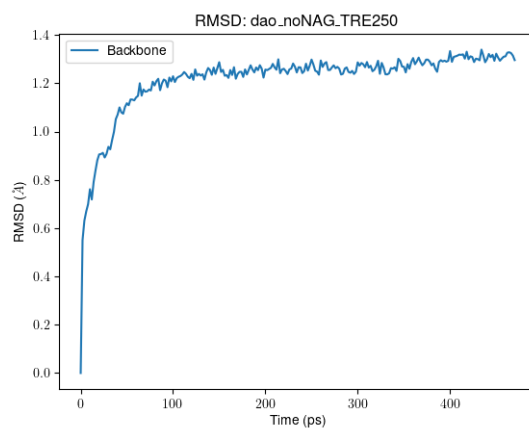**Figure S10:** RMSD for DAO without NAG with 100 trehalose molecules in water

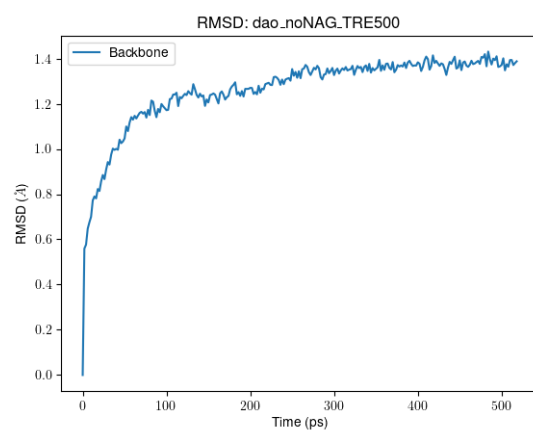**Figure S11:** RMSD for DAO without NAG with 250 trehalose molecules in water

Figure S12: RMSD for DAO without NAG with 500 trehalose molecules in water