



Universität für Bodenkultur Wien

Department für Materialwissenschaften
und Prozesstechnik

Institut für Molekulare Modellierung und Simulation

Betreuer: Prof. Dr. Chris Oostenbrink

**From amino acids to proteins: on the parametrization and
automation of molecular simulation**

Dissertation zur Erlangung des Doktorgrades
an der Universität für Bodenkultur Wien

Eingereicht von
Mag. Christian Margreitter

Wien, August 2016

Danksagung

Es ist knappe vier Jahre her, seit ich mein Doktoratsstudium beginnen durfte. Eine intensive, anstrengende Zeit, voller wertvoller Erfahrungen und ungemein lehrreich. Abseits von wissenschaftlichen Methoden, vermittelt ein Doktorat auch Lektionen für das Leben: Erfolge zu feiern und Rückschläge wegzustecken, Disziplin, Ausdauer und Improvisationsvermögen. Dabei helfen nicht zuletzt jene aus dem wissenschaftlichen und privaten Umfeld, sind Stütze, Rückhalt und Ratgeber. Den Wichtigsten dieser Menschen sei dieser Abschnitt gewidmet.

Von den wissenschaftlichen Wegbegleitern ist selbstverständlich **Chris Oostenbrink** besonders hervorzuheben, den ich als Mentor aber auch als Mensch und persönlichen Freund sehr schätzengelernt habe. Keine Idee, kein Projekt war zu abwegig oder zu ehrgeizig, um nicht trotzdem mit Rat und Tat unterstützt zu werden. Deine bemerkenswerte Auffassungsgabe und dein speziellen Humor werden mir immer in guter Erinnerung bleiben, danke für die schöne und wissenschaftlich so wertvolle Zeit! Zwar beinhalten nur zwei der Projekte in dieser Arbeit Kollaborationen, diese waren jedoch sehr fruchtbar. Zum Einen die Arbeit an den Phosphationen unter der fachkundigen (und geduldigen) Anleitung durch **Maria Reif**, zum Anderen das Antikörperprojekt mit **Renate Kunert**, **Alexander Mader** und **Patrick Mayrhofer**. Speziell die oft stundenlangen Besprechungen mit Patrick waren eine unerschöpfliche Quelle für neue Ideen und eine Gelegenheit, die eigenen Vorstellungen zu prüfen. Bedanken möchte ich mich auch bei der gesamten MMS Laborgruppe, sowohl für die Starthilfe zu Anfang als auch für den regen Austausch während der letzten Jahre. Besonders **Matthias Diem**, dessen Bachelorarbeit ich betreuen durfte und der einige der Projekte, die in dieser Thesis beschrieben sind, fortführen wird, danke ich für die gute Zusammenarbeit und den Spaß, den wir beim Konzipieren und Umsetzen hatten. Abschließend gilt mein Dank Professor **Friedrich Leisch** für seine Unterstützung bei der Entwicklung in R, **Sonja Wit** für ihren heroischen Kampf mit der Bürokratie, **Erik Breslmayr** für die Erstellung des Covers, **Florian Martys** für die generelle Beratung in Designfragen und natürlich **Noor Fakhari** und **Patrick Zahrl** für die hervorragend gelungene Produktion des PROMETHEUS Videos.

Abgesehen von Kollegen, sind die Menschen hinter den Kulissen oft ebenso wichtig für das Gelingen eines Projekts. Egal ob als Motivator, liebevolle Familie oder, ganz profan, als Finanzspritze, ohne **Judith**, **Hubert** und **Georg** hätte ich die letzten Jahre nicht in dieser Form genießen können. Vielen Dank für eure bedingungslose Liebe und unerschütterliches Vertrauen! Für kulinarische Unterstützung und intellektuelle Anregungen, muss ich mich bei meinen Omas **Herlinde** und **Maria**, sowie Onkel **Helmut** bedanken. Es ist mir immer eine Freude mit euch in gemütlicher Runde zu philosophieren. Bleibt noch, mich bei meiner lieben Tante **Margot**, meinen Cousins **Armin** und **Magdalena** sowie **Doris** und **Christoph** herzlich zu bedanken. Ihr macht für mich Familie zu dem, was sie im besten Fall sein soll.

Damit fehlt nur noch eine und zugleich die wichtigste Person in meinem Leben: meine Verlobte **Sophie**. Wir haben in den letzten Jahren gemeinsam zahlreiche Höhen und Tiefen gemeistert, Flexibilität bewiesen, sind Kompromisse füreinander eingegangen. Außerdem ist dein Verdienst an dieser Arbeit sehr konkret: du hast schließlich den Löwenanteil davon getippt, Tabellen ausgefüllt und auf Ansage programmiert (eine besonders "prüfende" Erfahrung für uns beide), damit ich meine lädierten Hände schonen konnte. Normalerweise meint man folgende Phrase eher metaphorisch: "Dies wäre ohne dich nicht möglich gewesen!" In diesem Fall ist es aber ganz wörtlich gemeint. Du bist in jeder Hinsicht für mich da - auch dafür liebe ich dich.

Realität ist das, was übrigbleibt, wenn man aufhört daran zu glauben.

Contents

	Page
1 Introduction	25
1.1 Molecular dynamics (MD)	27
1.1.1 Ensemble averages	28
1.1.2 Integration	29
1.1.3 Potential energy	29
1.1.4 Parametrization	34
1.1.5 Free energy	35
1.2 Proteins	36
1.3 This thesis' contribution	36
1.4 Funding	40
2 Update on phosphate and charged PTMs	41
2.1 Abstract	42
2.2 Introduction	42
2.3 Methods	43
2.3.1 Target values	45
2.3.2 Simulation setup	48
2.3.3 Methodology-independent free energies	51
2.3.4 One step perturbation	52
2.4 Results and discussion	53
2.4.1 Parametrization	53
2.4.2 Newly proposed parameters	54
2.5 Conclusion	57
2.6 Appendix / Supplementary material	58
3 On the optimization of the protein backbone dihedral angles by means of Hamiltonian reweighting	73
3.1 Abstract	74
3.2 Introduction	74
3.3 Methods	75
3.3.1 Model systems	75
3.3.2 Simulation setup	76
3.3.3 Comparison to experimental data	77
3.3.4 Hamiltonian reweighting	79
3.3.5 Ranking	84
3.4 Results and discussion	85
3.5 Conclusion	93
3.6 Appendix / Supplementary material	94
4 Post-translational modifications: effects on the backbone	107
4.1 Abstract	107
4.2 Introduction	107
4.3 Methods	108

4.4	Results	109
4.5	Conclusion	115
5	Anti-body humanization guided by molecular dynamics simulations	121
5.1	Abstract	121
5.2	Introduction	121
5.3	Methods	123
5.3.1	Expression of mAbs	123
5.3.2	Preparation of mAb variants	123
5.3.3	Affinity evaluation of mAb variants	124
5.3.4	In-silico score calculation	126
5.3.5	Molecular dynamics simulations	130
5.3.6	Fitting procedure	130
5.4	Results and Discussion	131
5.5	Conclusion	135
5.6	Appendix / Supplementary material	140
6	PROMETHEUS - automatization of molecular dynamics simulations	145
6.1	Description	148
6.1.1	Applications	148
6.1.2	Antibody simulation support	148
6.1.3	Structure	150
6.1.4	Level of control	152
6.1.5	Used third-party code	152
6.1.6	Workflow and video	152
6.2	Backend	154
6.2.1	Standalone programs	154
6.2.2	Support and internal functions	160
6.2.3	Macros	164
6.2.4	Classes	165
6.3	Frontend	171
6.3.1	Database structure	171
6.3.2	Cluster communication	178
6.3.3	Important functions and classes	178
6.4	Formats	179
6.4.1	Log / error file format	179
6.4.2	Antibody file format	179
6.4.3	Cluster configuration file format	180
6.4.4	Statistical / data file format	181
6.4.5	Instruction file format	181
6.5	Graphical User Interface	187
6.6	Outlook	193
7	MDplot - an R plotting package for molecular dynamics analysis	197
7.1	Abstract	198
7.2	Introduction	198
7.3	Plotting functions	199
7.3.1	clusters()	200
7.3.2	clusters.ts()	201

7.3.3	dssp_summary()	203
7.3.4	dssp_ts()	205
7.3.5	hbond()	207
7.3.6	hbond_ts()	208
7.3.7	noe()	211
7.3.8	ramachandran()	212
7.3.9	rmsd()	215
7.3.10	rmsd_average()	216
7.3.11	rmsf()	218
7.3.12	TIcurve()	220
7.3.13	timeseries()	221
7.3.14	xrmsd()	223
7.4	Additional functions and the bash interface	225
7.5	Conclusions	225
7.6	Acknowledgements	225
7.7	Appendix / Supplementary material	226
7.8	Loading functions	226
7.8.1	load_clusters()	226
7.8.2	load_clusters_ts()	226
7.8.3	load_dssp_summary()	227
7.8.4	load_dssp_ts()	228
7.8.5	load_hbond()	228
7.8.6	load_hbond_ts()	228
7.8.7	load_noe()	229
7.8.8	load_ramachandran()	229
7.8.9	load_rmsd()	230
7.8.10	load_rmsf()	230
7.8.11	load_TIcurve()	231
7.8.12	load_timeseries()	231
7.8.13	load_xrmsd()	232
8	Conclusion	235
9	Addendum	239
9.1	Dihedral parameters in proteins	239
9.1.1	Introduction	239
9.1.2	Methods	240
9.1.3	Results and discussion	241
9.2	Vienna-PTM 2.0	249
9.2.1	Improvements	249
9.2.2	Interface changes	251
9.2.3	Impact	257

List of Figures

- 1.1 Shows, how a system can be addressed at different levels of accuracy, zooming in from right to left. Molecular dynamics simulations are well-suited to deal with the macro-molecular level and to elucidate the physico-chemical foundations. 26
- 1.2 Shows the potential energy for the computational descriptions of covalent bonds (a), angles (b) and dihedral angles / torsions (c). The first two are harmonic potentials, the latter needs to take rotations into account (see equations). 31
- 1.3 The three curves show the attractive and repulsive energies arising from non-bonded interactions. A negative energy (i.e. the combined curve is below zero), means that the non-bonded interactions for a given pair of atoms are attractive. The distance, d , is on the x-axis, the energy on the y-axis. At a large distance (right), two ions with opposite (fixed) charge sign attract each other slightly, giving rise to a force pulling them together. At this distance, the vdW-interactions are negligibly small. When the two ions come closer (middle), not only the vdW-interaction adds to the attractive energy, but also the strength of the electrostatic one increases. Thus, the force becomes stronger. When a certain distance is undercut (when the vdW-radii start to overlap, left), the Pauli repulsion becomes dominant and the unfavourable energy leads to a separation of the ions. 33
- 1.4 The parametrization process starts with the validation of an initial set of parameters against experimental data. Afterwards, an iterative cycle of reparametrization and revalidation runs, until an acceptable deviation is reached. Usually, other factors than just a good match, such as consistency with the rest of the force field are also taken into account. When new experimental data becomes available or problems occur with the parameters in real-world applications or new properties become accessible due to enhanced computational resources, a new round of parametrization might start. 35
- 1.5 Model of an IgG antibody, which has a Y-like shape with two identical parts left and right (left picture). Each part consists of a heavy and a light peptide chain, linked by a disulphide bridge. The fold of the arms is very stable and built out of β -sheets interspersed by highly flexible loop regions. The sheets are called "framework"-regions and the loops, as they facilitate the actual binding, complementarity-determining regions or CDRs. They are shown with their van-der-Waals radii in the right picture. The CDRs are strongly mutated in a process called *antibody maturation* in which they adapt to a given epitope. 37
- 2.1 Graphical illustration of the contributions to ΔG_{solv} . For a detailed description of the contributions see the main text. 44

2.2	Alchemical changes from methyl-phosphate, in the neutral and charged (-1 e) states into the respective phosphate compounds. The free energy of mutation of the neutral species has not been changed while the charged ΔG_{mut} has been adapted to get a cycle closure within $0.6 \frac{\text{kJ}}{\text{mol}}$. The horizontal free energies have been calculated from experimental pK_a values, as described in the main text.	45
2.3	Example thermodynamic integration (TI) graph, as obtained for the cavity formation simulation of dihydrogen phosphate in water. This plot has been made by usage of the MDplot R package.	47
3.1	Graphical representation of the two backbone dihedral angles in the model system. The angles ϕ and ψ are defined by atoms C-N-C $_{\alpha}$ -C and N-C $_{\alpha}$ -C-N, respectively. Note, that the amino acids have been (in accordance to the experimental studies) blocked: an acetyl-group at the N-terminus and a methyl moiety at the C-terminus are used to ensure non-charged ends. . .	75
3.2	Workflow applied for Hamiltonian reweighting. From the configurations sampled in the simulations, the ensemble averages of dependent observables have been predicted. For the calculations of the J-values and Hamiltonians see equations 3.1 and 3.3, respectively. The considerations regarding the ranking of solutions is further described in the methods section of the paper.	80
3.3	Final deviations between simulated and experimental data using 54A7 parameters (a) and our suggested set (b). White bars indicate the deviation of the J-value (left axis) and shaded bars indicate deviations in the secondary structure propensities (right axis). Dotted lines indicate the subgroups used in the optimization.	85
3.4	Ramachandran plot of glycine and proline with different parameters. As is shown in (a), 54A7 fails to reproduce a distribution that is in agreement with literature (see reference [30]), while the distribution for #81883 in (b) seems to be in better agreement. Moreover, proline in 54A7 shows an unexpected peak at a ϕ and ψ of approximately -107° and -65° , respectively (c). For our combination #5623, which is suggested for the common amino acids, we observe (d) a significant shift towards the P $_{II}$ conformation, leaving only a minor fraction in the helical basin.	89
3.5	Potential-energy terms of table 3.1. The first row represents the suggested parameters for glycine, the second for alanine, the third for the common amino acids' subset and the last for the C $_{\beta}$ -branched amino acids, respectively. The left column shows the ϕ , the right the ψ angle.	90
3.6	Backbone energy surface for Ac-A-NHMe (using parameter combination #12572) when rotating the ϕ - and ψ angles. In (a) and (b), the energy surface arising from the non-bonded interactions only is shown in 3D and 2D representations, respectively. It is clear, that the addition of the dihedral angle potentials contributes only by a limited, but still crucial, amount as shown in (c) and (d). This modulation of the energy landscape particularly leads to a P $_{II}$ separation and forms a small α -helical basin, which is populated by 12.7% (estimated by simulation).	91

- 3.7 Average root-mean-square deviation (RMSD) with respect to the original structure for four independent simulations of HEWL using parameter set 54A8 (upper panel) and 54A8 with our suggested dihedral parameters (lower panel), respectively. The RMSD has been calculated based on the backbone atoms C, N and O exclusively. Both parameter sets show a stable structure over 50 ns. The plot has been generated using the R package MDplot, the black curve denotes the mean and the grey ones the respective minimum and maximum values at every time point. 92
- 3.8 Shows the accuracy of the projections based on the 54A7 trajectories for alanine for three different sets of backbone dihedral potentials. Note, that the projection is quite accurate even when different trends (e.g. propensities of #76630 versus those of #6546) are observed. 99
- 3.9 Example of scatter plot showing the deviation in J-value reproduction (x-axis) versus the summed up deviation in propensities (y-axis) for a selection of combinations from our screening set. In principle all combinations in this scatter plot could lead to reasonable agreement with experimental data. The combination #5623, which has been proposed for the description of the "common" amino acids (see main manuscript), is marked with a green dashed circle. 100
- 3.10 Prediction accuracy for the 15x15 bin Ramachandran plots on the example of serine. For combination #57468, which produced very good results in terms of matching the experimental results, a very narrow distribution within the basins is observed (a). Combination #5623 performs only slightly worse in the agreement with the experimental data, but leads to a wider sampling (b). The reliability of the prediction is shown by comparison to actual simulation data in panel (c). In (d), the simulated result with a higher resolution is shown. 101
- 3.11 Correlation plot of experimental J-values with the calculated ones, based on 54A7 (black), our suggested set (red) and the individually optimized (blue) parameters. Since alanine is a strong outlier for 54A7, we also report the correlation coefficient for the remaining residues only which leads to a slight improvement (see second value). The slopes are 0.380, 0.397, 0.789 and 0.957, and the intercepts are 4.025, 3.952, 1.331 and 0.289 for 54A7, 54A7 without alanine, our suggested set and the individually optimized set, respectively. The markers for Ala and Gly are positioned identically in the proposed and the individually optimized sets. 102
- 4.1 Shows the mean effect of the post-translational modifications as compared to their precursors (both using the updated backbone parameters, either #5623 or #86516). The group using #5623 amounts to 41, the one using #86516 to 4 members, respectively. Note, that for L3H and N3H the modifications lead to a change in the used backbone parameters because of an addition to their C_β atoms. 114

- 5.1 Protein A fishing from crude culture supernatants by bio-layer interferometry. The ForteBio Octet system was equipped with protein A biosensors to immobilize transiently expressed humanized Ab2/3H6 mutants from concentrated and crude culture supernatants.
(A) Real-time sensorgram of wt3H6, su3H6 and BM07 at different concentrations. Assay-step times were as follows: 60-s baseline in kinetics buffer, 1. Fishing: 1200-s immobilization of antibodies from crude culture supernatants, 2. Blocking with 100 µg/ml purified mAb su3H6 for 1200 s, followed by 120-s baseline/washing in kinetics buffer, 3. Binding measurements: 600-s mAb 2 F5 association (100 µg/ml) with immobilized Ab2/3H6 variants, followed by 1200-s dissociation in kinetics buffer only. **(B)** Association and dissociation curves extracted from raw data and aligned to baseline by the fortebio software. 125
- 5.2 Workflow of the simulation assisted humanization approach. In the training step (above), molecular dynamics simulations of the murine derived wild-type and selected mutant variants are performed and assessed in terms of a score (see equation 5.1), representing similarity to the wild-type loop conformations. The same set is expressed, and affinities are measured experimentally, allowing for the identification of the qualitative boundary separating binders from non-binders (the latter marked with a red asterisk). The second step (below) starts with the superhumanized antibody variant, and a set with selected backmutations. With the same procedure as before, a computational score can be calculated holding qualitative information on the expected binding affinities. The cutoff determined in the previous step can now be used to classify the results. Note, that this is only an illustrative diagram; the actual values obtained throughout this study are reported in figures 5.5 and 5.6 and table 5.1. 127
- 5.3 Sequence alignment of the heavy (A) and light (B) chain of the wild-type (wt3H6), the training (TR01-TR06) and the superhumanized (su3H6) variant. The CDR regions as defined by Kabat were conserved during the humanization process and kept constant in all variants throughout this study (blue background). Identical amino acids compared to the murine/wild-type template sequence are indicated as dots. Vernier zone residues of the heavy chain defined by Foote and Winter are marked by green frames. . . . 128
- 5.4 Sequence alignment of the heavy chain of the wild-type (wt3H6), the superhumanized (su3H6) and the simulated (predictive, BM01-BM11) variants. The CDR regions as defined by Kabat were conserved during the humanization process and kept constant in all variants (but BM10) throughout this study (blue background). Identical amino acids compared to the murine template sequence are indicated as dots. Vernier zone residues of the heavy chain defined by Foote and Winter are marked by green frames. No light chain mutations in respect to the superhumanized variant were applied to the backmutation variants. 129
- 5.5 Similarity scores and free energies of binding for the variants. For the calculation of the score, see equation 5.1. The grey bars indicate a score for the similarity to the murine/wild-type 3H6 antibody (left axis), while the white bars indicate experimentally determined binding free energies (right axis). For TR01, the simulation score and the experimentally determined binding free energy are clearly disagreeing, see main text. 131

5.6	Real-time bio-layer interferometry sensorgrams for determination of anti-idiotypic binding affinities of purified Ab2/3H6 variants to its target antibody mAb 2F5. Streptavidin biosensors were loaded with biotinylated mAb 2 F5 (20 µg/ml) followed by a washing/baseline step. Association (600 s) and dissociation (1200 s) of different Ab2/3H6 variants were measured at different concentrations or buffer only, respectively.	132
5.7	Similarity scores and free energies of binding for the wild-type, the superhumanized antibody and BM07. The experimental binding free energy of su3H6 was below the detection limit.	134
5.8	The importance of R98 is likely to be explained by its interaction with the surrounding tyrosines through cation-pi interactions, most notably with one located in CDR loop three (Y111). Therefore, it is of utmost importance that this position remains an arginine in order to retain binding affinity. The "tyrosine cage" at that position could, in conjunction with R98, lead to a more restricted local environment for the CDR loop, which is shown in red.	135
5.9	The average backbone atom root-mean-square deviation (RMSD) with respect to the crystal structure over the last 20 ns of all replicates for the training set. The RMSD is calculated for all backbone atoms of the framework regions after a least-squares fit on these atoms. The error bars indicate the standard deviation.	141
5.10	The average backbone atom root-mean-square deviation (RMSD) with respect to the crystal structure over the last 20 ns of all replicates for the superhumanized variants. The RMSD is calculated for all backbone atoms of the framework regions after a least-squares fit on these atoms. The error bars indicate the standard deviation.	142
5.11	β -strand average occurrences of conformations β over the last 40 ns over all replicates for the training set as determined by the program DSSP for the framework and complementary-determining regions (CDRs; in green). The classification of the crystal structure is shown in black dots (either 100% or 0%), which agrees well with the mean values of the trajectories. This indicates a stable secondary fold throughout the simulation.	143
6.1	Graphical explanation of the MAMA addressing system. The framework regions are shown in orange, the loops (i.e. the complementarity-determining regions or CDRs) in green.	149
6.2	PROMETHEUS consists of two distinct parts. The first is the frontend, serving as the organisational unit and the graphical user interface. It is based on a central database and also facilitates the necessary cluster communication. The backend, the other part, represents the executing unit. The central entity is the <code>executor</code> , which processes the respective instruction files. A set of other specialised binaries completes the backend (described in section 6.2.1).	151

- 6.3 Shows the root template directory at the top, one example folder ("template_TEP") in the middle and the content of the equilibration folder at the bottom. Of course, the content of each folder is arbitrary: all of it will be copied to the jobdirectory upon instantiation. The folder and file permissions necessary will be set appropriately for the user PROMETHEUS is running as. The values of the variables in the input files can be overwritten by the appropriate functions during the course of the execution, see section 6.2.2. 153
- 6.4 Screenshot of the user interface available for pdb2abf, which allows to set antibody-specific settings. Apart from disulphide bridges, MAMA-ranges can also be set which allow a relative addressing of the respective antibody regions. Other convenience functions allow to truncate, rename and remove protein chains. 156
- 6.5 UML 2.0 diagram of the top input / output (IO) classes (see also figure 6.6). The database communication from the backend-side has been implemented but is currently not in use. 166
- 6.6 UML 2.0 diagram of the top IO classes, using XML formatting (see also figure 6.5). All file formats defined by PROMETHEUS use a XML formatting. Examples and explanations for the respective formats, can be found in section 6.4. Note, that the ioLOGfile class is also used for the error file. All classes implement a main parsing (extract()), a XML-constructing (inject()) and several minor parsing functions. 167
- 6.7 UML 2.0 diagram of the IO classes, which load, write and parse GROMOS file formats. There are multiple convenience functions available to access a certain data field easily. 168
- 6.8 UML 2.0 diagram of the GROMOS parsing classes. Every block consists of entities, that in term hold the actual items (the variable names and associated values). The order and number of expected values is hard-coded in special header files. 169
- 6.9 UML 2.0 diagram of the internal protein and antibody cluster. Proteins consist of chains, consisting of residues, consisting of atoms. In addition, disulphide bridges can be specified, if the protein happens to be an antibody, its native class (inheriting from the protein) should be used, which offers additional information storage required in this case. For both the protein and the antibody class, PDB and ABF file streams are available. . 170
- 6.10 Shows the template overview for a certain user. Each of them has a name along with a short description and a timestamp. Templates are stored in the frontends' MySQL database and can be viewed in either the editor (see figure 6.16 for an example) or in the specialised graphical user interface shown in figure 6.11. They are either private (for the users' use only) or public, in which case it is read-accessible by all users. 187
- 6.11 The template editor allows to generate and maintain templates without the need of manipulating the source code directly. In the top section, meta-information such as the template description is monitored, followed by the header including variable definitions and cluster communication information. The main body represents the processing part, where the top-level steps are shown on the left according to the number of their respective sub-steps. Every top-level step can be manipulated on the right hand side. 188

- 6.12 Detailed view on the subjobs and the first task constituting a metajob. The subjobs are logical entities, typically different compounds. They can be batch-processed using the symbol in the upper right corner (the three arrows), which instantiates a selected template for all of them and forms processes (representing the actual execution). These processes are grouped into tasks, such as "*simulation*", "*secondary structure*" or "*plotting*" depending on the selected template. The processes can be started by use of the rocket symbol. The status column shows the progress of the process in actual steps performed and percent, respectively. By clicking on "manage files", the filebrowser (see figure 6.17) of the respective process is accessed. The user can either select all tasks at a time (which might impose some delay for large metajobs, due to the parsing of many instruction files) or select a single task at a time by a drop-down menu. 189
- 6.13 Shows the header of the frontend, together with the menu and the jobs running on the cluster. The ones associated with the user currently logged in are highlighted in turquoise. This output is produced by the respective LUCI program. 190
- 6.14 List of metajobs belonging to the user currently logged in. The most important information is the root directory, from which the subjob, task and process folders are derived. The metajobs can be accessed by clicking the eye symbol. Each of them has a three-letter series' name in this excerpt, but it can be chosen freely. 190
- 6.15 Excerpt of the administration panel showing the currently defined groups of users and the upper part of the list of permissions. For every user group, every permission is set either to be granted or prohibited. A list of all implemented permissions is available in table 6.5. 191
- 6.16 The file editor allows to directly manipulate templates, instruction files etc. in the browser window. It is based on CodeMirror [11] and CodeMirrorUI [12] and can be easily extended if necessary in further developments. For certain file types, special features are available e.g. text highlighting of XML files, syntax checks for instruction and cluster files and more. Depending on the source of the file, either mode "database" or "file" is used. An alternative for directly manipulating the source code of templates is the template GUI described above. 191
- 6.17 File browser showing the contents of a sub-folder regarding a process. Files and folders can be created, renamed, deleted, uploaded, downloaded and packed into a zipped archive. For certain file extensions (png, ins, ccf, ...) special menus are available by right-clicking onto them. This allows e.g. to reset instruction files (set the done attribute in steps to *false*), to check cluster configuration files for their validity or to display image files in the browser directly (see figure 6.18). 192
- 6.18 Figures such as a RMSD plot can be generated (and displayed) from calculated timeseries directly in the file browser. This is especially powerful if used in conjunction with MDplot (see chapter 7). 192
- 7.1 Shows the overall workflow typically applied in molecular dynamics simulations beginning with a single PDB structure as the input for a molecular dynamics simulation and ending with the graphical representation of the data obtained. For large amounts of data, generating figures might become a tedious, highly repetitive task. 199

7.2	The clusters are plotted along the x-axis and the number of configurations for each trajectory for every cluster on the y-axis. The number of clusters is limited in this example to nine, which is useful as many scarcely populated clusters can be omitted.	200
7.3	The plot shows a selection of the seven most populated clusters for six trajectories. The width of the bars in the top sub-plot is adjusted according to their number.	202
7.4	Example with <code>plotType</code> set to "dots" (default), "curves" or "bars". Note that the fractions do not necessarily sum up to a hundred percent, because some residues might not be in defined secondary structure elements all the time. In this figure, there is no legend plotted due to space limitations (see figure 7.5 for a colour-code explanation).	203
7.5	Example showing all of the defined secondary structure elements per residue over time. Note, that for this example plot a sparse data set was used to reduce the size of the example data file.	206
7.6	The acceptor residues are plotted on the x-axis whilst the donors are shown on the y-axis. The different colours indicate the occurrences throughout the whole trajectory.	207
7.7	Example figure generated by <code>hbond.ts()</code> for both an identifier and acceptor residues' selection. The labels for the hydrogen bonds may be printed as identifiers or with names composed of residue names (in single- or three-letter code) and the names of the participating atoms.	209
7.8	Example plot showing two different replicates of a protein simulation (they share the molecule, but have different initial velocities). Note, that the maximum value (x-axis) over all replicates is used for the plot. The sum over all violations from left to right is shown by an additional curve on top. The number of violations may be given as fractions (in %), as shown above, or absolute numbers (flag <code>printPercentages</code> either <code>TRUE</code> or <code>FALSE</code>).	211
7.9	Two dimensional plot version "sparse" of the <code>ramachandran()</code> function with enabled contour plotting. The number of bins in which the dihedrals are grouped can be specified independently.	213
7.10	Three dimensional example of <code>ramachandran()</code> . In addition to the colour, the height (z-axis) also represents the number of dihedrals per bin.	213
7.11	This plot shows the RMSD curves for two different trajectories. The time is given in nanoseconds, which requires a properly set <code>factor</code> parameter.	215
7.12	In black, the mean RMSD value at a given timepoint is given and in grey the respective minimum and maximum values. In this example, two rather similar curves have been used.	217
7.13	Plot showing two different RMSF curves.	219
7.14	A forward and backward thermodynamic integration curve with the resulting hysteresis between them (precision as permitted by the error).	221
7.15	Shows a timeseries with parameter <code>snapshotsPerTimeInterval</code> set in such a way, that the proper time in nanoseconds is plotted. In addition, the legend has been moved to the top-right position.	222
7.16	An example <code>xrmsd()</code> plot showing only the upper half because of the mirroring of the values.	224

- 9.1 Average backbone atom root-mean-square-deviation plot for the four replicates of the lysozyme protein simulations using the 54A8 parameter set. The grey curves show the maximum and minimum values for the four runs at every snapshot (i.e. the spread) and the black curve the mean values. 242
- 9.2 Average backbone atom root-mean-square-deviation plot for the four replicates of the lysozyme protein simulations using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. Note, that unlike the curves in figure 9.1, there is no increase in the RMSD at the end of the trajectories. 242
- 9.3 Distribution of Nuclear Overhauser Effect (NOE) violations for lysozyme using the 54A8 parameters. The curves on top of the bars, are the sum of the violations. 243
- 9.4 Nuclear Overhauser Effect (NOE) violations for lysozyme using our suggested parameters. 243
- 9.5 Average backbone atom root-mean-square-deviation plot for three replicates of the GCN4 trigger protein using the 54A8 parameter set. 244
- 9.6 Average backbone atom root-mean-square-deviation plot for three replicates of the GCN4 trigger protein using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. The spread is comparable to the one in figure 9.5, both sets of simulations show a strong increase in the RMSD values (most likely from the flexible residues at the termini). 244
- 9.7 DSSP summary plot over the whole trajectory for the GCN4 trigger peptide simulations using the 54A8 parameter set. While the first two simulations are quite similar, the secondary structure propensities of the third are rather different, especially in the helix-type of the right-handed helical core region.
- Legend: ● ... 3-Helix, ● ... 4-Helix, ● ... 5-Helix, ● ... Turn, ● ... β -Strand, ● ... β -Bridge, ● ... Bend. 245
- 9.8 DSSP summary plot over the whole trajectory for the GCN4 trigger peptide simulations using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. Compared to figure 9.7, there is a shift in the helical populations.
- Legend: ● ... 3-Helix, ● ... 4-Helix, ● ... 5-Helix, ● ... Turn, ● ... β -Strand, ● ... β -Bridge, ● ... Bend. 246
- 9.1 The new front page. There is, in addition to the top bar, a second navigation menu available on the left hand side (not shown). The upload form is at the bottom (not shown) and allows to upload PDB files from the users' hard drive as well as to give a PDB identifier in which case the denoted file is retrieved automatically. 252
- 9.2 A screenshot of the old homepage (version 1.0) as described in the original publication [1]. A major criticism was the representation of the modifications as the actual process, the names of the resulting post-translationally modified amino acids were not shown. The new interface provides more information on the result. 253

- 9.3 New implementation of the modification menu. The jpeg-based "pearlson-a-string" have been replaced by CSS-enhanced buttons. This allows to easily integrate new modifications into the webserver. By clicking on them, the overlay-menu shown in figure 9.4 opens. 254
- 9.4 The modification menu for a selected amino acid (serine 56 in figure 9.3). By using the drop-down menu, post-translational modifications can be selected, which are graphically shown at the right hand side. The menu is different for every type of amino acid. The original residue, the Vienna-PTM internal modification code (see also the modifications listed on the homepage), the new residues' three-letter-code and (if available) the respective ChemSpider and Pubchem identifiers and their links are shown in the left-bottom part. 255
- 9.5 The selected changes are represented in the change of the abbreviation of the respective button (residue name) and a color code, explained at the bottom. The colors are selected according to the resulting compound, not the precursor (canonical amino acids are shown in gray). Clicking on "Process" starts the modification (and minimization, if selected) and forwards to the results page. 256
- 9.6 The IP-addresses from which jobs have been submitted, are widely spread over the world. However, "geolocation" is not perfectly accurate and sometimes leads to slightly shifted marks. 257

List of Tables

2.1	Collection of standard state dihydrogen phosphate hydration free energies from the literature.	45
2.2	Proposed partial charges of the four phosphate-containing moieties parametrized in this study. Dihydrogen phosphate has been parametrized first, targeting an experimental hydration free energy 2.1. The other ions have been parametrized relative to dihydrogen phosphate. The van-der-Waals interactions as well as the bonded interactions have not been changed from the 54A7 parameters.	49
2.3	List of free energy contributions to the standard state solvation free energy (equation 2.10) for the dihydrogen phosphate simulation using the new parameters. For a description of the individual terms, see the main text. .	54
2.4	List of free energy changes (in $\frac{\text{kJ}}{\text{mol}}$) calculated from the three thermodynamic cycles used, together with the (absolute) cycle closure deviation. The first and last values of each series have been calculated from experimental pK_a values. The thermodynamic cycle used for methyl-phosphate is depicted as an example in figure 2.2. In the first column, X represents one of the phosphate derivatives, while P represents dihydrogen phosphate.	55
2.5	List of post-translationally modified amino acids that have been updated in the context of this study. The phosphate parameters have been taken from the parametrization described in this work, while parameters for the other groups have been proposed in the publication of Reif et al. [3]. The nomenclature of the three-letter abbreviations follows the one previously described by Petrov et al. [6] in 2013.	56
2.6	List of free energy contributions to the standard state solvation free energy (equation 2.10) for the dihydrogen phosphate simulation using the new parameters. For a description of the individual terms, see the main text. .	58
2.7	List of parameters for the protein post-translational modifications reparametrized in this study.	59
3.1	Backbone parameters of the GROMOS force field and those of the suggested set. All other combinations mentioned in the text are provided in table S3.5.	76
3.2	All combinations used for the reweighting workflow. Every angle is described by either one or two potential-energy functions. The possible values for the parameters of equation 3.3 are given in brackets. We accounted for the possibility that one potential energy term might suffice by adding a force constant of zero to the second potential energy terms. The total number of unique combinations sums up to 97344. This number consists of $(3^2 * 2^2 * 4^2)/2 = 288$ combinations with two potential energy terms and $(3^1 * 2^1 * 4^1) = 24$ combinations with one potential energy term summing up to 312 for each of the angles. Combining these parameter combination for both the ϕ and ψ angles leads to $312^2 = 97344$ combinations in our set.	79

3.3	Detailed results of the predicted and simulated parameter combinations. In brackets, the deviation to the experimental target values is reported: negative values indicate that the computational values are too high and vice versa for positive ones. The combination given in italic letters indicates the best hit when optimizing the amino acids individually and thus gives the best hit possible with our screening set of parameters.	82
3.4	Summary of the averaged performance of the different sets in terms of agreement with experimental data. The absolute values of the individual deviation have been averaged and are reported together with their standard deviation. For the propensities, the renormalized data is reported.	86
3.5	Combinations for the individually optimized amino acids.	94
3.6	Parameters used for blocking for the Ac-X-NHMe simulations.	95
3.7	List of the target values extracted from the studies of Avbelj et al. [2] and Grdadolnik et al. [3]. The propensities have been retrieved from Raman spectroscopy data calibrated by the J-values of the former study.	96
3.8	Listing of the dihedral angle areas used for classification. This table holds the basins [17], which were used for the classification with DISICL [17, 18] for all compounds investigated. The original values were reported by Hollingsworth et al. [19]. A structure with a pair of ϕ/ψ backbone dihedrals, that falls within one of these regions is assigned to belong to the secondary structure element given in the first column. All remaining structures are considered to be unclassified. The experimental comparison was based on helical_R , β and P_{II}	97
3.9	Comparison of dynamical parameters of different combinations. On the left of each column, the number of visits per basin is reported, while on the right the average residence time per basin is given in picoseconds. Some combinations in our suggested set increase the torsional energy barriers (especially in the ϕ -angle, see figure 3.5), which in turn leads to higher residence times. Still, a sufficient number of transitions between the basins is observed. All values are reported for trajectories of 100 ns length.	98
4.1	J-values (in Hz) and secondary structure propensities (in %/100) obtained for commonly found post-translational modifications and their canonical amino-acid precursors. Both the values using the 54A7 backbone parameters and the ones reported in chapter 3 as well as versions of S1P and T1P using the old phosphate parameters and the ones derived in chapter 2, are reported. The values in brackets denote the change in respect to the 54A7 parameters.	110
5.1	<i>In-silico</i> score predictions for all simulated variants based on the (non-binding) su3H6 antibody.	133
5.2	Properties of amino acid side-chains selected for establishing wt3H6 double mutants used as a training panel (TR01-06) for MD simulations. The critical functions of these framework positions are described in Mader and Kunert, Protein Eng. Des. Sel. PEDS, 23, 947 - 954 (2010). Spatial positions in the crystal structure 3BQU are summarized.	140

5.3	Conserved amino acid residues at the position equivalent to R98 in the human IGHV germline genes (IMGT/Gene-DB). Search parameters: homo sapiens (species), variable (gene type), functional (functionality), IGH (locus) and IG (molecular component).	
		141
6.1	Table of backend programs, together with their parameter lists and short descriptions.	157
6.2	List of support functions used in the backend programs. Some are of general use (namespace: <code>supfunc</code>), others implement functionality when the GROMOS simulation package is used (namespace: <code>GROMOS</code>). These functions are available from within the programs, not by calling them (see <code>internal</code> functions for comparison).	160
6.3	Internal functions provided by the backend they offer general tasks, such as the evaluation of a regular expression. For an example, how to call them, see the format definition in listing 6.18.	161
6.4	List of internal GROMOS functions provided by the program executor. These functions add necessary parts in the execution of a GROMOS based simulation and enhance the convenience significantly.	162
6.5	Table of the currently implemented permissions, used to restrict access to certain functions in the frontend. New permissions can be added to the database easily and readily applied in further development.	173
6.6	List of exemplary frontend functions, together with their parameters, return values and a short description.	178
7.1	Lists all of the currently available plotting functions that have been implemented in <code>MDplot</code> . All functions accept a boolean parameter (<code>barePlot</code>), that allows to print the plotting area only.	199

Abstract

Molecular dynamics simulations have evolved into a broadly used and reliable tool in recent decades. The key to this progress are both increasingly powerful computers and the higher accuracy of our computational models, which are constantly improved. This thesis' first goal is to support the latter developments by working on a better description of phosphate moieties and the amino acid backbone torsions in the context of GROMOS parameter sets. Phosphates are pivotal in the nucleic acids' backbone, lipids and as a major group of protein post-translational modifications. The backbone torsions on the other hand are utilized in every simulation using peptides and proteins and therefore of high importance. The second aim was to link simulations and experiments of antibodies by the establishment of an *in-silico* binding prediction approach. Experimentally verified, it allows to get a qualitative forecast on the antibodies' binding behaviour from simulations without the hassle to calculate full binding free energies. This type of large projects made an automated workflow, simulating multiple candidates automatically, highly advantageous. We have therefore implemented the molecular dynamics workflow manager PROMETHEUS. It allows to define a workflow logic for a distinct project and to subsequently perform the execution completely automated an arbitrary number of times for all subjobs part of the project. Furthermore, in order to readily see the results afterwards and to provide the user with publishable figures, the R package MDplot has been developed, which is able to parse and plot the simulation data directly. This work tries to push the boundaries of molecular dynamics simulations in terms of accuracy and applicability further, by provision of appropriate computational tools and force field parameters.

Zusammenfassung

Molekulardynamiksimulationen haben sich über die letzten Jahrzehnte zu einem weit verbreiteten und zuverlässigen Werkzeug entwickelt. Der Schlüssel dazu war sowohl die ständige Zunahme der Leistungsfähigkeit von Computern, als auch die Genauigkeit der ständig verbesserten Computermodelle. Das erste Ziel dieser Arbeit ist die weitere Verbesserung dieser Modelle durch die Reparametrisierung von Phosphaten und der Rückgratdihedrale von Aminosäuren in GROMOS Kraftfeldern. Phosphate sind nicht nur wichtig im Rückgrat von Nukleinsäuren und in Lipiden, sondern stellen auch eine der bedeutendsten Gruppen von post-translationalen Modifikationen in Proteinen dar, während die erwähnten Torsionspotentiale in jeder Peptid- oder Proteinsimulation verwendet werden und damit zu den am Häufigsten verwendeten Parametern zählen. Das zweite Ziel war die Herstellung einer Verbindung zwischen der experimentellen und theoretischen Analyse der Bindungseigenschaften von Antikörpern. Die hierfür entwickelte Methode erlaubt eine qualitative Vorhersage des Bindungsverhaltens durch Simulationen, ohne jedoch auf die schwierige Berechnung der freien Energie dieses Prozesses angewiesen zu sein. Um diese grossen Projekte standardisiert und automatisiert durchführen zu können, war die Entwicklung eines Prozessmanagers für automatisierte Simulationen und Analysen naheliegend: PROMETHEUS erlaubt die Definition und einfache Wiederverwendbarkeit einer Prozesslogik für eine beliebige Anzahl von Unterprozessen eines Projektes. Im Zusammenhang damit steht die Entwicklung eines Pakets in R, MDplot, das die Generierung und Bereitstellung von (publikationstauglichen) Graphiken vereinfacht und ein direktes Laden der Analysedaten einer Simulation ermöglicht. Diese Arbeit versucht, die Genauigkeit und Anwendbarkeit von Molekulardynamiksimulationen zu verbessern, indem geeignete Softwareanwendungen und Kraftfeldparameter zur Verfügung gestellt werden.

Publications

This thesis has led to the following publications:

- Margreitter C., Reif M., Oostenbrink C., *Update on phosphate and charged post-translationally modified amino acid parameters in the GROMOS force field*, submitted to: J. Comput. Chem. (2016), chapter 2
- Margreitter C., Oostenbrink C., *On the Optimization of the Protein Backbone Dihedral Angles by Means of Hamiltonian Reweighting*, J. Chem. Inf. Model. (2016), online, doi: <https://dx.doi.org/10.1021/acs.jcim.6b00399>, chapter 3
- Margreitter C., Mayrhofer P., Kunert R., Oostenbrink C., *Anti-body humanization guided by molecular dynamics simulations*, J. Mol. Recogn. (2015), online, doi: <https://dx.doi.org/10.1002/jmr.2527>, chapter 5
- Margreitter C., Oostenbrink C., *MDplot - an R plotting package for molecular dynamics analysis*, submitted to: The R Journal, chapter 7

Introduction

”Was die Welt im Innersten zusammenhält ...”

The desire to understand the world around us, might be as old as abstract thinking itself. However, the answers given as the rationale behind phenomena, great and small, such as the inexplicable forces of nature or the overwhelming variety life adopts, have changed in the course of time. From early ideas of archaic super-natural entities, to the complex pantheons and lore of institutionalized religions: all offered explanations for the ”real-world”, not just (tempting) afterlife perspectives. The first steps towards a more descriptive approach (with the focus on the ”how”, not the ”why”) were still intermingled with magic conceptions (e.g. alchemy) but laid the foundation of modern science as we understand it: ideally based on rational, controlled observations of experiments (empiricism), established knowledge and the criticism on and principle falsifiability of proposed ideas, we proceed step-by-step to an ever better (but nevertheless fragmentary, partial) model of reality.

Natural scientists have developed a toolbox and a code of conduct to disclose the secrets of nature bit by bit, including elaborations on sub-atomic matter, the core of stars and comprehensive studies on the human species and other organisms. The latter are investigated in biology, a scientific field sometimes regarded a sub-discipline of physics, trying to elucidate the part of the universes’ matter that forms life. There are different definitions of the criteria of life, among which *metabolism*, *growth*, *reproduction* and *variation* are often named. In this biological sense we are a mere collection of chemical processes - a complex one though. And as we begin to understand more and more of the manifold of the signaling pathways, enzymes and compounds we are made of, we are touching nothing less than the essence of our very existence. Knowledge e.g. on the principles of inheritance, mutation, genetic variety and evolution has the proximate potential to change the way we anticipate ourselves and our surroundings. Besides philosophical and cultural implications, there are indeed more ”profane” impacts on our lives, arising from the enhanced abilities to manipulate our habitat - as a profound understanding of something is the first step of adeptly influencing it. The development of the medical sciences, to name a striking example, has brought a hereto unknown life expectancy and living quality.

However, the complexity and sheer amount of what we know and what we investigate leads us to multiple frontiers and limits which only give way to coordinated and persisting efforts. The combination of many minds also offers the chance for synergistic effects between multiple fields of research, one complementing the limitations and blind spots of the other. In recent decades, a new technique gained importance throughout the realm of natural sciences (and sometimes even made progress possible): computational approaches.

In biophysics, computational simulations close an important gap between the physical world of small and individual molecules and the observation of the chemical world, i.e. of large collections of macro-molecules and oligo-complexes studied in wet-lab experiments (figure 1.1). We are nowadays able, not only to observe e.g. the functional effect a certain mutation has on a given protein (which is an extraordinary ability by itself) but it is also possible to nail down the physico-chemical forces, this effect arises from. We can monitor them computationally with such a fine-grained time and space resolution, that - sufficient computational potency provided - we can "watch" a receptor bind, a cleft form or a protein fold.

Besides the role as "spectators", computational modelling offers also more active advantages: we can force certain events to happen (and use elaborate correction protocols to obtain unbiased results afterwards), we can build molecules in an instant (without the hassle to find a proper reaction pathway in advance) and we can even change the laws of nature (and estimate for example, what the effect of a theoretical solvent would be). Such experiments allow us to deepen our understanding of the physical world even further and bear fruit in so important fields as the drug design process [1]. In order to do all that and more, we have to abstract from reality, i.e. we build models.¹ The answer to the frequent question, "Can you do that with your computer magic?" is quite complex and always depends on the context. For example, we are (theoretically) able to describe a protein-ligand binding process with extremely high precision by application of quantum mechanics, but we will fail to even calculate the proteins' wave function, Ψ , let alone to retrieve any useful information on the dynamics of binding. The reason being, that the chosen model (quantum mechanics) is too demanding computationally² (and presumably will be for the next decades to come).

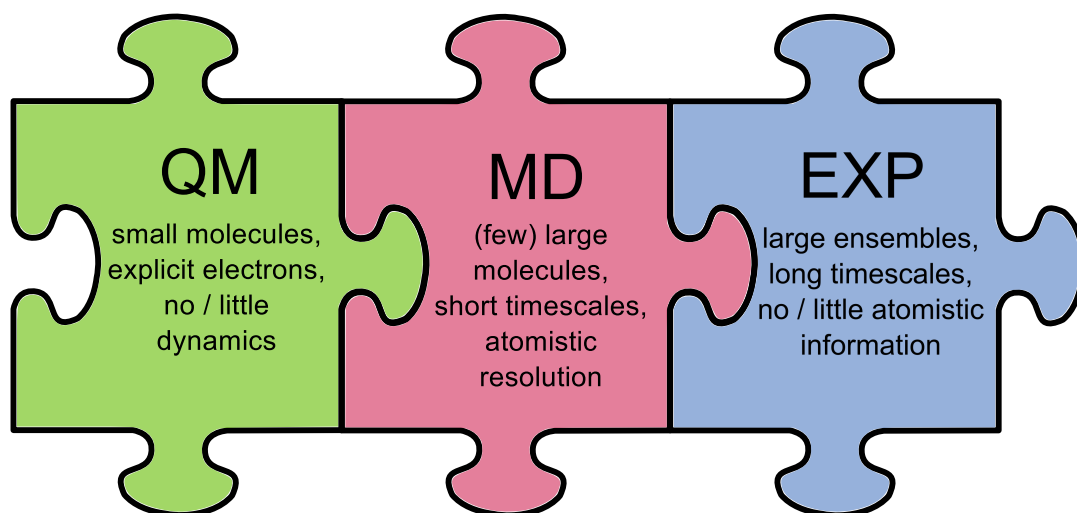


Figure 1.1: Shows, how a system can be addressed at different levels of accuracy, zooming in from right to left. Molecular dynamics simulations are well-suited to deal with the macro-molecular level and to elucidate the physico-chemical foundations.

¹The term *abstraction* refers to the neglect of all properties of reality, that are not relevant in a certain perspective.

²It's like describing the course of a football game by following the balls' translational path and its interaction with individual grass leaves. Theoretically possible and very accurate, but probably not very efficient to obtain the information one's interested in.

On the other end of the spectrum of computational methods lie docking algorithms. Although frequently applied as they are extremely fast and applicable to hundreds of thousands of ligands, they suffer from fundamental limitations (rigid protein, neglect of solvent entropy, ...) hampering their ability to quantitatively estimate a binding free energy. Or in other words, this approach might be too "rough" to tackle the initial question on a binding process in detail. In this particular case, molecular dynamics simulations might offer the best trade-off between accuracy (degrees of freedom) and applicability.

Thus, it is safe to say, that the question or the problem at hand determines the type of solution. This thesis uses molecular dynamics simulations to describe molecules of different sizes. What kind of questions might be subjected successfully to a molecular dynamics simulation study to gain an answer, is the topic of the following sections.

1.1 Molecular dynamics (MD)

The smallest complex entity composing organisms is a single molecule, bearing a small piece of information in its chemical structure, either directly as in a nucleic acid or indirectly such as the surface of a protein which is recognized by its binding partners. With molecular dynamics simulations, we usually find ourselves on this level: one or a few biomolecules in a box of water, maybe with the addition of a few ions. The theoretical description of these systems is done classically, meaning that we solve Newton's well-known equations of motion for atoms, treating them as deterministic particles. As one of the major abstractions, we include the electron clouds into the atoms (the famous Born-Oppenheimer approximation) and describe their behavior with simple functional forms. It is very critical how these equations are combined and parametrized, which requires experience as well as a cumbersome trial-and-error approach. A biomolecular force field [2] is the sum of the equations and their parameters and gives rise to the calculation of the potential energy and ultimately the forces. The forces in turn are required to solve the above-mentioned equations of motion. At subsequent time points, we calculate the atoms' velocities and propagate our system through time, i.e. we move the atoms in space. However, we need to do that very many times (because the time steps need to be small in order to produce a meaningful result). In the end, we get a collection of configurations of a system along the time axis, a so-called *trajectory*. How do we link this trajectory to the real world or at least to experiments? We are usually not interested in one molecule, but the average behavior of hundreds of thousands of molecules typically present in a wet-lab experiment. Moreover, experiments reach time scales of seconds to minutes, while we must restrain ourselves to nano- or milliseconds that are realistically accessible by computer simulations.

The required link is provided by the *ergodic theorem*, according to which it is the same to observe one system over a long time (infinity) or to monitor many independent molecules of the same kind. This means, as long as we sample long enough,³ we can calculate any observable (that depends on the structure) as an ensemble average. Because structures with comparably high, unfavourable potential energies are unlikely to occur, they do not contribute much to the probability-weighted (i.e. ensemble) average. This means, that even if we do not see these high-energy configurations during our simulations (since they

³What this really means has been subject to on-going discussions. However, there are some methods to test whether it is likely enough, that the sampling is sufficient ("convergence is reached") and all important configurations have been sampled according to their probability.

are too short) the resulting error is often negligible. However, if we ignore configurations not because of their high energy but because we never reached them in the simulation (e.g. due to energy barriers), we need to apply *enhanced sampling techniques* to obtain reliable ensemble averages. These ensemble averages can be compared to experimental observables, such as those derived from NMR experiments (J-coupling constants or NOE intensities) or measured hydration free energies.

1.1.1 Ensemble averages

In principle, a molecule can occur in a nearly infinite number of states [3]. The positions of its constituting atoms can be spread over the whole space considered and the velocities, the atoms have, can similarly adopt all values in all directions. However, it is clear, that not all of these position-velocity pairs are of the same probability. For example, covalent bonds have a certain length, d_0 , that they adopt most often. Oscillations from this type-specific value are energetically unfavorable, i.e. these distances have a higher energy than d_0 . This means, that we can in turn calculate the probability to obtain a certain configuration from its energy, depending on the coordinates, q , and the momenta, p (equation 1.1). The denominator is the normalization constant, required since the probability to be in a certain configuration depends on the probabilities to be in other configurations.

$$P(q, p) = \frac{\exp^{-H(q, p)/k_B T}}{\iint \exp^{-H(q, p)/k_B T} dq dp} \quad (1.1)$$

The collection of all configurations, together with their probabilities, is what we call an ensemble. In order to calculate an ensemble average, denoted by angle brackets, of a quantity A , we apply equation 1.2.

$$\langle A \rangle = \iint A(q, p) P(q, p) dp dq \quad (1.2)$$

These ensemble averages are equivalent to the experimentally determined values.⁴ The Hamiltonian, H , in the equations above represents the total energy (kinetic plus potential energies) of a configuration. The kinetic energy is calculated by equation 1.3, reflecting the momenta of the particles. For a description of the potential energy, see section 1.1.3.

$$E_{kin} = \frac{3}{2} N k_B T = \sum_i^N \frac{m_i v_i^2}{2} \quad (1.3)$$

where N is the number of particles, k_B is Boltzmann's constant, T is the absolute temperature and m_i and v_i are the mass and velocity of particle i , respectively.

⁴Given that the Hamiltonian, H , is absolutely correct, the observable, A , is computed absolutely correctly and the integration really covers all of the phase-space.

1.1.2 Integration

The term "simulation" in the context of molecular dynamics (MD) means to follow our system through time, i.e. to shoot a movie of a molecular system. In order to do so, we take small discrete timesteps (2 fs), our "picture rate", and calculate what our system will look like in the very near future, i.e. we *integrate* over time (see equation 1.4).

$$x(t + \Delta t) = x(t) + v(t + \frac{1}{2}\Delta t) * \Delta t \quad (1.4)$$

To do that, we need the atom positions of the last snapshot or our initial structure, $x(t)$ or $x(t_0)$, and the updated velocities of the atoms, $v(t + \frac{1}{2}\Delta t)$, at intervals shifted half a timestep Δt with respect to the coordinates (the so-called *leap-frog algorithm*). The velocities are calculated according to equation 1.5:

$$v(t + \frac{1}{2}\Delta t) = v(t - \frac{1}{2}\Delta t) + a(t) * \Delta t \quad (1.5)$$

Acceleration, $a(t)$, times a time-interval is again a velocity, which is added to the "old" velocity calculated before. Now, by Newton's second law of motion, we know that the acceleration is the force, F , divided by the mass, m . Therefore, we need to determine the forces on the atoms prior to the integration. We can do this, by calculating the negative partial derivative of the potential energy U (see section 1.1.3 for the contributions to this sum) with respect to the coordinates x . This is shown in equation 1.6. To sum it up: the new velocities are the sum of the old velocities and changes in the velocities, introduced by the potential energy that appears for the atom configuration half a timestep before.

$$F = -\frac{\partial U}{\partial x} = m * a \rightarrow a = \frac{F}{m} \quad (1.6)$$

1.1.3 Potential energy

The potential energy, U , is the sum of various energetic contributions (equation 1.7), that define the energy stored in a body or between bodies [4, 5]. Together with the kinetic energy it forms the total energy. In an isolated system, the total energy stays constant and according to the laws of thermodynamics, it is only possible to convert the potential and the kinetic energy into each other, not to generate or destroy them. The potential energy is mainly determined by the electrons of the participating atoms. In molecular modelling we split the description of the electrons into simpler functional forms, e.g. we describe the restricted rotation around a bond with partial double-bond-character by an appropriate dihedral angle potential and the electrostatic interactions in a different way. In general, we distinguish two kinds of interactions that contribute to the potential energy: the bonded and non-bonded interactions. The first ones occur along covalent bonds while

the others are mediated through space. The following description tries to introduce the most used interactions in molecular dynamic simulations.

$$U^{total} = U^{bonds} + U^{angles} + U^{torsions} + U^{vdW} + U^{Coulomb} + \dots \quad (1.7)$$

1.1.3.1 Bonded interactions

The basic structure of (bio)molecules is determined by the covalent bonds between their constituting atoms. A simple and widely-used description in the context of MD is to define a distance at which the energy is zero and to use a harmonic potential-energy function to calculate the energy (giving rise to a force) that comes from deviations introduced by the movements of atoms. The second parameter in harmonic oscillators is the force constant, that defines the "steepness" of the energy function. The following descriptions include the equations of the potential-energy functions and general explanations. Figure 1.2 shows examples for the potential-energy curves.

Bonds Covalent bonds arise from sharing electrons between atoms. As the strength and the minimum distance vary in respect to the binding partners and type of bond, a set of possible combinations is needed in our forcefield. The bonding also explains the polarity distribution among the atoms, based on their electro-negativity. We describe these electrostatic interactions by the Coulomb potential, but neighboring atoms, connected to each other by three (or four) bonds or less, are excluded in this respect (since the bonded interactions take care of that). Equation 1.8 shows the calculation of the potential energy contributions by bonds. The energy minimum distance, $d_{type(i)}^0$, and the force constant, $k_{type(i)}$, are the two parameters used.

$$U^{bonds} = \sum_{bonds\ i}^N \frac{1}{2} k_{type(i)} (d_i - d_{type(i)}^0)^2 \quad (1.8)$$

Strikingly, the energy resulting from bond-stretching is much higher, than the one arising from angles (which is reflected in the parameters by different orders of magnitude for the respective force constants). Thus, deviations in the bond length are very limited at room temperature (0.006 nm) and do not contribute much to the dynamics. Furthermore, the vibrational frequencies are so high, that a classical treatment is not really appropriate anyway. Therefore, we tend to take out this degree of freedom by constraining the bond lengths to their respective minimum at every timestep (by e.g. SHAKE or LINCS). As the duration of the integration timestep can thus be increased from 0.5 to 2 fs, we are able (neglecting the constraining effort) to quadruple the efficiency of our simulations.

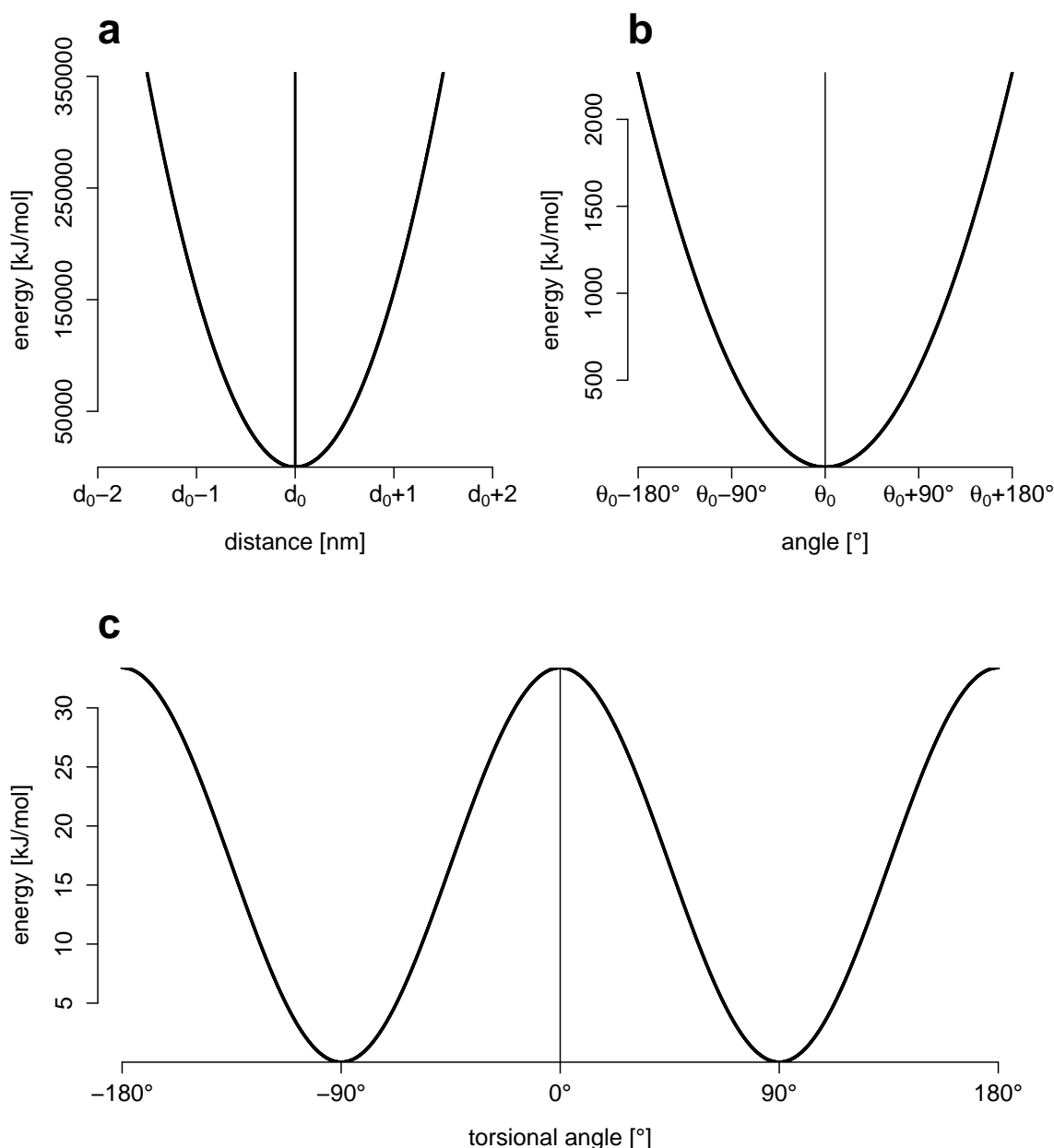


Figure 1.2: Shows the potential energy for the computational descriptions of covalent bonds (a), angles (b) and dihedral angles / torsions (c). The first two are harmonic potentials, the latter needs to take rotations into account (see equations).

Angles Formed by three atoms joined together by bonds (see above), angles are considered to be mostly of local effect inside a macro-molecule. The minimum angle, $\theta_{type(i)}^0$, and a force constant, $k_{type(i)}$ (much lower as those for bonds) are the two parameters used to again calculate the harmonic potential energy for deviations in degrees (equation 1.9). Angles can deviate about 10° from their minimum value (often about 110°) at room temperature.

$$U^{angles} = \sum_{angles\ i}^N \frac{1}{2} k_{type(i)} (\theta_i - \theta_{type(i)}^0)^2 \quad (1.9)$$

Dihedral angles / torsions Four atoms subsequently bound to each other may experience rotations around the middle bond. The dihedral angle is defined to be the angle between the plane formed by the first three atoms and the plane formed by the last three atoms of that fragment. Rotations around dihedrals can have large effects on the structure and are a very crucial part of a force field (e.g. the amino acids' backbone dihedrals). As shown in equation 1.10, torsions are not described by a harmonic potential, but by a periodic function (to take into account full rotations). The parameters used are the force constant, $k_{type(i)}$, the multiplicity, $m_{type(i)}$, which is the number of minima over 360° and a shift, $\delta_{type(i)}$, that moves the cosine function. The energy profile imposed by torsions might seem small compared to the others. However, despite these (individually) low energy contributions, the dihedral angles are of high importance in describing the dynamics (see chapter 3).

$$U^{torsions} = \sum_{torsions\ i}^N k_{type(i)} \cos(m_{type(i)}\phi_i + \delta_{type(i)}) \quad (1.10)$$

Note, that multiple terms may be applied to the same set of four atoms to account for more complex energy barriers.

Improper dihedrals Sometimes it is convenient to introduce additional terms to the potential energy, either to bias the simulation in a desired way or to account for effects, that are not covered by the "regular" terms. One of the latter kind is the definition of so-called *improper dihedrals*. They are built like normal dihedrals (using four atoms) and can be used to enforce a certain behavior, e.g. to keep a phenol-ring flat.⁵

1.1.3.2 Non-bonded interactions

There are two interactions between atoms mediated through space, which need to be considered in MD simulations: van-der-Waals (vdW) and electrostatic / Coulomb interactions (see figure 1.3). However, they represent rather rough approximations and currently undertaken efforts aim at the refinement of electron-dependent non-bonded interactions, for example by the introduction of polarisability.

⁵In a quantum mechanics description, the delocalization of the π -electrons in the ring system would ensure that. Since we describe no electrons / orbitals explicitly, that simplest solution is to use an improper dihedral instead.

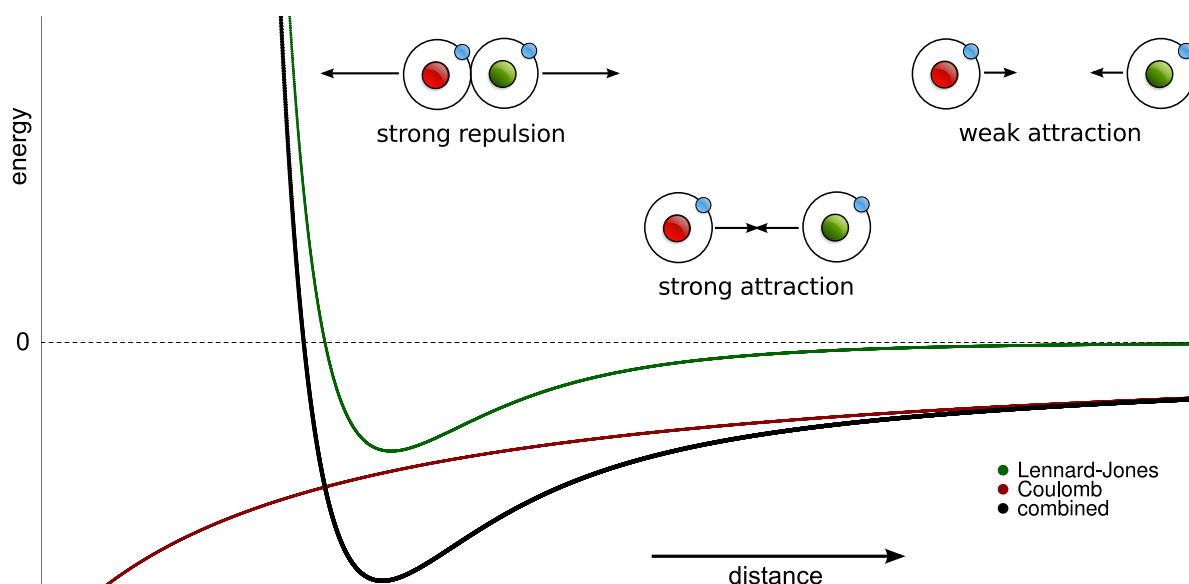


Figure 1.3: The three curves show the attractive and repulsive energies arising from non-bonded interactions. A negative energy (i.e. the combined curve is below zero), means that the non-bonded interactions for a given pair of atoms are attractive. The distance, d , is on the x-axis, the energy on the y-axis. At a large distance (right), two ions with opposite (fixed) charge sign attract each other slightly, giving rise to a force pulling them together. At this distance, the vdW-interactions are negligibly small. When the two ions come closer (middle), not only the vdW-interaction adds to the attractive energy, but also the strength of the electrostatic one increases. Thus, the force becomes stronger. When a certain distance is undercut (when the vdW-radii start to overlap, left), the Pauli repulsion becomes dominant and the unfavourable energy leads to a separation of the ions.

Non-polar interactions The distribution of electrons is fluctuating around the atoms' nucleus, which leads to transient, small dipoles (transient charges δ^+ / δ^-). If two atoms come close to each other and the kinetic energy is not too high, they will form an aligned dipole system, i.e. they usually induce a dipole such, that an attractive force emerges. This *dispersive attraction* is approximated in the right term of the Lennard-Jones-potential (LS) shown in equation 1.11. It is very short-ranged (and thus can be treated by a cut-off, beyond which the interaction is neglected) and diminishes with d , the distance, to the power of -6. The susceptibility to dipole inductions and the own ability to do so is different for every atom.

Therefore, a list of parameters needs to be provided: both the well-depth (ε) and the zero-energy distance (σ) need to be defined for every pair of possible atom types. In the GROMOS force-field, there are several atom types for some elements (a hydroxy and a carboxy oxygen, for example), sharing the same mass, but having molecule-dependent partial charges and type specific van-der-Waals interaction parameters. For efficiency reasons, the repulsion that occurs when two atoms come too close (see figure 1.3) is calculated as being to the power of -12 (just meaning "very steep"). The physical background of this repulsion stems from the Pauli principle that claims that no two electrons can have the same spin-orbit quantum numbers, i.e. electronic orbitals cannot overlap.

$$U^{LJ} = \sum_{\text{pairs } i < j} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] \quad (1.11)$$

Coulomb interactions As mentioned earlier, we also use the concept of "permanent dipoles" or partial charges, denoted as q^+ or q^- . They are not introduced temporarily but stem from the different electro-negativities of different elements. The atoms in a water molecule for example share electrons through their bonds, but the electron density is concentrated at the oxygen,⁶ i.e. the oxygen "pulls" the electrons due to its higher electro-negativity. While the vdW-interactions take also place in non-polar molecules, the electrostatic ones depend directly on these uneven average distributions of electrons, which we describe by assigning partial charges to the atoms. The Coulomb potential forms the basis of electrostatics and delivers the energy (attractive or repulsive, depending on the signs of the partial charges, q_i) associated with these interactions (see equation 1.12). The main problem in the context of molecular dynamics simulations is the reach of the electrostatic interaction: it decays linearly with r and should in principle be calculated for a very long range (definitely longer than our default cut-off radius of 1.4 nm). Still, for the sake of calculation speed, we can use a cut-off - as long as we apply enhanced protocols (reaction-field outside the cut-off distance, etc.) to mitigate the occurring errors. The only parameters needed are the partial charges of every atom to calculate the overall contribution in a pair-wise manner. However, these parameters depend on the atoms' (bonded) neighbors and are thus different for each individual molecule. In the GROMOS force field, we try to dissect molecules into smaller compounds, often just functional groups, and transfer the parameters by analogy. This approach effectively limits the propagated effect of neighboring atoms to the electron configurations to stop at the charge group boundaries.

$$U^{Coulomb} = \sum_{\text{pairs } i < j} \frac{1}{4\pi\epsilon_0\epsilon_r} \frac{q_i q_j}{r_{ij}} \quad (1.12)$$

1.1.4 Parametrization

As has been elaborated, the underlying parameters associated with the molecules have a direct and decisive effect on the results. The more accurate our model, the more precise are predictions and the better the applicability of the method. In the GROMOS force field, we usually try to parametrize a small molecule and compare its properties to experimental data. If the match seems sufficient after a number of iterative adaptations (figure 1.4), bigger systems are decomposed into small chemical moieties in such a way, that the parameters can be transferred by analogy. Afterwards, building blocks such as amino acids (and whole proteins) can be reconstructed. The broader the library of parametrized small molecules, the better the physical representation provided by the model afterwards. Therefore, and because new experimental data and application experience becomes available over time, a force-field parameter set is constantly extended, changed and validated.

⁶In MD, we often use the simple approximation of a water molecule as a "sphere", e.g. when we "roll" it over a surface.

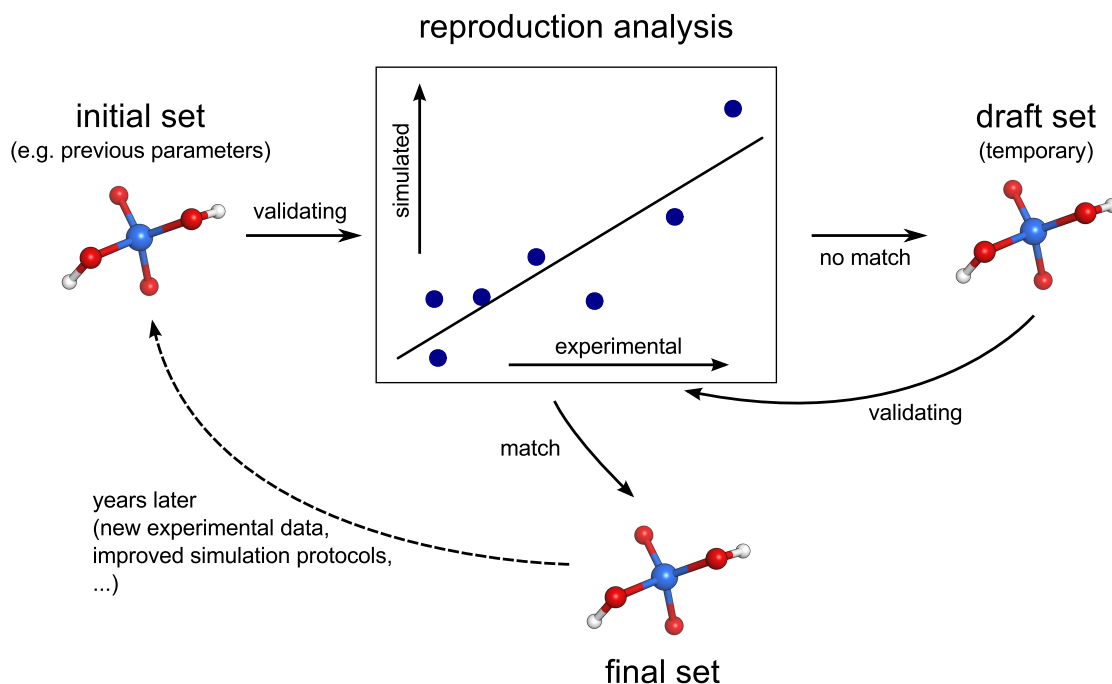


Figure 1.4: The parametrization process starts with the validation of an initial set of parameters against experimental data. Afterwards, an iterative cycle of reparametrization and revalidation runs, until an acceptable deviation is reached. Usually, other factors than just a good match, such as consistency with the rest of the force field are also taken into account. When new experimental data becomes available or problems occur with the parameters in real-world applications or new properties become accessible due to enhanced computational resources, a new round of parametrization might start.

1.1.5 Free energy

As a simple definition, the free energy is the combination of an energetic (enthalpic) and an entropic term that defines the probability of a process to take place. To use a binding process as an example again, there will be an unfavorable enthalpic contribution upon binding because of the loss of solvent-solute interactions (as parts of the binding partners will be buried afterwards). Moreover, the binding leads to a restriction in intrinsic degrees of freedom, giving rise to a negative and therefore unfavorable entropic change ΔS . But it will also release highly ordered water molecules to the bulk (entropy gain) and the favorable interactions between the binding partners must also be taken into account. In the end, a specific dynamic (binding) equilibrium will be the result according to the association and dissociation rates determined by the favorable and unfavorable contributions. This is quantified by the free energy, ΔG . To reliably access free energy changes is a major advantage of molecular dynamics simulations [6–8]. As described before even unphysical processes such as the alchemical change of a proton into a methyl-group can be appropriately calculated to estimate relative free energies. In this study, hydration free energies have been calculated by thermodynamic integration (TI)⁷ [10].

⁷In MD, thermodynamic integration and Bennets Acceptance Ratio (BAR) [9] are considered to be very accurate methods and are often used for validating other approaches.

1.2 Proteins

One of the major classes of biomolecules (and also pivotal for this thesis) are proteins. They consist mainly of long, linear polypeptide chains built from amino acids as primitive building blocks. As they often work as enzymes (proteins with catalytic activity), they are key players for structural and dynamic studies, connected to a manifold of diseases⁸ and frequently targeted in pharmaceutical drug design. In order to simulate proteins properly, the amino acid level needs to be addressed first. In a way similar to actual biosynthesis, we parametrize amino acids individually and connect these residue-wise parameters afterwards in order to form the *topology* of the protein. Needless to say, that errors in the building blocks propagate to all higher levels of complexity.

In addition to the 20 canonical amino acids, nature has developed a large set of post-translational modifications to enhance chemical diversity. They serve as signaling moieties, regulatory switches (for they are often reversible) or occur randomly due to non-catalyzed chemical reactions. More than 400 types have been reported [11, 12] and most proteins are believed to be exposed to post-translational modifications. Naturally, simulating these modified amino acids is an ability, often required in molecular dynamics simulations when dealing with proteins and is also addressed in this thesis.

A particularly important group of proteins are antibodies. These agents of the immune system target certain surface structures by specific binding (see figure 1.5) and set a manifold of immune processes into motion. Their ability to adapt to such a variety of surfaces (epitopes) is crucial in order to keep pace with the extremely high rate of evolutionary changes observed in bacteria and viruses. Antibodies facilitate that, by being subjected to an evolutionary process themselves: to find an antibody directed against a particular epitope, certain genes (native germlines) are randomly mutated, validated and, if binding succeeds, produced massively. In recent years, antibodies have additionally gained importance as drug molecules themselves, due to their diversity and specificity. Because this process is very complex and complicates drug discovery, model organisms are often used to generate initial antibodies. However, since the germlines are organism-specific, adverse side-effects might take place when administering these antibodies as drugs. In order to overcome these limitations, protocols such as super-humanization can be applied, which is the basis of the project described in chapter 5 of this work.

1.3 This thesis' contribution

As elaborated in the previous sections, accurate parameters are crucial since the energy differences that govern macro-molecules are typically on a very fine scale. For example, it might be sufficient to introduce a single mutation into a sequence of a hundred amino acids to change its behaviour or even to alter its structure significantly. The reason is, that a small contribution might suffice to end up in a completely different free energy minimum. Therefore, parameters must be accurate on a comparable level.

In the context of this study, these small free energy differences, arising from mutations and changes in parameters played a major role. First, we contributed to the on-going development of the GROMOS force field by parametrizing phosphate ions to better match

⁸Most often because of critical mutations in their sequence, but their expression might also be abnormally up- or down-regulated.

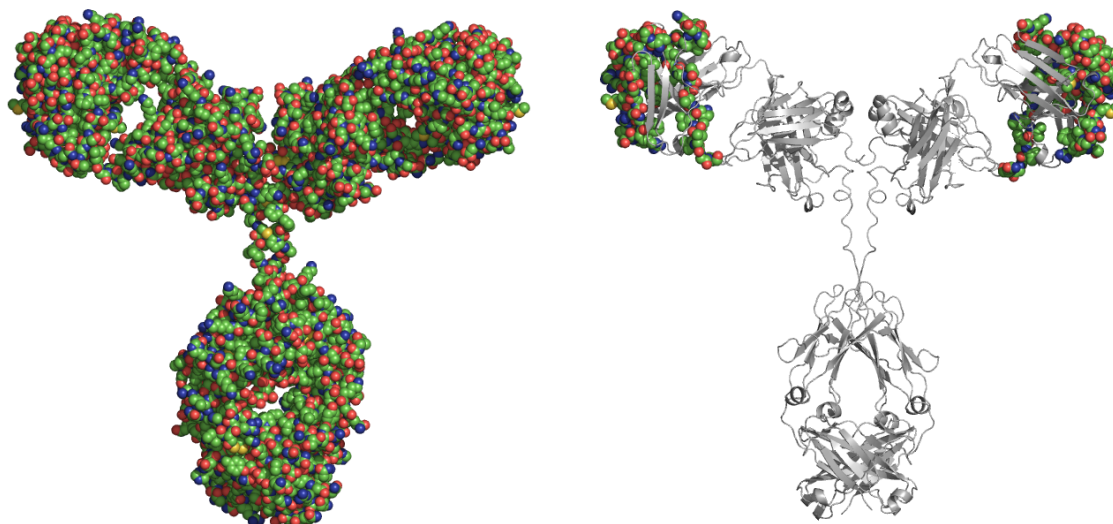


Figure 1.5: Model of an IgG antibody, which has a Y-like shape with two identical parts left and right (left picture). Each part consists of a heavy and a light peptide chain, linked by a disulphide bridge. The fold of the arms is very stable and built out of β -sheets interspersed by highly flexible loop regions. The sheets are called "framework"-regions and the loops, as they facilitate the actual binding, complementarity-determining regions or CDRs. They are shown with their van-der-Waals radii in the right picture. The CDRs are strongly mutated in a process called *antibody maturation* in which they adapt to a given epitope.

experimental hydration free energies (chapter 2). These moieties are very important in the backbone of nucleic acids and the head groups of (membrane-forming) lipids, but also as the dominant type of protein post-translational modifications (PTMs). We could show, that the old partial charges (used for the electrostatic interactions) underestimated the experimental target free energies, while the newly proposed parameter set performs significantly better. Simulations using phosphate moieties are likely to profit from these new parameters - for example, when simulating proteins containing phospho-serine, phospho-threonine or phospho-tyrosine.

In chapter 3, we analysed the performance of the amino acids backbone dihedral angle potentials in the context of very small systems (capped amino acids). Surprisingly, the parameter set 54A7, which is used successfully on the protein scale, fails to reproduce experimental J-values and secondary structure propensities as well as Ramachandran distributions obtained from a PDB survey by a significant amount. Besides finding a rationale behind this observed discrepancy on different levels of peptide complexity, we could apply a new large-scale approach to screen and identify more promising parameter candidates. Another result was the identification of the C_β -branching pattern as the most critical difference, leading to different parameter suggestions for four amino acid sub-groups. This effort might serve as an important first step in the reparametrization of the amino acids' dihedral angle energy potential. In the long run, this could lead to a better handling of small peptide fragments and intrinsically unstructured proteins.

As mentioned before, small changes might affect large proteins, which was shown in the antibody humanization project (chapter 5). In this study, we developed a method to predict the effect of (back-)mutations in the framework regions of an antibody on its binding affinity - without actual calculations of any binding free energies.⁹ Instead, we extracted and condensed structural and dynamical features of the antibody loop regions and calculated a predictive score from that data, based on comparison to the wild-type (a confirmed binder). This score calculation was based on a clustering algorithm. The comparison to a training and a prediction dataset, validated by binding experiments performed by our collaborators, revealed successful predictions and the correct identification of a critical amino acid embedded in a "tyrosin-cage". To strengthen the accuracy of our predictions, a manifold of simulations had to be performed including numerous replicate simulations.

In order to execute these large series of logically similar simulations and their subsequent analysis, we have implemented PROMETHEUS, a molecular dynamics workflow manager (chapter 6). Its main function is to separate the execution logic from the actual compounds of interest, e.g. antibodies with different sequences are simulated in one way and small molecules, requiring different settings, in another. In addition to easily subject an arbitrary number of series' members to a certain batch processing (e.g. a simulation), they are structured in an elaborate way and can be manipulated by an user-friendly web-based interface.

Furthermore, since the first step in examining and explaining analysis results is not to crunch through numbers but to simplify these in graphical representations, i.e. figures, we developed an automated plot generation tool to generate most of the commonly required plot types directly from the users' bash scripts or PROMETHEUS instruction files. This R package, called MDplot, has comprehensive parsing and selection abilities and offers straightforward loading functions to simplify usage. A list of all plotting, loading and support functions can be found in chapter 7. Finally, some smaller contributions to the field are described in two addenda at the end of the thesis.

The following chapters describe all of the mentioned parts of this thesis in detail and provide an outlook on the developments yet to come.

Sapere aude!

⁹Which would be difficult due to the large binding interface involved and the intrinsic flexibility of the loop regions.

References

- [1] Jacob D. Durrant and J. Andrew McCammon. “Molecular dynamics simulations and drug discovery”. In: *BMC Biology* 9 (2011), p. 71. DOI: [10.1186/1741-7007-9-71](https://doi.org/10.1186/1741-7007-9-71).
- [2] Olgun Guvench and Alexander Jr. MacKerell. “Comparison of Protein Force Fields for Molecular Dynamics Simulations”. In: *Molecular Modeling of Proteins*. Ed. by Andreas Kukol. Methods Molecular Biology 443. Humana Press, Jan. 2008, pp. 63–88.
- [3] Donald Allan McQuarrie. *Statistical Mechanics*. 1st edition. Sausalito, Calif: University Science Books, May 2000.
- [4] Wilfred F. van Gunsteren et al. “Biomolecular Modeling: Goals, Problems, Perspectives”. In: *Angewandte Chemie International Edition* 45.25 (June 2006), pp. 4064–4092. DOI: [10.1002/anie.200502655](https://doi.org/10.1002/anie.200502655).
- [5] William L. Jorgensen and Julian Tirado-Rives. “Potential energy functions for atomic-level simulations of water and organic and biomolecular systems”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.19 (May 2005), pp. 6665–6670. DOI: [10.1073/pnas.0408037102](https://doi.org/10.1073/pnas.0408037102).
- [6] D. L. Beveridge and F. M. DiCapua. “Free Energy Via Molecular Simulation: Applications to Chemical and Biomolecular Systems”. In: *Annual Review of Biophysics and Biophysical Chemistry* 18.1 (1989), pp. 431–492. DOI: [10.1146/annurev.bb.18.060189.002243](https://doi.org/10.1146/annurev.bb.18.060189.002243).
- [7] Peter Kollman. “Free energy calculations: Applications to chemical and biochemical phenomena”. In: *Chemical Reviews* 93.7 (Nov. 1993), pp. 2395–2417. DOI: [10.1021/cr00023a004](https://doi.org/10.1021/cr00023a004).
- [8] Yuqing Deng and Benoît Roux. “Computations of Standard Binding Free Energies with Molecular Dynamics Simulations”. In: *The Journal of Physical Chemistry B* 113.8 (Feb. 2009), pp. 2234–2246. DOI: [10.1021/jp807701h](https://doi.org/10.1021/jp807701h).
- [9] C. H. Bennett. “Efficient estimation of free energy differences from Monte Carlo data”. In: *Journal of Computational Physics* 22 (Oct. 1976), pp. 245–268. DOI: [10.1016/0021-9991\(76\)90078-4](https://doi.org/10.1016/0021-9991(76)90078-4).
- [10] J.G. Kirkwood. “Statistical mechanics of fluid mixtures”. In: *J. Chem. Phys.* (1935), pp. 300–313.
- [11] George A. Khoury, Richard C. Baliban, and Christodoulos A. Floudas. “Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database”. In: *Scientific Reports* 1 (Sept. 2011). DOI: [10.1038/srep00090](https://doi.org/10.1038/srep00090).
- [12] Christopher T. Walsh, Sylvie Garneau-Tsodikova, and Gregory J. Gatto. “Protein posttranslational modifications: the chemistry of proteome diversifications”. In: *Angewandte Chemie (International Ed. in English)* 44.45 (Dec. 2005), pp. 7342–7372. DOI: [10.1002/anie.200501023](https://doi.org/10.1002/anie.200501023).

1.4 Funding

This work was supported by the European Research Council (ERC; grant number 260408), the Austrian Science Fund (FWF; grant number P 25056) and the Vienna Science and Technology Fund (WWTF; grant number LS08-QM03).

Update on phosphate and charged post-translationally modified amino acid parameters in the GROMOS force field

Adapted from:

Margreitter C., Reif M. M., Oostenbrink C., *Update on phosphate and charged post-translationally modified amino acid parameters in the GROMOS force field*, submitted to: J. Comput. Chem. (2016)

Author contributions:

CM performed and analyzed all simulations, MR developed necessary code, MR and CO supervised the work and assisted in writing the manuscript.

In the context of bio-molecular simulations, the most important decision that has to be made in advance concerns the degree of resolution necessary to tackle the respective scientific question. Quantum mechanics generally represents the most appropriate and thorough approach but is also prohibitively expensive in terms of computational resources for larger bio-molecules and relevant time-scales. Therefore, simplifications to the pure electrostatic descriptions have been introduced, such as the treatment of overlapping electron orbitals as bonded interactions or the assignment of fixed partial charges on atoms instead of describing their complex and polarizable electron clouds in detail. These simplifications require a balance between these bonded and non-bonded parameters in order to form a so-called *self-consistent set* of parameters. The basis of molecular dynamics simulations is therefore laid by the functional forms (the equations, i.e. the mathematical formulations of the interactions) and the parameters associated with them. Together, this is what we call a *force field*. The equations are well studied and more or less the same in most of the implementations of molecular dynamics engines, but to find a consistent set of parameters is a very demanding task. One development approach is to assign parameters based on quantum mechanical calculations (as in the case of AMBER and CHARMM, for example). In the context of GROMOS, however, experimentally accessible data for small

molecules is used to calibrate an initial set of parameters iteratively for these molecules. Afterwards, these parameters are further refined for bigger systems such as proteins. Naturally, the development of parameter sets is an open process as new experimental data or practical experience in the application of the force field becomes available or new properties become computationally accessible. A recent reparametrization of the charged amino acids aimed at the improved description of the involved functional groups. In addition to that, in this chapter, we describe the development of new partial charges for different kinds of phosphate ions using a method-independent approach providing reliable parameters. Phosphate groups are pivotal not only in the DNA backbone but also in various post-translational modifications applied to amino acid side-chains and small metabolic molecules. Based on the newly derived phosphate parameters and parameters established for other charged moieties reported by Reif et al., we propose new parameters for our previously reported set of parameters for post-translationally modified amino acids.

2.1 Abstract

In this study, we propose newly derived parameters for phosphate ions in the context of the GROMOS force field parameter sets. The non-bonded parameters used up to now lead to a hydration free energy of the dihydrogen phosphate ion, which is too hydrophobic when compared to experimentally derived values, which made the reparametrization of the phosphate moiety necessary. Phosphate species are of great importance in biomolecular simulations not only because of their crucial role in the backbone of nucleic acids but also since they represent one of the most important types of post-translational modifications to protein side-chains and are an integral part in many lipids. Our re-parametrization of the free dihydrogen phosphate (H_2PO_4^-) and three derivatives (methyl-phosphate, dimethyl-phosphate and phenyl-phosphate) lead, in conjunction with the previously updated charged side-chains in the GROMOS parameter set 54A8, to new nucleic acid backbone parameters and a 54A8 version of the widely used GROMOS protein post-translational modification parameter set.

2.2 Introduction

The accuracy and applicability of molecular dynamics simulations always depend on the parameters used in the underlying force field. Critical testing is therefore of utmost importance, to ensure a proper representation of the most significant interactions in our model simulations. Parameter sets change in the course of time to reflect newly gained knowledge and to account for improvements in the models - thus, development is never completed and there is always room for further improvements. These adaptations include bond parameters, angles, torsions and non-bonded interactions alike. The final benchmark of our parametrization efforts is the reproduction and eventually prediction of experimental values. Commonly used in this context is the experimentally determined hydration free energy, especially in the validation of partial charge assignments [1, 2].

Among the most recent updates in terms of the GROMOS force field is the reparametrization of charged biologically relevant moieties by Reif et al. [3, 4]. In these publications, the authors focussed on the reparametrization of charged groups that are pivotal in the context of peptide and protein simulations, including the side-chains of aspartate, glutamate, lysine, histidine and arginine, as well as on a set of ions. This new parameter set, 54A8, was shown to perform significantly better in reproducing experimental observables

for small molecules. However, one of the most important charged groups has not been addressed, namely dihydrogen phosphate (H_2PO_4^-) and its derivatives. These moieties are crucial not only in the backbone of nucleic acids but also occur frequently in modifications of the side-chains of certain amino acids, so-called post-translational modifications. Moreover, the head groups of many lipids in membranes contain them as well. A search in the uniprot database [5] shows, that among all reported post-translational modification (PTM) sites, the phosphorylation of various residues, including threonine, tyrosine and serine, represents roughly 75% [6]. Because of the big change they induce in the target amino acids' chemical nature, phosphorylations are frequently used as reversible modulators of enzyme activity and selectivity.

It is thus highly important to provide well validated phosphate group parameters, which is the major aim of the current study. In order to achieve that, an experimental hydration free energy and experimental pK_a values have been used in a method-independent approach established earlier, to optimize the partial charges of phosphate, methyl-phosphate, di-methyl-phosphate and phenyl-phosphate, respectively. Comparison to the results obtained for the GROMOS 54A7 parameter set shows that the hydration free energy was strongly underestimated in this parameter set. In conjunction with parameters obtained from the 54A8 parameter set [3], we furthermore report new parameters for the PTM force field [6] for all post-translational modifications involving a charged side-chain. The new post-translational parameter set is already implemented in our Vienna-PTM webserver resource [7]. With this contribution to the most recent GROMOS force field parameter set we hope to enable more appropriate studies involving post-translational modifications and nucleic acids.

2.3 Methods

In order to access the hydration free energy of the various phosphate compounds computationally, it is useful to split the different contributions for individual assessment [3, 8, 9] as outlined graphically in figure 2.1. The first free-energy contribution arises from the (uncharged) cavity formation due to the van-der-Waals interactions between the solute and the surrounding solvent. The second term is the raw charging free energy, i.e. due to the effective electrostatic interactions of the individual solute atoms with the solvent. The size of this term depends on the values assigned to the partial charges, which are the key target of our study, and on the employed electrostatic-scheme as well as the size of the simulated system. There are certain effects that need to be addressed in order to obtain the standard intrinsic single-ion charging free energy from this raw one. The necessary correction terms are further described below.

In order to compare the methodology-independent standard intrinsic single-ion free energies of hydration to experimental data, one needs to select an appropriate value for the (experimentally elusive) hydration free energy of the proton. Here, we choose the same value as was used in the calibration of the 54A8 parameter set, $-1100 \frac{\text{kJ}}{\text{mol}}$ [3, 8] for the standard absolute intrinsic proton hydration free energy. After a careful consideration of the available experimental data (see *Results and discussion*), the partial charges for the dihydrogen phosphate ion were recalibrated using this approach.

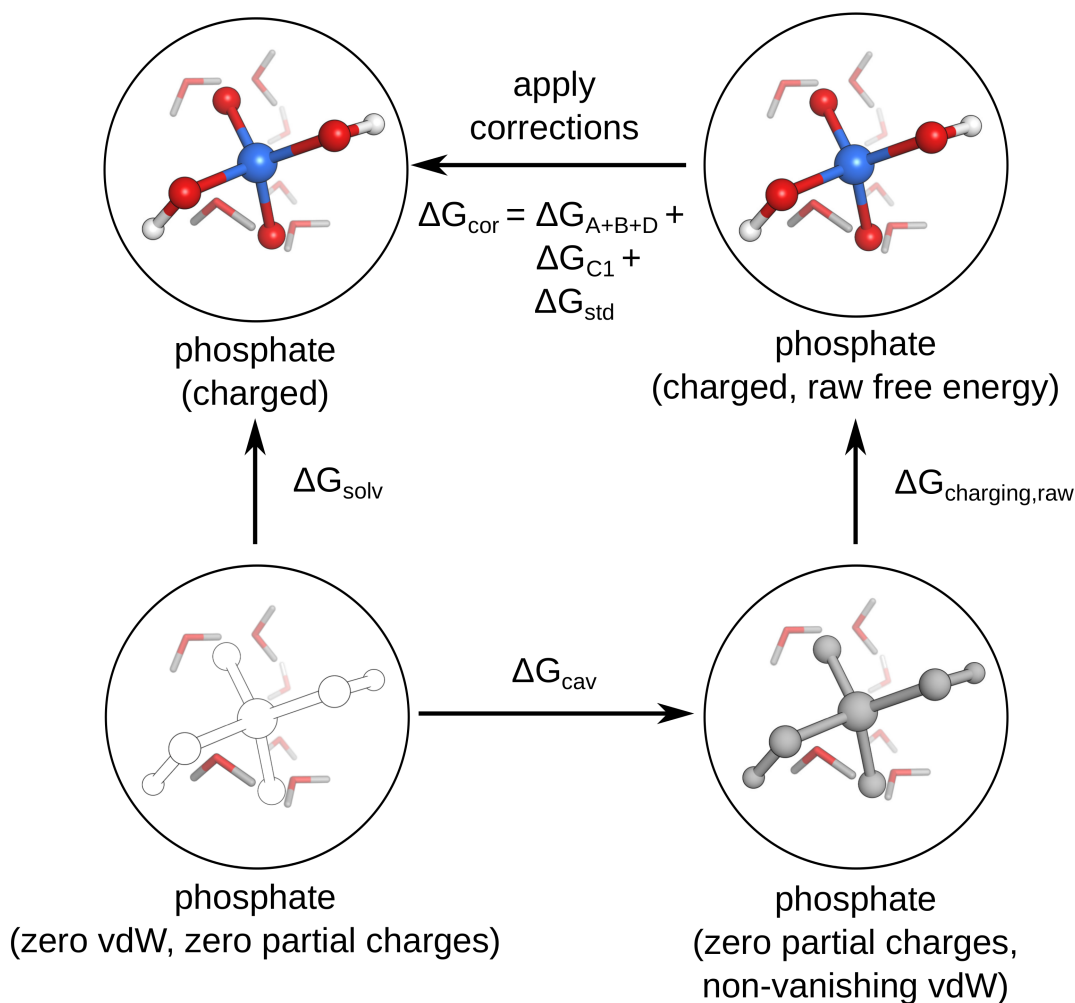


Figure 2.1: Graphical illustration of the contributions to ΔG_{solv} . For a detailed description of the contributions see the main text.

The other species investigated, namely methyl-phosphate, di-methyl-phosphate and phenyl-phosphate, have been calibrated to be in accordance to the obtained new parameters for the charged dihydrogen phosphate by appropriate thermodynamic cycles (figure 2.2, see below). Throughout this study, the standard state is defined by a pressure of 1 bar, a temperature of $T=298.15$ K and a concentration of 1 molal.

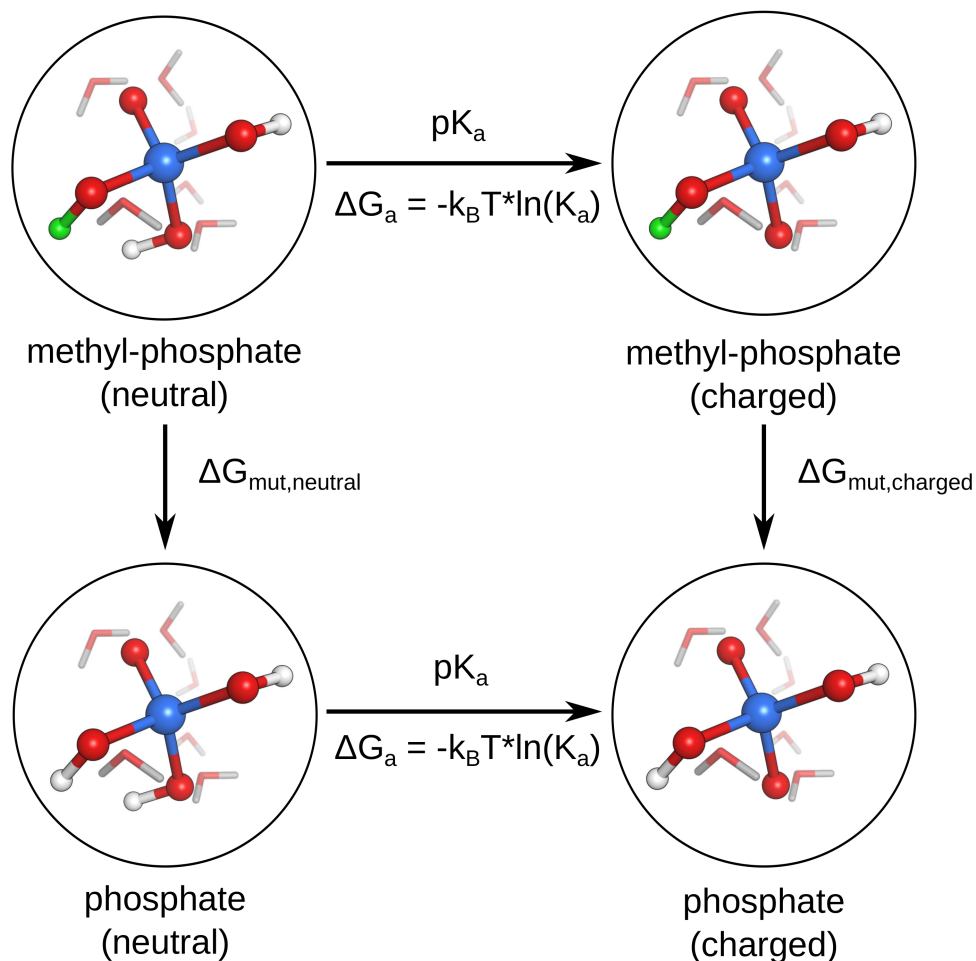


Figure 2.2: Alchemical changes from methyl-phosphate, in the neutral and charged ($-1e$) states into the respective phosphate compounds. The free energy of mutation of the neutral species has not been changed while the charged ΔG_{mut} has been adapted to get a cycle closure within $0.6 \frac{\text{kJ}}{\text{mol}}$. The horizontal free energies have been calculated from experimental pK_a values, as described in the main text.

2.3.1 Target values

In order to optimize our parameter set, a target value for the hydration free energy of dihydrogen phosphate had to be determined. Multiple values have been reported in literature (see table 2.1) and used before in force-field development [10].

Table 2.1: Collection of standard state dihydrogen phosphate hydration free energies from the literature.

source	$\Delta G_{\text{solv}}^{\ominus} [\frac{\text{kJ}}{\text{mol}}]$
Marcus [11]	-465
Li et al. [12]	-285
George et al. [13]	-318
This work (based on [14])	-436

We decided to base the target calculation on the conventional hydration entropy, ΔS_{conv}° , reported by Marcus and Loewenschuss [14], where the partial molar entropy of the aqueous proton has been set to zero (see *Results and discussion* for the rationale). By subtraction of the negative value of the absolute gas-phase proton entropy [14], the conventional hydration entropy, $\Delta S_{conv}^{\ominus}$, on our conventional scale (i.e. relative to the proton hydration entropy) can be calculated (equation 2.1). Note the additional factor of -1 to account for the opposite signs of the two involved species.

$$\begin{aligned}\Delta S_{conv}^{\ominus}[H_2PO_4^-] &= \Delta S_{conv}^{\circ}[H_2PO_4^-] - (-1)\Delta S_{conv}^{\circ}[H^+] \\ &= -188.2 \text{ J/mol/K} - (-1) * (-108.84 \text{ J/mol/K}) \quad (2.1) \\ &= -297.0 \text{ J/mol/K}\end{aligned}$$

In order to obtain the conventional hydration free energy, $\Delta_{hydr}G_{conv}^{\ominus}$, relative to the proton hydration free energy, the conventional enthalpy, $\Delta H_{conv}^{\ominus}$, reported in reference [15] (based on setting the proton hydration enthalpy to zero) can be used (shown in equation 2.2).

$$\begin{aligned}\Delta_{hydr}G_{conv}^{\ominus} &= \Delta H_{conv}^{\ominus} - T\Delta S_{conv}^{\ominus} \\ &= -1625 \text{ kJ/mol} - 298.15 \text{ K} * -297.0 \text{ J/mol/K}/1000 \quad (2.2) \\ &= -1536 \text{ kJ/mol}\end{aligned}$$

Finally, the intrinsic hydration free energy can be determined by subtracting the absolute intrinsic proton free energy of hydration (for GROMOS parameter set 54A8: $-1100 \frac{\text{kJ}}{\text{mol}}$), see equation 2.3.

$$\begin{aligned}\Delta_{hydr}G^{\ominus} &= \Delta_{hydr}G_{conv}^{\ominus}[H_2PO_4^-] - \Delta_{hydr}G_{conv}^{\ominus}[H^+] \\ &= -1536 \text{ kJ/mol} - (-1100 \text{ kJ/mol}) = -436 \text{ kJ/mol} \quad (2.3)\end{aligned}$$

Therefore, a target value of $-436 \frac{\text{kJ}}{\text{mol}}$ has been used as the absolute intrinsic dihydrogen phosphate hydration free energy.

For the other phosphate-based moieties, appropriate thermodynamic cycles have been constructed (see figure 2.2 for an example), with the optimized dihydrogen phosphate parameters as anchor point. The alchemical free energy changes (vertical arrows) were calculated by thermodynamic integration (TI, see example figure 2.3), slowly changing one compound into the other, see below.

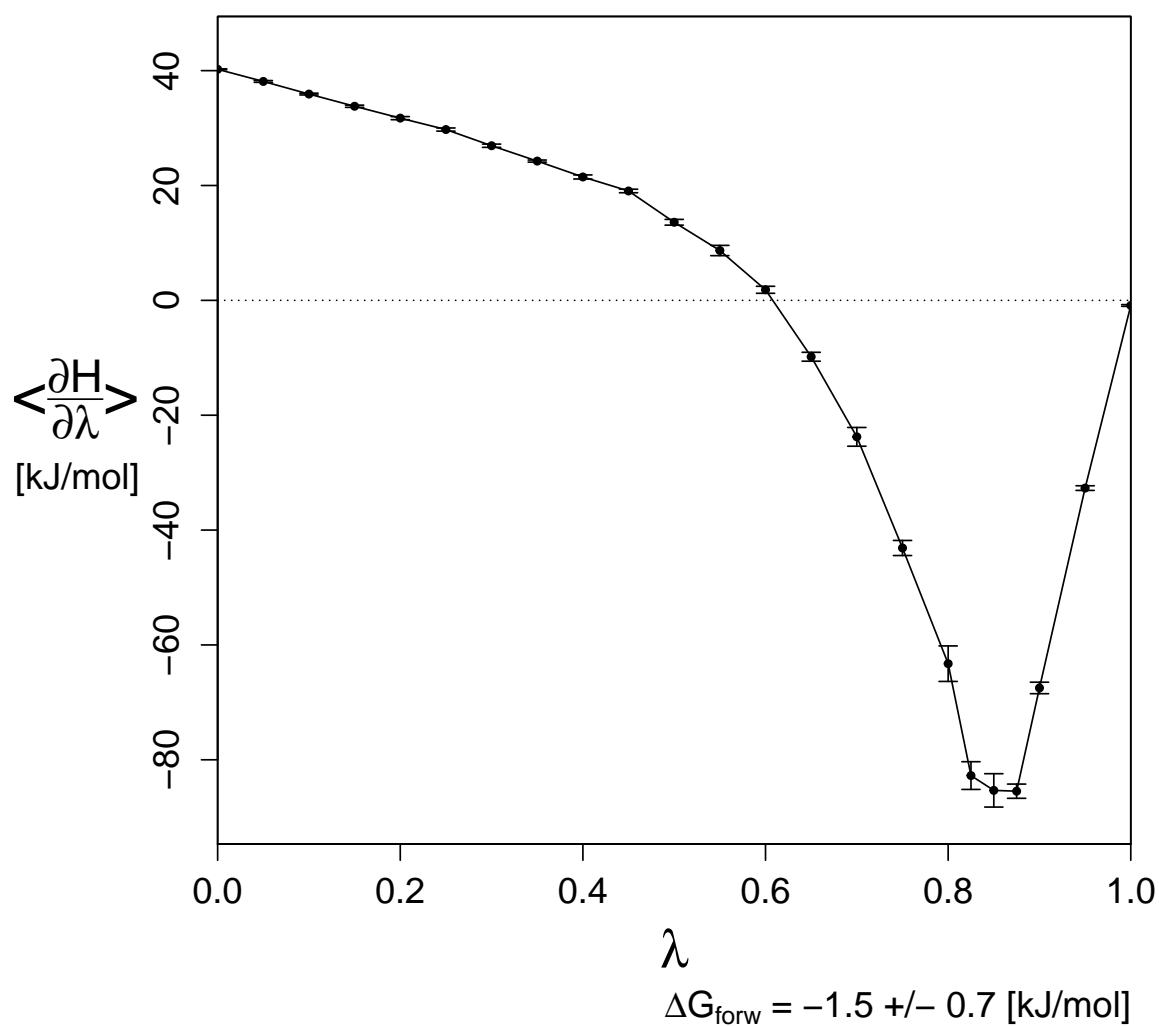


Figure 2.3: Example thermodynamic integration (TI) graph, as obtained for the cavity formation simulation of dihydrogen phosphate in water. This plot has been made by usage of the MDplot R package.

Since they do not involve a change in the net charge of the system, their calculation is straightforward. The horizontal free energies can be calculated from the experimental pK_a values. The partial charges of the charged phosphates have subsequently been changed such, that cycle closure is achieved within $0.6 \frac{\text{kJ}}{\text{mol}}$. As the pK_a is the negative, decadic logarithm of the association constant K_a as shown in equations 2.4 and 2.5. Considering the acid/base pair HA/A^- ,

$$K_a = \frac{[\text{A}^-][\text{H}_3\text{O}^+]}{[\text{HA}][\text{H}_2\text{O}]} \quad (2.4)$$

$$pK_a = -\log_{10} K_a = \frac{-\ln K_a}{\ln 10} \quad (2.5)$$

the natural logarithm of K_a can be calculated from the pK_a according to equation 2.6.

$$\ln K_a = -(\ln 10) * pK_a = -2.303 * pK_a \quad (2.6)$$

And finally, the corresponding free energy change can be calculated from the association constant, K_a , by applying equation 2.7, where R is the ideal gas constant (in kJ/mol).

$$\Delta G_a = -RT * \ln K_a = RT * 2.303 * pK_a \quad (2.7)$$

According to figure 2.2 the difference in the vertical arrows, $\Delta\Delta G_{\text{mut}}$, should be identical to the difference in the horizontal arrows, $\Delta\Delta G_a$, i.e.:

$$\Delta\Delta G_{\text{mut}} = 2.303 * RT * \Delta pK_a = \Delta G_{\text{mut,charged}} - \Delta G_{\text{mut,neutral}} \quad (2.8)$$

2.3.2 Simulation setup

All simulations have been performed using the GROMOS [16, 17] simulation package with the 54A7 [18] parameter set of the GROMOS force field or partial charge parameters as reported in table 2.2, implemented in the 54A7 parameter set. The compounds were placed in a water box in the absence of counter-ions. The water-boxes were initialized with a 1.6 nm minimum distance of the solute to the box walls, resulting in 1463 to 2101 water molecules (depending on the solute). Prior to the production simulations, the systems

were heated up to and equilibrated at 298.15 K and this temperature was maintained throughout. A weak thermostat-coupling with two baths for the solute and solvent, respectively (relaxation time, τ_T , equals 0.1 ps) and weak barostat-coupling (relaxation time, τ_P , of 0.5 ps and an isothermal compressibility of $7.51 \cdot 10^{-4} \text{ (kJ} \cdot \text{mol}^{-1} \cdot \text{nm}^{-3})^{-1}$, as appropriate for water at a temperature of 298.15 K and 1 bar) were applied [19]. The SHAKE algorithm [20] was used to maintain the bond lengths at the energy minimum. An integration time step of 2 fs was used, the configurations were stored every picosecond and the energies and Hamiltonian λ -derivatives every 0.5 picoseconds. Interactions within 1.4 nm were calculated at every timestep from a pairlist updated every step. Long range interactions were approximated with a reaction field [21] contribution to the energies and forces, accounting for a homogeneous medium with relative dielectric constant of 61 beyond the cut-off of 1.4 nm [22].

Table 2.2: Proposed partial charges of the four phosphate-containing moieties parametrized in this study. Dihydrogen phosphate has been parametrized first, targeting an experimental hydration free energy 2.1. The other ions have been parametrized relative to dihydrogen phosphate. The van-der-Waals interactions as well as the bonded interactions have not been changed from the 54A7 parameters.

	# atom	atom name	group	partial charge [e]
dihydrogen phosphate	1	O1	1	-0.635
	2	P		0.088
	3	O2		-0.635
	4	O3		-0.317
	5	HO3		0.408
	6	O4		-0.317
	7	HO4		0.408
methyl-phosphate	1	O1	1	-0.635
	2	P		0.052
	3	O2		-0.635
	4	O3		-0.255
	5	HO3		0.408
	6	O4		-0.210
	7	C4		0.275
di-methyl-phosphate	1	O1	1	-0.635
	2	P		0.140
	3	O2		-0.635
	4	O3		-0.210
	5	C3		0.275
	6	O4		-0.210
	7	C4		0.275

	# atom	atom name	group	partial charge [e]
phenyl-phosphate	1	C4	1	-0.140
	2	H4		0.140
	3	C5	2	-0.140
	4	H5		0.140
	5	C3	3	-0.140
	6	H3		0.140
	7	C6	4	-0.140
	8	H6		0.140
	9	C2	5	-0.140
	10	H2		0.140
	11	C1	6	0.330
	12	O1		-0.200
	13	P		-0.068
	14	O2		-0.635
	15	O3		-0.635
	16	O4		-0.200
	17	HO4		0.408

Free-energy differences were computed using the thermodynamic integration approach. A coupling parameter, λ , is introduced connecting the Hamiltonians of two states [23, 24]. The free-energy difference is subsequently computed using

$$\Delta G = \int_0^1 \left\langle \frac{\partial H}{\partial \lambda} \right\rangle d\lambda \quad (2.9)$$

, where H is the Hamiltonian of the system and angular brackets denote trajectory averaging. In practice, the integral is evaluated numerically, using a finite number of λ states. The number of λ states used varied from 32 to 38 depending on the curvature of the free-energy profile. The simulation time at the various λ points varied between 1 and 6 nanoseconds depending on the initial error estimates. Atoms experiencing large changes were described with soft-core interactions ($\alpha_{\text{LJ}}=0.5$; $\alpha_{\text{Coulomb}}=0.5 \text{ nm}^2$) [25]. Non-bonded interactions between the atoms within the phosphate group were excluded, which allows omission of a corresponding TI simulation in vacuum. We made the assumption, that the conformational sampling in vacuum and water of the ion is similar.

2.3.3 Methodology-independent free energies

The calculation of solvation free energies from microscopic molecular systems with effective electrostatic interactions inevitably suffers from methodology-dependent artefacts [9, 26]. Following the thermodynamic cycle in figure 2.1, the standard intrinsic solvation free energy $\Delta G_{solv}^{\ominus}$ is computed according to

$$\Delta G_{solv}^{\ominus} = \Delta G_{cav} + \Delta G_{chg}^{raw} + \Delta G_{cor} + \Delta G_{std}^{\ominus} \quad (2.10)$$

This standard solvation free energy is representative of the above-mentioned standard-state conditions.

2.3.3.1 Free energy of cavity formation

The free energy change associated with the formation of the cavity around a molecule, ΔG_{cav} , is due to the van-der-Waals (vdW) interactions of the (uncharged) molecule with the surrounding solvent. It can be calculated by an alchemical transformation of the molecule with all partial charges set to zero and full vdW interactions into a molecule completely deprived of both kinds of non-bonded interactions (see figure 2.1). To be compatible with the direction of the raw charging thermodynamic integration, this value has been multiplied by -1. From the difference of the box volumes of this thermodynamic integration at $\lambda=0$ and $\lambda=1$, an estimate for the effective volume of the dihydrogen phosphate ion was obtained, which is required for the calculation of corrections to the charging free energy (see below).

2.3.3.2 Raw charging free energy

The raw charging free energy, ΔG_{chg}^{raw} , can be obtained from a thermodynamic integration changing the molecule with all partial charges set to zero into the fully charged state, while the van-der-Waals (vdW) interactions remain fully installed throughout. However, this free energy is dependent on the used simulation parameters, most explicitly on the scheme to compute the electrostatic interactions, and in order to obtain a method-independent value, certain corrections have to be applied [8]. Note, that the correction scheme employed here assumes that the raw charging free energy is exempt of ion self-term contributions [9].

2.3.3.3 Free energy contributions of corrections

Here, we separate the corrections into two contributions previously described as ΔG_{A+B+D} and ΔG_{C_1} [8, 9]:

$$\Delta G_{cor} = \Delta G_{A+B+D} + \Delta G_{C_1} \quad (2.11)$$

The first correction term consists of three individual contributions (see reference [8]). Type A corrections attribute to incorrect solvent polarisation around the ion and incomplete

interactions of the ion with the solvent due to non-Coulombic electrostatic interactions schemes. Type B addresses finite size and artificial periodicity artifacts of the simulated system. Type D corrects for the inaccurate dielectric permittivity of the solvent model. The contribution to ΔG_{A+B+D} can be calculated according to equation 2.12.

$$\Delta G_{A+B+D} = \Delta G_{chg}^{NPBC(FD)} - \Delta G_{chg}^{PBC,RF(FFT)} + \Delta G_{chg}^{PBC,LS(FFT)} - \Delta G_{chg}^{PBC,LS(FD)} \quad (2.12)$$

Where ΔG_{chg} are Poisson-Boltzmann estimates of the charging free energy under periodic (PBC) or non-periodic (NPBC) boundary conditions based on an implicit-solvent representation of the system, computed using a finite difference (FD) or fast-fourier transform (FFT) solver and a reaction field (RF) or lattice sum (LS) approximation to the Coulombic interactions [8]. For details concerning the employed parameters, see reference [3].

The C type corrections [27] are necessary to account for the error in the potential at the ion site induced here by usage of a reaction-field in combination with molecule-based cutoff truncation. They are calculated according to equation 2.13 [26]:

$$\Delta G_{C_1} = -q_I \frac{2(\varepsilon_{BW} - 1)}{2\varepsilon_{BW} + 1} * \left(1 - \frac{V_I}{(4/3)\pi R_C^3} \right) \xi'_S \quad (2.13)$$

where V_I is the ionic volume, q_I the full ion charge ($-1 e$), ε_{BW} is the reaction-field permittivity (61), R_C is the cut-off distance (1.4 nm) and ξ'_S is the exclusion potential of the solvent model (here, $77.4 \text{ kJ mol}^{-1} e^{-1}$ was used as the value [3]).

To reach standard conditions, ΔG_{std}^\ominus is the free energy corresponding to the standard-state conversion, i.e. from the simulated system to a system of 1 bar pressure in the gas phase and 1 molal concentration in the aqueous phase. For hydration in the context of molecular dynamics simulations and the above described standard state conditions, it amounts to $7.95 \frac{\text{kJ}}{\text{mol}}$ [9].

2.3.4 One step perturbation

In order to test multiple parameter sets for their match to the experimental values, instead of running multiple simulations, one-step perturbation predictions were applied. From a single simulation obtained under the regime of a given Hamiltonian A, this procedure allows to predict e.g. the hydration free energy, using a different Hamiltonian B, without the need for another thermodynamic integration calculation [28].

In general, the accuracy of such predictions is even quantitative [29], as long as the phase-space overlap between the respective ensembles of states A and B is sufficient.

$$\Delta G_B = \Delta G_A + \Delta G_{BA} = \Delta G_A - k_B T * \ln \langle e^{-(H_B - H_A)/k_B T} \rangle_A \quad (2.14)$$

In practice this means that the effect of slight modifications of the partial charges can be predicted quite efficiently.

2.4 Results and discussion

In the paper of Steinbrecher et al. [10], the authors based their parametrization on reproducing the difference between the solvation free energies of the neutral and charged species, together with the pK_a and gas-phase basicity in a Pearson cycle [30]. We have taken a slightly different approach and directly computed the standard intrinsic hydration free energy of the dihydrogen phosphate ion, and subsequently used this as an anchor point to compute relative free energies to the other species, using the thermodynamic cycle in figure 2.2 and experimentally determined pK_a values. Still, for dihydrogen phosphate an experimentally determined absolute intrinsic hydration free energy is required. To our knowledge, there are three different values proposed in literature (table 2.1). Steinbrecher et al. used a value of $-318 \frac{\text{kJ}}{\text{mol}}$ reported by George [13]. However, this is reported as a hydration energy (using ΔH^0 as a symbol), rather than a free energy. The value reported by Li [12] is similar, but the one by Marcus [11] deviates significantly. In their publication on theoretical methods, Li et al. report $-285 \frac{\text{kJ}}{\text{mol}}$, without giving a source (probably an adaption of George et al.). Remains the value by Marcus ($-465 \frac{\text{kJ}}{\text{mol}}$) reported in reference [11], presumably based on earlier publications [14, 15]. However, this value seems to stem from the combination of the conventional hydration entropy with an absolute hydration enthalpy, which is inconsistent. Instead, we calculated what we believe a better solution according to the description in the *Methods* section, leading to a target value of the dihydrogen phosphates' standard absolute intrinsic free energy of hydration of $-436 \frac{\text{kJ}}{\text{mol}}$.

The standard absolute intrinsic hydration free energy of dihydrogen phosphate when using our previous parameters (54A7), was computed to be $-348.4 \frac{\text{kJ}}{\text{mol}}$. This is too hydrophobic by approximately $88 \frac{\text{kJ}}{\text{mol}}$. In order to find parameters providing a better match with experimental data, we adjusted the partial charges (see below).

2.4.1 Parametrization

A major requirement in the parametrization process was the maintenance of the hydrogen bonding potential of the atoms, primarily the most solvent-exposed ones. To achieve that goal, the partial charges of the hydrogen atom and the doubly bound oxygen atoms have been maintained to be $+0.408 e$ and $-0.635 e$, respectively, in accordance with previous force-field parameter sets [18], while the charges of the other atoms were allowed to change during the optimization procedure. First, the charges on the hydroxy-oxygen atoms and the carbon atoms have been set such that the target value was approached. The partial charge on the phosphorus atom was always adjusted set to make the net charge of the compound amount to $-1 e$. Since the four oxygen atoms shield the phosphorus atom due to

their tetrahedral conformation from most direct interactions with the solvent molecules, changes in its partial charge do not affect the compounds' behaviour significantly.

2.4.2 Newly proposed parameters

The reported parameters for dihydrogen phosphate are listed in table 2.2. The sum of contributions to the free energy of solvation, as shown in table 2.3, lead to a ΔG_{solv} of $-434.9 \frac{\text{kJ}}{\text{mol}}$ for our parameters. This value is in quite good agreement with the experimental target value of $-436 \frac{\text{kJ}}{\text{mol}}$, especially compared to the value obtained for the original 54A7 parameter set (see above and supplementary table 2.6 for details) which was too hydrophobic.

Table 2.3: List of free energy contributions to the standard state solvation free energy (equation 2.10) for the dihydrogen phosphate simulation using the new parameters. For a description of the individual terms, see the main text.

term	$\text{H}_2\text{PO}_4^- [\frac{\text{kJ}}{\text{mol}}]$
ΔG_{cav}	1.5 ± 0.7
$\Delta G_{\text{chg}}^{\text{raw}}$	-439 ± 1
$\Delta G_{\text{A+B+D}}$	-80.2
ΔG_{C_1}	74.9
$\Delta G_{\text{std}}^{\ominus}$	7.95
$\Delta G_{\text{solv}}^{\ominus}$	-434.9 ± 1.7

For the other compounds, the pK_a value for p-tolyl phosphate used by Steinbrecher et al. could not be traced, therefore we used the value for phenyl-phosphate reported by Chanley and Feageson [31]. The thermodynamic cycles are reported in table 2.4.

Table 2.4: List of free energy changes (in $\frac{\text{kJ}}{\text{mol}}$) calculated from the three thermodynamic cycles used, together with the (absolute) cycle closure deviation. The first and last values of each series have been calculated from experimental pK_a values. The thermodynamic cycle used for methyl-phosphate is depicted as an example in figure 2.2. In the first column, X represents one of the phosphate derivatives, while P represents dihydrogen phosphate.

	methyl-phosphate	di-methyl-phosphate	phenyl-phosphate
$\text{HX} \rightarrow \text{X}^-$	8.4 [*]	7.5 [*]	5.7 [*]
$\text{X}^- \rightarrow \text{P}^-$	-14.2 ± 0.7	-32.0 ± 1.0	1.0 ± 1.0
$\text{HX} \rightarrow \text{HP}$	-17.0 ± 1.0	-36.0 ± 1.0	-4.0 ± 1.0
$\text{HP} \rightarrow \text{P}^-$	11.3 [*]	11.3 [*]	11.3 [*]
cycle	0.1	0.2	0.6

^{*} calculated from the respective experimental pK_a using equation 2.7 (phosphate: 2.7 [32], methyl-phosphate: 2.0 [33], di-methyl-phosphate: 1.8 [32] and phenyl-phosphate: 1.0 [31])

The other phosphate moieties have been parametrized to minimize the absolute deviations in these cycles (see table 2.4), which have maximum values of $0.6 \frac{\text{kJ}}{\text{mol}}$.

The post-translational modifications listed in table 2.5 (parameters in the supplementary table 2.7) have been parametrized according to the approach published earlier [6], making use of previously established parameters for small charged molecules and the newly derived ones for phosphates provided by this study.

Table 2.5: List of post-translationally modified amino acids that have been updated in the context of this study. The phosphate parameters have been taken from the parametrization described in this work, while parameters for the other groups have been proposed in the publication of Reif et al. [3]. The nomenclature of the three-letter abbreviations follows the one previously described by Petrov et al. [6] in 2013.

name (charge [e])	basis	abbreviation
phosphoarginine (0)	ARG	R0P
omega-N-methylarginine (+1)	ARG	RMC
symmetric-dimethylarginine (+1)	ARG	RMS
asymmetric-dimethylarginine (+1)	ARG	RMA
3-hydroxyaspartate (-1,S)	ASP	D3H
3-hydroxyaspartate (-1,R)	ASP	DH3
phosphoaspartate (-1)	ASP	D1P
4-carboxyglutamate (-1)	GLU	ECN
4-carboxyglutamate (-2)	GLU	ECA
3-methylhistidine (+1)	HIS	H3C
1-phosphohistidine (-1)	HIS	H11
3-phosphohistidine (-1)	HIS	H31
1-methylhistidine (+1)	HIS	H1C
5-hydroxylysine (+1,S)	LYS	KHP
phospholysine (-1)	LYS	K1P
N6,N6,N6-trimethyllysine	LYS	K3C
N6,N6-dimethyllysine (+1)	LYS	K2C
N6-methyllysine	LYS	KMC
carboxyllysine (-1)	LYS	KCA
5-hydroxylysine (+1,R)	LYS	KPH
phosphoserine (-1)	SER	S1P
phosphothreonine (-1)	THR	T1P
phosphotyrosine (-1)	TYR	Y1P

The PTM-parameters are available both in GROMOS and two versions of the GROMACS format at the Vienna-PTM webserver <http://vienna-ptm.univie.ac.at>.

2.5 Conclusion

In the current work we have developed new phosphate and post-translational modification parameters in order to improve their physico-chemical representation in molecular dynamics simulations using the GROMOS force field. The phosphate parameters have been calibrated to reproduce experimentally determined standard absolute intrinsic hydration free energies. To this end, raw calculated charging free energies were transformed into standard intrinsic free energies of hydration by application of correction terms. The calibrated species are dihydrogen phosphate ($-1\ e$), methyl-phosphate ($-1\ e$), di-methyl-phosphate ($-1\ e$) and phenyl-phosphate ($-1\ e$). Subsequently, the parameters have been used, together with parameters recently proposed for other charged moieties, to assemble an updated post-translational modification parameter set for GROMOS force fields. In the case of phosphorylation, the change in hydration free energy is large and it is thus likely to affect simulations significantly. This newly proposed parameter set will hopefully be of great use in further simulation studies involving charged post-translational modifications of proteins or certain lipids and might lead to an update of the nucleic acids' backbone parameters.

2.6 Appendix / Supplementary material

Table 2.6: List of free energy contributions to the standard state solvation free energy (equation 2.10) for the dihydrogen phosphate simulation using the new parameters. For a description of the individual terms, see the main text.

term	H_2PO_4^- [$\frac{\text{kJ}}{\text{mol}}$]
ΔG_{cav}	1.5 ± 0.7
$\Delta G_{\text{chg}}^{\text{raw}}$	-354 ± 1
$\Delta G_{\text{A+B+D}}$	-78.7
ΔG_{C_1}	74.9
$\Delta G_{\text{std}}^{\ominus}$	7.95
$\Delta G_{\text{solv}}^{\ominus}$	-348.4 ± 1.7

Table 2.7: List of parameters for the protein post-translational modifications reparametrized in this study.

	# atom	atom name	group	partial charge [e]
D1P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.253
	6	OD1		-0.450
	7	OD2		-0.069
	8	PE		0.383
	9	OZ1		-0.635
	10	OZ2		-0.635
	11	OZ3		-0.255
	12	HZ3		0.408
D3H / DH3	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB	3	0.266
	5	OG1		-0.674
	6	HG1		0.408
	7	CG2	4	0.430
	8	OD1		-0.715
	9	OD2		-0.715
ECA	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG		0.000
	6	CD1	3	0.430
	7	OE1		-0.715
	8	OE2		-0.715
	9	CD2	4	0.430
	10	OE3		-0.715
	11	OE4		-0.715

	# atom	atom name	group	partial charge [e]
ECN	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG		0.000
	6	CD1	3	0.330
	7	OE1		-0.450
	8	OE2		-0.288
	9	HE2	4	0.408
	10	CD2		0.430
	11	OE3		-0.715
	12	OE4		-0.715
H1C	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.140
	6	ND1		-0.030
	7	HD1		0.320
	8	CD2		0.060
	9	HD2		0.140
	10	CE1		0.060
	11	HE1		0.140
	12	NE2		-0.030
	13	CZ		0.200

	# atom	atom name	group	partial charge [e]
H31	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	ND1		0.000
	7	CD2	4	0.130
	8	HD2		0.140
	9	CE1		0.130
	10	HE1		0.140
	11	NE2		-0.540
	12	PE3	5	0.117
	13	OZ1		-0.635
	14	OZ2		-0.635
	15	OZ3		-0.255
	16	HZ3		0.408
K1P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD		0.000
	7	CE	4	0.000
	8	NZ	5	-0.310
	9	HZ		0.310
	10	PH	6	0.117
	11	OI1		-0.635
	12	OI2		-0.635
	13	OI3		-0.255
	14	HI3		0.408

	# atom	atom name	group	partial charge [e]
KCA	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD		0.000
	7	CE	4	0.000
	8	NZ	5	-0.310
	9	HZ		0.310
	10	CH	6	0.430
	11	OI1		-0.715
	12	OI2		-0.715
KHP / KPH	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD	4	0.266
	7	OE1		-0.674
	8	HE1		0.408
	9	CE2	5	0.200
	10	NZ		0.050
	11	HZ1		0.250
	12	HZ2		0.250
	13	HZ3		0.250
KMC	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD		0.000
	7	CE	4	0.200
	8	NZ		0.100
	9	HZ1		0.250
	10	HZ2		0.250
	11	CH		0.200

	#	atom	atom name	group	partial charge [e]
RMA	1		N	1	-0.310
	2		H		0.310
	3		CA	2	0.000
	4		CB		0.000
	5		CG		0.000
	6		CD	3	0.090
	7		NE		-0.105
	8		HE		0.345
	9		CZ		0.210
	10		NH1		-0.360
	11		HH11		0.345
	12		HH12		0.345
	13		NH2		-0.050
	14		CT1		0.090
	15		CT2		0.090
RMC	1		N	1	-0.310
	2		H		0.310
	3		CA	2	0.000
	4		CB		0.000
	5		CG		0.000
	6		CD	3	0.090
	7		NE		-0.105
	8		HE		0.345
	9		CZ		0.010
	10		NH1		-0.360
	11		HH11		0.345
	12		HH12		0.345
	13		NH2		-0.105
	14		HH2		0.345
	15		CT		0.090

	# atom	atom name	group	partial charge [e]
T1P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.275
	5	OG1	3	-0.210
	6	PD		0.052
	7	OE1		-0.635
	8	OE2		-0.635
	9	OE3		-0.255
	10	HE3		0.408
	11	CG2	4	0.000
Y1P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG		0.000
	6	CD1	3	-0.140
	7	HD1		0.140
	8	CD2	4	-0.140
	9	HD2		0.140
	10	CE1	5	-0.140
	11	HE1		0.140
	12	CE2	6	-0.140
	13	HE2		0.140
	14	CZ	7	0.330
	15	OH		-0.200
	16	PT		-0.068
	17	OI1		-0.635
	18	OI2		-0.635
	19	OI3		-0.200
	20	HI3		0.408

	# atom	atom name	group	partial charge [e]
H3C	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.200
	6	ND1		-0.030
	7	CE3		0.200
	8	CD2		0.000
	9	HD2		0.140
	10	CE1		0.060
	11	HE1		0.140
	12	NE2		-0.030
	13	HE2		0.320
H11	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	ND1		-0.540
	7	CD2		0.130
	8	HD2		0.140
	9	CE1		0.130
	10	HE1		0.140
	11	NE2		0.000
	12	PZ	4	0.117
	13	OH1		-0.635
	14	OH2		-0.635
	15	OH3		-0.255
	16	HH3		0.408

	# atom	atom name	group	partial charge [e]
K2C	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD		0.000
	7	CE	4	0.200
	8	NZ		0.150
	9	HZ		0.250
	10	CH1		0.200
	11	CH2		0.200
K3C	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG	3	0.000
	6	CD		0.000
	7	CE	4	0.200
	8	NZ		0.200
	9	CH1		0.200
	10	CH2		0.200
	11	CH3		0.200

	# atom	atom name	group	partial charge [e]
R0P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG		0.000
	6	CD		0.090
	7	NE		-0.105
	8	HE		0.345
	9	CZ		0.010
	10	NH1	3	-0.360
	11	HH11		0.345
	12	HH12		0.345
	13	NH2		-0.015
	14	HH2		0.345
	15	PT	4	0.117
	16	OI1		-0.635
	17	OI2		-0.635
	18	OI3		-0.255
	19	HI3		0.408
RMS	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB		0.000
	5	CG		0.000
	6	CD		0.090
	7	NE		-0.105
	8	HE		0.345
	9	CZ		0.010
	10	NH1	3	-0.105
	11	HH1		0.345
	12	CT1		0.090
	13	NH2		-0.105
	14	HH2		0.345
	15	CT2		0.090

	# atom	atom name	group	partial charge [e]
S1P	1	N	1	-0.310
	2	H		0.310
	3	CA	2	0.000
	4	CB	3	0.275
	5	OG		-0.210
	6	PD		0.052
	7	OE1		-0.635
	8	OE2		-0.635
	9	OE3		-0.255
	10	HE3		0.408

References

- [1] Chris Oostenbrink, Alessandra Villa, Alan E. Mark, and Wilfred F. van Gunsteren. “A biomolecular force field based on the free enthalpy of hydration and solvation: the GROMOS force-field parameter sets 53A5 and 53A6”. In: *Journal of Computational Chemistry* 25.13 (Oct. 2004), pp. 1656–1676. DOI: [10.1002/jcc.20090](https://doi.org/10.1002/jcc.20090).
- [2] Devleena Shivakumar, Edward Harder, Wolfgang Damm, Richard A. Friesner, and Woody Sherman. “Improving the Prediction of Absolute Solvation Free Energies Using the Next Generation OPLS Force Field”. In: *Journal of Chemical Theory and Computation* 8.8 (Aug. 2012), pp. 2553–2558. DOI: [10.1021/ct300203w](https://doi.org/10.1021/ct300203w).
- [3] Maria M. Reif, Philippe H. Hünenberger, and Chris Oostenbrink. “New Interaction Parameters for Charged Amino Acid Side Chains in the GROMOS Force Field”. In: *Journal of Chemical Theory and Computation* 8.10 (Oct. 2012), pp. 3705–3723. DOI: [10.1021/ct300156h](https://doi.org/10.1021/ct300156h).
- [4] Maria M. Reif, Moritz Winger, and Chris Oostenbrink. “Testing of the GROMOS Force-Field Parameter Set 54A8: Structural Properties of Electrolyte Solutions, Lipid Bilayers, and Proteins”. In: *Journal of Chemical Theory and Computation* 9.2 (Feb. 2013), pp. 1247–1264. DOI: [10.1021/ct300874c](https://doi.org/10.1021/ct300874c).
- [5] The UniProt Consortium. “UniProt: a hub for protein information”. In: *Nucleic Acids Research* 43.D1 (Jan. 2015), pp. D204–D212. DOI: [10.1093/nar/gku989](https://doi.org/10.1093/nar/gku989).
- [6] Drazen Petrov, Christian Margreitter, Melanie Grandits, Chris Oostenbrink, and Bojan Zagrovic. “A Systematic Framework for Molecular Dynamics Simulations of Protein Post-Translational Modifications”. In: *PLoS Computational Biology* 9.7 (July 2013), e1003154. DOI: [10.1371/journal.pcbi.1003154](https://doi.org/10.1371/journal.pcbi.1003154).
- [7] Christian Margreitter, Drazen Petrov, and Bojan Zagrovic. “Vienna-PTM web server: a toolkit for MD simulations of protein post-translational modifications”. In: *Nucleic Acids Research* 41.Web Server issue (July 2013), W422–426. DOI: [10.1093/nar/gkt416](https://doi.org/10.1093/nar/gkt416).
- [8] Maria M. Reif and Philippe H. Hünenberger. “Computation of methodology-independent single-ion solvation properties from molecular simulations. IV. Optimized Lennard-Jones interaction parameter sets for the alkali and halide ions in water”. In: *The Journal of Chemical Physics* 134.14 (Apr. 2011), p. 144104. DOI: [10.1063/1.3567022](https://doi.org/10.1063/1.3567022).
- [9] Mika A. Kastenholtz and Philippe H. Hünenberger. “Computation of methodology-independent ionic solvation free energies from molecular simulations. II. The hydration free energy of the sodium cation”. In: *The Journal of Chemical Physics* 124.22 (June 2006), p. 224501. DOI: [10.1063/1.2201698](https://doi.org/10.1063/1.2201698).
- [10] T. Steinbrecher, J. Latzer, and D. A. Case. “Revised AMBER Parameters for Bioorganic Phosphates”. In: *Journal of Chemical Theory and Computation* 8.11 (Nov. 2012), pp. 4405–4412. DOI: [10.1021/ct300613v](https://doi.org/10.1021/ct300613v).
- [11] Yizhak Marcus. “Thermodynamics of solvation of ions. Part 5.-Gibbs free energy of hydration at 298.15 K”. In: *Journal of the Chemical Society, Faraday Transactions* 87.18 (Jan. 1991), pp. 2995–2999. DOI: [10.1039/FT9918702995](https://doi.org/10.1039/FT9918702995).
- [12] Jiabo Li, Tianhai Zhu, Gregory D. Hawkins, Paul Winget, Daniel A. Liotard, Christopher J. Cramer, and Donald G. Truhlar. “Extension of the platform of applicability of the SM5.42R universal solvation model”. In: *Theoretical Chemistry Accounts* 103.1 (1999), pp. 9–63. DOI: [10.1007/s002140050513](https://doi.org/10.1007/s002140050513).

- [13] Philip George, Robert J. Witonsky, Mendel Trachtman, Clara Wu, William Dorwart, Linda Richman, William Richman, Fahd Shurayh, and Barry Lentz. “Squiggle-H₂O. An enquiry into the importance of solvation effects in phosphate ester and anhydride reactions”. In: *Biochimica et Biophysica Acta (BBA) - Bioenergetics* 223.1 (Nov. 1970), pp. 1–15. DOI: [10.1016/0005-2728\(70\)90126-X](https://doi.org/10.1016/0005-2728(70)90126-X).
- [14] Y. Marcus and A. Loewenschuss. “Chapter 4. Standard entropies of hydration of ions”. In: *Annual Reports Section "C" (Physical Chemistry)* 81.0 (Jan. 1984), pp. 81–135. DOI: [10.1039/PC9848100081](https://doi.org/10.1039/PC9848100081).
- [15] Yizhak Marcus. “The thermodynamics of solvation of ions. Part 2.-The enthalpy of hydration at 298.15 K”. In: *Journal of the Chemical Society, Faraday Transactions 1: Physical Chemistry in Condensed Phases* 83.2 (Jan. 1987), pp. 339–349. DOI: [10.1039/F19878300339](https://doi.org/10.1039/F19878300339).
- [16] Nathan Schmid, Clara D. Christ, Markus Christen, Andreas P. Eichenberger, and Wilfred F. van Gunsteren. “Architecture, implementation and parallelisation of the GROMOS software for biomolecular simulation”. In: *Computer Physics Communications* 183.4 (Apr. 2012), pp. 890–903. DOI: [10.1016/j.cpc.2011.12.014](https://doi.org/10.1016/j.cpc.2011.12.014).
- [17] Markus Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: [10.1002/jcc.20303](https://doi.org/10.1002/jcc.20303).
- [18] Nathan Schmid, Andreas P. Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E. Mark, and Wilfred F. van Gunsteren. “Definition and testing of the GROMOS force-field versions 54A7 and 54B7”. In: *European Biophysics Journal* 40.7 (July 1, 2011), pp. 843–856. DOI: [10.1007/s00249-011-0700-9](https://doi.org/10.1007/s00249-011-0700-9).
- [19] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81.8 (Oct. 15, 1984), pp. 3684–3690. DOI: [10.1063/1.448118](https://doi.org/10.1063/1.448118).
- [20] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J. C Berendsen. “Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes”. In: 23.3 (Mar. 1977), pp. 327–341. DOI: [10.1016/0021-9991\(77\)90098-5](https://doi.org/10.1016/0021-9991(77)90098-5).
- [21] Ilario G. Tironi, René Sperb, Paul E. Smith, and Wilfred F. van Gunsteren. “A generalized reaction field method for molecular dynamics simulations”. In: *The Journal of Chemical Physics* 102.13 (Apr. 1, 1995), pp. 5451–5459. DOI: [10.1063/1.469273](https://doi.org/10.1063/1.469273).
- [22] Tim N. Heinz, Wilfred F. van Gunsteren, and Philippe H. Hünenberger. “Comparison of four methods to compute the dielectric permittivity of liquids from molecular dynamics simulations”. In: *The Journal of Chemical Physics* 115.3 (July 15, 2001), pp. 1125–1136. DOI: [10.1063/1.1379764](https://doi.org/10.1063/1.1379764).
- [23] J.G. Kirkwood. “Statistical mechanics of fluid mixtures”. In: *J. Chem. Phys.* (1935), pp. 300–313.
- [24] Sereina Riniker, Luzi J. Barandun, Francois Diederich, Oliver Krämer, Andreas Steffen, and Wilfred F. van Gunsteren. “Free enthalpies of replacing water molecules in protein binding pockets”. In: *Journal of Computer-Aided Molecular Design* 26.12 (Dec. 2012), pp. 1293–1309. DOI: [10.1007/s10822-012-9620-8](https://doi.org/10.1007/s10822-012-9620-8).

- [25] Thomas C. Beutler, Alan E. Mark, Rene C. van Schaik, Paul R. Gerber, and Wilfred F. van Gunsteren. “Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations”. In: *Chemical Physics Letters* 222.6 (June 1994), pp. 529–539. DOI: [10.1016/0009-2614\(94\)00397-1](https://doi.org/10.1016/0009-2614(94)00397-1).
- [26] Philippe Hünenberger and Maria Reif. *Single-Ion Solvation: Experimental and Theoretical Approaches to Elusive Thermodynamic Quantities*. 1st ed. Royal Society of Chemistry, 2011.
- [27] M. A. Kastenholtz and Philippe H. Hünenberger. “Computation of methodology-independent ionic solvation free energies from molecular simulations. I. The electrostatic potential in molecular liquids”. In: *The Journal of Chemical Physics* 124.12 (Mar. 2006), p. 124106. DOI: [10.1063/1.2172593](https://doi.org/10.1063/1.2172593).
- [28] Robert W. Zwanzig. “HighTemperature Equation of State by a Perturbation Method. I. Nonpolar Gases”. In: *The Journal of Chemical Physics* 22.8 (Aug. 1954), pp. 1420–1426. DOI: [10.1063/1.1740409](https://doi.org/10.1063/1.1740409).
- [29] Zhixiong Lin, Chris Oostenbrink, and Wilfred F. van Gunsteren. “On the use of one-step perturbation to investigate the dependence of NOE-derived atom-atom distance bound violations of peptides upon a variation of force-field parameters”. In: *European Biophysics Journal* 43.2-3 (Feb. 2014), pp. 113–119. DOI: [10.1007/s00249-014-0943-3](https://doi.org/10.1007/s00249-014-0943-3).
- [30] Ralph G. Pearson. “Ionization potentials and electron affinities in aqueous solution”. In: *Journal of the American Chemical Society* 108.20 (Oct. 1986), pp. 6109–6114. DOI: [10.1021/ja00280a002](https://doi.org/10.1021/ja00280a002).
- [31] J. D. Chanley and Edward Feageson. “The Mechanism of the Hydrolysis of Organic Phosphates. III. Aromatic Phosphates¹”. In: *Journal of the American Chemical Society* 77.15 (Aug. 1955), pp. 4002–4007. DOI: [10.1021/ja01620a015](https://doi.org/10.1021/ja01620a015).
- [32] W. D. Kumler and John J. Eiler. “The Acid Strength of Mono and Diesters of Phosphoric Acid. The n-Alkyl Esters from Methyl to Butyl, the Esters of Biological Importance, and the Natural Guanidine Phosphoric Acids”. In: *Journal of the American Chemical Society* 65.12 (Dec. 1943), pp. 2355–2361. DOI: [10.1021/ja01252a028](https://doi.org/10.1021/ja01252a028).
- [33] C. A. Bunton, D. R. Llewellyn, K. G. Oldham, and C. A. Vernon. “The reactions of organic phosphates. Part I. The hydrolysis of methyl dihydrogen phosphate”. In: *Journal of the Chemical Society (Resumed)* 0 (Jan. 1958), pp. 3574–3587. DOI: [10.1039/JR9580003574](https://doi.org/10.1039/JR9580003574).

On the optimization of the protein backbone dihedral angles by means of Hamiltonian reweighting

Adapted from:

Margreitter C., Oostenbrink C., *On the Optimization of the Protein Backbone Dihedral Angles by Means of Hamiltonian Reweighting*, J. Chem. Inf. Model. (2016), online, doi: <https://dx.doi.org/10.1021/acs.jcim.6b00399>

Author contributions:

CM performed and analyzed all simulations and developed the necessary code, CO derived the project, supervised the work and assisted in writing the manuscript.

The applicability and usefulness of molecular dynamics (MD) simulations is directly dependent on the method's capability of reproducing the physico-chemical properties of various compounds properly. Therefore, the parametrization of force fields is a pivotal and often tedious task, alternating between the definition of new parameter sets and their subsequent validation. As new experimental data become available, additional checks might be performed to ensure the force field's broad scope of application. In the context of GROMOS, the parametrization usually relies on the reproduction of experimental data obtained for small molecules such as ethanol, assuming their principle transferability to moieties in bigger systems such as proteins. These properties include the heat of vaporization, hydration free energies and phase distributions. However, until now the deduction of the backbone dihedral angles of amino acids was mainly based on chemical intuition and subsequent refinement during application. Moreover, all amino acid species shared the same dihedral potentials which might be a just assumption for most of them but, as we will show, fails in cases where amino acids differ in their side-chain structures at C_β . Even though target data is available, it remains very difficult to find a set of backbone energy potentials which concomitantly satisfies all experimental values. For this reason we tried to cover a very wide region of this multi-dimensional optimization problem which becomes only possible by replacing the otherwise necessary trial simulations by quantitatively correct predictions made using the Hamiltonian reweighting approach.

3.1 Abstract

Molecular dynamics simulations depend critically on the accuracy of the underlying force fields in properly representing biomolecules. Hence, it is crucial to validate the force field parameter sets in this respect. In the context of GROMOS, this is usually achieved by comparing simulation data to experimental observables for small molecules. In this study, we develop new amino acid backbone dihedral angle energy potentials based on the widely used 54A7 parameter set by matching to experimental J-values and secondary structure propensity scales, respectively. In order to find the most appropriate backbone parameters, close to 100000 different combinations of potentials have been screened. However, since the sheer number of the combinations considered prohibits actual molecular dynamics simulations for each of them, we instead predicted the values for every combination using Hamiltonian reweighting. While the original 54A7 parameter set fails to reproduce the experimental data, we are able to provide parameters that match significantly better. However, to ensure applicability in the context of larger peptides and full proteins, further studies have to be undertaken.

3.2 Introduction

The functional forms that are the basis of virtually all classical force fields are very similar. In general, the interactions between atoms comprising biological macromolecules, which are in principle resulting from their electrons that should be described with quantum mechanics. In a force field these interactions are approximated by the assignment to various well-known and simpler equations, for example harmonic oscillators to account for covalent bonds. The parameters that are used in these simplifications, e.g. force constants and minimum distances, have been subjected to repeated adjustments. Depending on the computational resources that are available, the model (i.e. force field) gets more detailed and accurate but also much more complex. Moreover, these parameters are connected and changes in one might require adaption of others as well. Furthermore, as accessible simulation times increase, comparisons to more extensive experimental data become realistic, potentially requiring further updates of the force field parameters. Another reason for the evolution of force field parameters over time is the availability of new experimental data which allows further testing and revalidations. In this context, we attempted to test the protein backbone 54A7 [1] parameters against experimental data and, if necessary, to find better solutions. In the GROMOS parametrization philosophy, the parametrization relies on the reproduction of experimental data obtained for small molecules such as ethanol, assuming their principle transferability to moieties in bigger systems such as proteins. In contrast, the torsion angles in the protein backbone have been assigned by chemical intuition and subsequent refinement at the peptide and protein level [1]. As experimental reference values, we chose NMR-derived coupling constants ($^3J(\text{HN}, \text{H}_\alpha)$) [2] and secondary structure propensities from Raman spectroscopy for all canonical dipeptides [3]. The former has been used in this kind of analysis before [4, 5], but especially the calculation of the J-value from the ϕ -backbone dihedral angle by using the Karplus equation is rather inaccurate. Furthermore, the J-coupling constant bear only very limited information on the ψ -backbone dihedral angle. Thus, the inclusion of the secondary structure information from the latter is a highly relevant addition.

As will become clear, the backbone parameters currently in use are not suited to describe the characteristics of the dipeptides, failing in both dimensions of our target values. In

order to find proper parameters, we screened close to 100000 parameter combinations for the 20 amino acids in question thereby covering a wide range of possible solutions. It is noteworthy, that we did not bias our search space except by the selection of the initial parameters. Because of the vast number of combinations considered, we used Hamiltonian reweighting to predict the likely outcome of an actual simulation for a given set of parameters. Provided that the initial simulations using the 54A7 parameters show a sufficient overlap with the (hypothetical) target ones, it is possible to quantitatively predict the ensemble averages of structural properties which are to be matched to the experiments.

3.3 Methods

3.3.1 Model systems

In the current work, we have focused on the backbone dihedral angles of the blocked amino acids, which are formed by blocking the termini of a single amino acid through acetylation of the N-terminus and N-methylation at the C-terminus. Because this compound now has two dipeptide bonds, it is commonly referred to as "dipeptide". Figure 3.1 shows the alanine dipeptide. The blocking of the termini is usually applied to exclude interactions between the charged ends and the side-chain of the central amino acid, especially in the case of charged side-chains. However, it has been reported [6], that blocking with an acetyl- and N-methyl-group, respectively, does not significantly change the intrinsic propensities, which in turn means that the influence of charged ends is negligible. However, to stay as close to the experiments as possible, we added an acetyl- and N-methyl-group to the amino acids as well. The protonation states of the amino acids were set to match these in the respective experiments: histidine, lysine and arginine are positively charged while aspartate and glutamate are negatively charged.

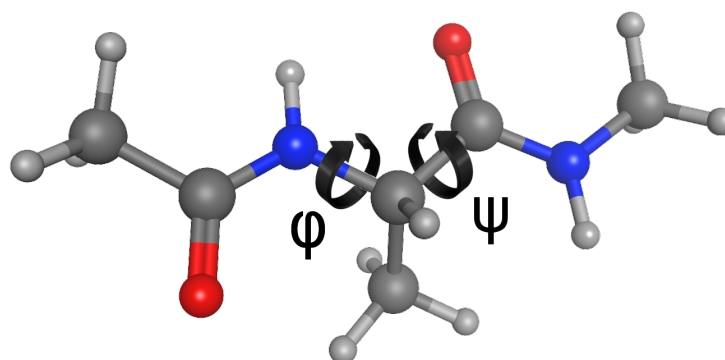


Figure 3.1: Graphical representation of the two backbone dihedral angles in the model system. The angles ϕ and ψ are defined by atoms C-N-C $_{\alpha}$ -C and N-C $_{\alpha}$ -C-N, respectively. Note, that the amino acids have been (in accordance to the experimental studies) blocked: an acetyl-group at the N-terminus and a methyl moiety at the C-terminus are used to ensure non-charged ends.

3.3.2 Simulation setup

The simulations have been performed using the GROMOS [7, 8] simulation package with the 54A7 [1] parameter set of the GROMOS force field or the backbone parameters, reported in tables 3.1 and S3.5, implemented in the 54A7 set. The compounds were placed in a water box in the absence of counter-ions. Force field parameters for the blocked termini are listed in table S3.6. The water-boxes were initialized with a 1.4 nm minimum distance of the solute to the box walls. Prior to the production simulations, the systems were equilibrated from 60 K to 300 K in five discrete steps with a simulation length of 20 ps each. All simulations were carried out at 300 K and at a constant pressure of 1 atm, respectively, unless explicitly stated differently. A weak thermostat-coupling with two baths for the solute and solvent, respectively (τ equals 0.1) and weak barostat-coupling (relaxation time of 0.5 ps and an isothermal compressibility of $4.575 \cdot 10^{-4} (\text{kJ} \cdot \text{mol}^{-1} \cdot \text{nm}^{-3})^{-1}$) were applied [9]. The SHAKE algorithm [10] was used to maintain the bond distances at the energy minimum. An integration time step of 2 fs was used and the configurations were stored every picosecond. Interactions within 0.8 nm were calculated at every timestep from a pairlist that was updated every five steps. Intermediate range interactions up to a distance of 1.4 nm were calculated at pairlist updates and kept constant between updates. Long range interactions were approximated with a reaction field [11] contribution to the energies and forces, accounting for a homogeneous medium with relative dielectric constant [12] of 61 beyond the cut-off of 1.4 nm. The lengths of the trajectories are 100 ns. In all cases, 100% of the trajectories were used for analysis.

Table 3.1: Backbone parameters of the GROMOS force field and those of the suggested set. All other combinations mentioned in the text are provided in table S3.5.

combination	backbone angle potential-energy functions						description
	ϕ			ψ			
	K [$\frac{\text{kJ}}{\text{mol}}$]	shift [$^\circ$]	mult.	K [$\frac{\text{kJ}}{\text{mol}}$]	shift [$^\circ$]	mult.	
45A3*/ 53A6*	1.0	180.0	6	1.0	0.0	6	see [13], [14]
54A7 / 54A8	2.8	0.0	3	3.5	180.0	2	see [1]
	0.7	180.0	6	0.4	0.0	6	
#81883	1.0	180.0	2	3.0	180.0	2	glycine
	3.0	180.0	1	5.0	180.0	1	
#12572	5.0	0.0	3	1.0	180.0	2	alanine
	5.0	180.0	6	1.0	180.0	3	
#5623	3.0	180.0	2	1.0	0.0	1	common amino acids
	5.0	0.0	3	1.0	180.0	3	
#86516	3.0	0.0	3	5.0	0.0	1	C_β -branched
	5.0	180.0	6	5.0	0.0	2	

* For these parameter sets, only one potential-energy term has been used to describe the ϕ and ψ backbone dihedral angles, respectively.

3.3.3 Comparison to experimental data

The first step in remodelling the energy potentials of the backbone dihedral angles in amino acids, is the sound selection of experimental data to compare to. In the case of this study, we chose the J-value measurements of the "dipeptide" (blocked amino acids) series published by Avbelj and co-workers [2]. This observable represents a time average dependent on the ϕ -angle in the protein backbone. In conjunction, we also included the subsequently published propensity scale by Grdadolnik and co-workers [3] as our experimental target values (see table S3.7). These two studies form a consistent set, since the J-values from the first study have been used to calibrate the Raman spectroscopy measurements in the latter.

In general, there are two ways to define and distinguish secondary structure elements in peptides and proteins. First, the hydrogen bond pattern of the backbone is often typical and is used for classification, for example by the widely used Define Secondary Structure of Proteins (DSSP) [15] algorithm. This approach requires, however, a minimum fragment length of at least four, which is needed in order to recognize e.g. an α -helical conformation. For this reason, this approach cannot be used for the small compounds investigated in this study. The second possibility is to make use of the dihedral angles comprising the Ramachandran plots, where certain areas indicate backbone conformations associated with secondary structure elements [16]. Recently, the DIhedral-based Segment Identification and CLassification (DISICL) [17, 18] toolbox has been developed by our group. DISICL is able to process and annotate both proteins and nucleic acids in terms of their secondary structure, based on two consecutive pairs of dihedral angles. Because the current systems investigated are very small, we have simplified the DISICL secondary structure region definitions (see table S3.8). Consequently, in the current work we make no distinction between different types of helices (α , 3_{10} - and π -helices), whose basins in the Ramachandran plot are next to each other. This simplification allows comparisons to the experimental data used as a target. Note, that other secondary structure elements have been neglected because they are not covered by the experimental study used. In contrast to Hollingsworth' original definitions [19], we define the ϕ -angle representing the border between the β and P_{II} basins to be at -100° , which is in closer agreement with the average of the basin centres in the experimental study of Grdadolnik et al. [3] and the observed distributions. Moreover, this value is a compromise of the values used in literature (ranging from -110° to -90° , see references [20–23]). Since the Raman experiments used as a target are based on a fitting procedure, reporting propensities that sum up to 100%, we normalized our results to account for that. This approach avoids locking our systems in the three basins, where other or unclassified regions in the Ramachandran plot might also be visited occasionally and allows for a better comparability. However, in all cases but glycine, which is further discussed below, this renormalization did not affect the best set of parameters nor did it influence the overall score significantly. Unclassified regions

typically show an occupancy below a value of about 5%.

One critical experimental observable in the context of investigations on small peptide systems is $^3J(\text{HN}, \text{H}_\alpha)$. This coupling constant as determined by NMR represents an ensemble average and is connected to the $\text{HN-N-C}_\alpha\text{-H}_\alpha$ dihedral angle. It can be calculated according to the Karplus equation as given in equation 3.1. However, the results depend strongly on the choice of the three parameters A, B and C in this equation. We chose the Pardi parameters ($A=6.4$, $B=-1.4$ and $C=1.9$) [24] which have been used for the protein backbone in the context of GROMOS studies before [25, 26]. Since H_α is not represented by an actual atom in the GROMOS force field, we approximate the $\text{HN-N-C}_\alpha\text{-H}_\alpha$ dihedral angle by subtracting 60 degrees from ϕ , making use of the planarity on the peptide bond and the tetrahedral coordination at C_α . It is noteworthy, that comparisons of J-values from experiments with their equivalents calculated from MD simulations are only possible with limited accuracy [27] (typically about 0.5 Hertz).

$$^3J(\text{HN}, \text{H}_\alpha) = A * \cos^2(\phi - 60) - B * \cos(\phi - 60) + C \quad (3.1)$$

In order to compare the experimental results with our predicted propensities and J-values,

the deviation for the J-values (given in Hertz) and the percentages (given in fractional propensities) are weighted equally, meaning that the deviations of the J-value and the three secondary structure basins are summed up (see below).

3.3.4 Hamiltonian reweighting

As will be seen in the results section, the 54A7 parameter set did not reproduce the experimentally determined propensities and J-values very well. We therefore attempted a systematic re-parametrization of the backbone dihedral angle parameters.

In order to determine the effect various torsional angle parameters have on the properties used for calibration, the simplest approach is to perform multiple molecular dynamics simulations in a brute-force fashion. However, the potential number of combinations is vast (roughly 100000 combinations, see table 3.2) and this is beyond reach. Instead a

Table 3.2: All combinations used for the reweighting workflow. Every angle is described by either one or two potential-energy functions. The possible values for the parameters of equation 3.3 are given in brackets. We accounted for the possibility that one potential energy term might suffice by adding a force constant of zero to the second potential energy terms. The total number of unique combinations sums up to 97344. This number consists of $(3^2 * 2^2 * 4^2)/2 = 288$ combinations with two potential energy terms and $(3^1 * 2^1 * 4^1) = 24$ combinations with one potential energy term summing up to 312 for each of the angles. Combining these parameter combination for both the ϕ and ψ angles leads to $312^2 = 97344$ combinations in our set.

	force constants	shifts	multiplicities
ϕ	{1,3,5}	{0,180}	{1,2,3,6}
	{0,1,3,5}	{0,180}	{1,2,3,6}
ψ	{1,3,5}	{0,180}	{1,2,3,6}
	{0,1,3,5}	{0,180}	{1,2,3,6}

one-step perturbation protocol has been used to predict the observables of interest from one single simulation using the current GROMOS 54A7 parameter set. The overall workflow is described in figure 3.2. First, the timeseries of the experimental observables, i.e. the J-value and the propensities as well as the dihedral angles ϕ and ψ (see figure 3.1), were calculated from simulations which have been performed for the 20 canonical amino acids using the 54A7 parameter set. Afterwards, the timeseries of the Hamiltonian for a given parameter set, as well as the reference (54A7) values were calculated based on the backbone dihedral angles as well. By reweighting the ensemble to the updated Hamiltonian, we were able to project the ensemble average values for the individual quantities, Q , which were afterwards compared to the target values and ranked by their experimental match. Equation 3.2 goes back to the umbrella sampling technique by Torrie and Valleau [28] and is widely used to obtain ensemble averages for Hamiltonians that are slightly different from the simulated one.

$$\langle Q \rangle_A = \frac{\langle Q e^{-(H_A - H_R)k_B T} \rangle_R}{\langle e^{-(H_A - H_R)k_B T} \rangle_R} \quad (3.2)$$

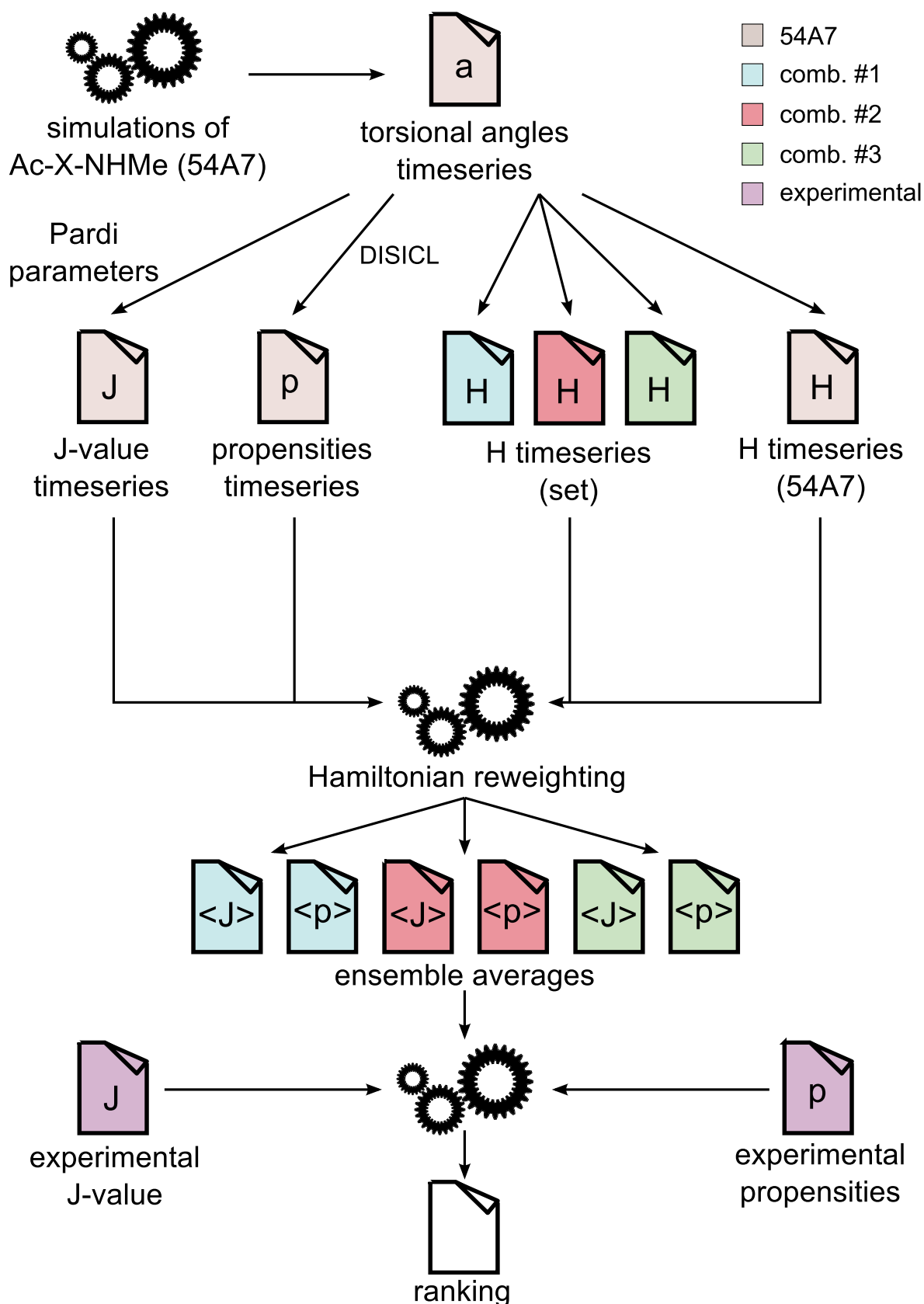


Figure 3.2: Workflow applied for Hamiltonian reweighting. From the configurations sampled in the simulations, the ensemble averages of dependent observables have been predicted. For the calculations of the J-values and Hamiltonians see equations 3.1 and 3.3, respectively. The considerations regarding the ranking of solutions is further described in the methods section of the paper.

Here, H_A and H_R represent the Hamiltonians of a parameter set A and of the reference parameters R . $k_B T$ is the Boltzmann constant multiplied by the absolute temperature and the angular brackets indicate an ensemble average obtained from a simulation using the reference Hamiltonian.

The actual calculation of the Hamiltonians from the dihedral angle time series (figure 3.2) consists in our case of the summation over the four backbone dihedral angle energy potentials (see equation 3.3).

$$H = \sum_{i=0}^n k_i * (\cos(s_i) * \cos(m_i * \phi) + 1) + \sum_{j=0}^r k_j * (\cos(s_j) * \cos(m_j * \psi) + 1) \quad (3.3)$$

The variables k , s and m are the force constants (either 1, 3 or 5 $\frac{kJ}{mol}$), the shifts (either 0° or 180°) and the multiplicities, i.e. the number of minima (either 1, 2, 3 or 6), respectively. Since there are multiple torsional potentials per angle possible, variables n and r represent the number of individual ϕ and ψ potentials, respectively. Our set of potential combinations was built by one or two torsional potentials per angle (see table 3.2). For every combination, the timeseries of the Hamiltonian was calculated using the combination's parameters and the ensemble average of the observables Q was calculated using equation 3.2.

The ranking of parameter sets is described in more detail below. All graphs showing the results have been composed by the R packages `MDplot` (recently published) and `RPrometheus`. It has been shown, that these predictions are quite accurate (even quantitatively) provided that a significant part of the sampled configurations are relevant both for the reference and the target ensembles [29]. In order to ensure that this approach is accurate enough we have predicted the J-value and the propensities for three randomly chosen forcefield parameter sets using our 54A7 trajectories (see figure S3.8). It is noteworthy that we are able to project the values within a few percent of inaccuracy, though the dihedral angle parameters between the two parameter sets are significantly different (see table 3.1). Moreover, we report also the actual simulated values in table 3.3 for the selected parameter sets for comparison.

Table 3.3: Detailed results of the predicted and simulated parameter combinations. In brackets, the deviation to the experimental target values is reported: negative values indicate that the computational values are too high and vice versa for positive ones. The combination given in italic letters indicates the best hit when optimizing the amino acids individually and thus gives the best hit possible with our screening set of parameters.

		J-value [Hz]	propensities [%/100]			Δ^a	
			α	β	P_{II}	abs.	ren.
GLY	54A7 ^b	5.1 (+0.7)	0.014 (+0.646)	0.241 (-0.121)	0.076 (+0.144)	1.651	1.976
	#81883	5.8 (+0.0)	0.210 (+0.450)	0.062 (+0.058)	0.032 (+0.188)	0.736	0.270
	#81883 ^b	5.9 (+0.0)	0.221 (+0.439)	0.058 (+0.062)	0.030 (+0.190)	0.701	0.256
	#5623	4.4 (+1.4)	0.002 (+0.658)	0.202 (-0.082)	0.008 (+0.212)	2.362	3.076
ALA	54A7 ^b	7.9 (-1.8)	0.111 (-0.001)	0.388 (-0.098)	0.444 (+0.156)	2.045	2.048
	#12572	6.3 (-0.2)	0.114 (-0.004)	0.259 (+0.031)	0.592 (+0.008)	0.283	0.283
	#12572 ^b	6.2 (-0.1)	0.127 (-0.017)	0.239 (+0.051)	0.597 (+0.003)	0.211	0.224
	#5623	7.1 (-1.1)	0.039 (+0.071)	0.530 (-0.240)	0.375 (+0.225)	1.616	1.623
ARG	54A7 ^b	7.8 (-0.9)	0.156 (-0.086)	0.291 (+0.099)	0.497 (+0.043)	1.168	1.131
	#5623	7.1 (-0.2)	0.067 (+0.003)	0.306 (+0.084)	0.583 (-0.043)	0.370	0.380
	#5623 ^b	7.1 (-0.2)	0.080 (-0.010)	0.303 (+0.087)	0.565 (-0.025)	0.362	0.380
	#67656	6.9 (+0.0)	0.116 (-0.046)	0.259 (+0.131)	0.392 (+0.148)	0.355	0.192
ASN	54A7 ^b	8.4 (-1.0)	0.000 (+0.020)	0.590 (-0.010)	0.387 (+0.013)	1.003	1.008
	#5623	7.8 (-0.3)	0.000 (+0.020)	0.613 (-0.033)	0.346 (+0.054)	0.437	0.448
	#5623 ^b	7.8 (-0.3)	0.000 (+0.020)	0.619 (-0.039)	0.341 (+0.059)	0.438	0.450
	#96795	7.5 (+0.0)	0.000 (+0.020)	0.600 (-0.020)	0.389 (+0.011)	0.061	0.064
ASP ^c	54A7 ^b	7.7 (-0.8)	0.093 (-0.043)	0.324 (+0.136)	0.517 (-0.027)	0.996	1.017
	#5623	7.0 (-0.1)	0.045 (+0.005)	0.372 (+0.088)	0.533 (-0.043)	0.216	0.222
	#5623 ^b	7.0 (+0.0)	0.056 (-0.006)	0.354 (+0.106)	0.533 (-0.043)	0.185	0.199
	#31541	7.0 (-0.1)	0.045 (+0.005)	0.383 (+0.077)	0.541 (-0.051)	0.213	0.217
CYS	54A7 ^b	8.0 (-0.7)	0.179 (-0.149)	0.381 (+0.159)	0.393 (+0.037)	1.005	0.976
	#5623	7.3 (+0.0)	0.055 (-0.025)	0.488 (+0.052)	0.408 (+0.022)	0.139	0.096
	#5623 ^b	7.3 (+0.1)	0.038 (-0.008)	0.488 (+0.052)	0.427 (+0.003)	0.113	0.106
	#50989	7.3 (+0.0)	0.029 (+0.001)	0.531 (+0.009)	0.424 (+0.006)	0.026	0.012
GLU ^c	54A7 ^b	7.6 (-1.0)	0.126 (-0.076)	0.264 (+0.096)	0.559 (+0.031)	1.153	1.116
	#5623	6.9 (-0.2)	0.057 (-0.007)	0.263 (+0.097)	0.644 (-0.054)	0.378	0.394
	#5623 ^b	6.8 (-0.2)	0.046 (+0.004)	0.269 (+0.091)	0.652 (-0.062)	0.357	0.368
	#87099	6.6 (+0.1)	0.091 (-0.041)	0.224 (+0.136)	0.435 (+0.155)	0.382	0.192
GLN	54A7 ^b	8.2 (-1.0)	0.000 (+0.080)	0.489 (-0.009)	0.491 (-0.051)	1.160	1.180
	#5623	7.5 (-0.4)	0.000 (+0.080)	0.499 (-0.019)	0.468 (-0.028)	0.507	0.540
	#5623 ^b	7.1 (+0.1)	0.064 (+0.016)	0.295 (+0.185)	0.597 (-0.157)	0.448	0.458
	#83856	7.1 (+0.0)	0.000 (+0.080)	0.462 (+0.018)	0.535 (-0.095)	0.193	0.194
HIS ^c	54A7 ^b	8.0 (-0.1)	0.211 (-0.171)	0.331 (+0.249)	0.397 (-0.017)	0.567	0.585
	#5623	7.4 (+0.5)	0.078 (-0.038)	0.383 (+0.197)	0.470 (-0.090)	0.835	0.848
	#5623 ^b	7.4 (+0.5)	0.060 (-0.020)	0.421 (+0.159)	0.446 (-0.066)	0.735	0.742
	#89861	7.9 (+0.0)	0.040 (+0.000)	0.250 (+0.330)	0.154 (+0.226)	0.586	0.130
ILE	54A7 ^b	7.5 (-0.2)	0.118 (-0.098)	0.237 (+0.283)	0.603 (-0.143)	0.714	0.735
	#86516	7.6 (-0.3)	0.005 (+0.015)	0.502 (+0.018)	0.452 (+0.008)	0.311	0.299
	#86516 ^b	7.7 (-0.3)	0.015 (+0.005)	0.509 (+0.011)	0.427 (+0.033)	0.369	0.350
	#5623	6.7 (+0.6)	0.042 (-0.022)	0.145 (+0.375)	0.797 (-0.337)	1.374	1.386
	#28191	7.3 (+0.0)	0.016 (+0.004)	0.455 (+0.065)	0.504 (-0.044)	0.123	0.124

		J-value [Hz]	propensities [%/100]			Δ^a	
			α	β	P_{II}	abs.	ren.
LEU	54A7 ^b	7.6 (-0.7)	0.174 (-0.074)	0.214 (+0.136)	0.562 (-0.012)	0.902	0.93
	#5623	6.9 (+0.0)	0.077 (+0.023)	0.213 (+0.137)	0.667 (-0.117)	0.277	0.294
	#5623 ^b	6.9 (+0.0)	0.047 (+0.053)	0.231 (+0.119)	0.683 (-0.133)	0.335	0.352
	#9315	6.9 (+0.0)	0.105 (-0.005)	0.321 (+0.029)	0.493 (+0.057)	0.101	0.039
LYS ^c	54A7 ^b	7.8 (-1.0)	0.153 (-0.113)	0.293 (+0.117)	0.496 (+0.054)	1.254	1.215
	#5623	7.1 (-0.3)	0.066 (-0.026)	0.307 (+0.103)	0.579 (-0.029)	0.428	0.445
	#5623 ^b	7.1 (-0.3)	0.071 (-0.031)	0.297 (+0.113)	0.554 (-0.004)	0.408	0.436
	#80612	6.8 (+0.1)	0.028 (+0.012)	0.328 (+0.082)	0.583 (-0.033)	0.207	0.222
MET	54A7 ^b	7.8 (-0.8)	0.152 (-0.122)	0.288 (+0.182)	0.471 (+0.029)	1.133	1.108
	#5623	7.1 (-0.1)	0.059 (-0.029)	0.261 (+0.209)	0.528 (-0.028)	0.346	0.405
	#5623 ^b	7.1 (-0.1)	0.044 (-0.014)	0.307 (+0.163)	0.603 (-0.103)	0.370	0.386
	#28751	7.1 (-0.1)	0.031 (-0.001)	0.399 (+0.071)	0.433 (+0.067)	0.249	0.126
PHE	54A7 ^b	8.2 (-1.0)	0.133 (-0.073)	0.43 (+0.060)	0.381 (+0.069)	1.232	1.191
	#5623	7.5 (-0.3)	0.048 (+0.012)	0.473 (+0.017)	0.433 (+0.017)	0.316	0.290
	#5623 ^b	7.4 (-0.3)	0.046 (+0.014)	0.467 (+0.023)	0.441 (+0.009)	0.306	0.284
	#25538	7.2 (+0.0)	0.086 (-0.026)	0.390 (+0.100)	0.444 (+0.006)	0.142	0.142
PRO	54A7 ^b	6.6	0.256	0.002	0.729	-	-
	#5623 ^b	5.0	0.112	0.000	0.857	-	-
SER	54A7 ^b	8.0 (-1.0)	0.195 (-0.155)	0.389 (+0.081)	0.370 (+0.120)	1.316	1.288
	#5623	7.3 (-0.3)	0.050 (-0.010)	0.545 (-0.075)	0.347 (+0.143)	0.518	0.534
	#5623 ^b	7.3 (-0.3)	0.043 (-0.003)	0.564 (-0.094)	0.333 (+0.157)	0.554	0.572
	#67652	7.0 (+0.0)	0.036 (+0.004)	0.382 (+0.088)	0.479 (+0.011)	0.103	0.088
THR	54A7 ^b	7.4 (-0.1)	0.407 (-0.377)	0.137 (+0.443)	0.421 (-0.031)	0.891	0.916
	#86516	7.3 (+0.0)	0.037 (-0.007)	0.410 (+0.170)	0.464 (-0.074)	0.281	0.290
	#86516 ^b	7.3 (+0.0)	0.052 (-0.022)	0.409 (+0.171)	0.434 (-0.044)	0.257	0.266
	#5623	6.6 (+0.8)	0.192 (-0.162)	0.110 (+0.470)	0.655 (-0.265)	1.647	1.680
	#92991	7.4 (-0.1)	0.021 (+0.009)	0.462 (+0.118)	0.475 (-0.085)	0.262	0.262
TRP	54A7 ^b	8.0 (-1.1)	0.121 (-0.101)	0.401 (+0.039)	0.424 (+0.116)	1.366	1.326
	#5623	7.3 (-0.4)	0.051 (-0.031)	0.461 (-0.021)	0.451 (+0.089)	0.521	0.524
	#5623 ^b	7.3 (-0.4)	0.041 (-0.021)	0.472 (-0.032)	0.449 (+0.091)	0.504	0.507
	#93571	6.9 (+0.0)	0.008 (+0.012)	0.391 (+0.049)	0.572 (-0.032)	0.103	0.108
TYR	54A7 ^b	8.3 (-1.2)	0.114 (-0.044)	0.473 (-0.003)	0.356 (+0.104)	1.351	1.365
	#5623	7.6 (-0.5)	0.041 (+0.029)	0.505 (-0.035)	0.411 (+0.049)	0.533	0.536
	#5623 ^b	7.5 (-0.4)	0.031 (+0.039)	0.498 (-0.028)	0.429 (+0.031)	0.498	0.500
	#6094	7.1 (+0.0)	0.088 (-0.018)	0.399 (+0.071)	0.470 (-0.010)	0.099	0.106
VAL	54A7 ^b	7.5 (-0.2)	0.124 (-0.104)	0.233 (+0.277)	0.601 (-0.131)	0.712	0.733
	#86516	7.6 (-0.3)	0.010 (+0.010)	0.499 (+0.011)	0.445 (+0.025)	0.356	0.337
	#86516 ^b	7.7 (-0.4)	0.003 (+0.017)	0.527 (-0.017)	0.428 (+0.042)	0.486	0.490
	#5623	6.7 (+0.6)	0.052 (-0.032)	0.141 (+0.369)	0.787 (-0.317)	1.328	1.342
	#24957	7.3 (+0.0)	0.015 (+0.005)	0.457 (+0.053)	0.509 (-0.039)	0.107	0.108

^a The overall Δ is the sum of the absolute values of the deviation in Hz for the J-value and the discrepancies in the propensities, see equation 3.4. Column ⟨abs.⟩ refers to the deviation when using the absolute occurrences of the three secondary structure classes while column ⟨ren.⟩ refers to the average deviations when the propensities are first renormalized to 100%.

^b These values have been computed from real simulations and not been projected.

^c In terms of protonation states, we used GROMOS parameters for HISH (+1 charged doubly protonated), LYSH (+1 charge, protonated) and the dissociated versions of glutamic and aspartic acid (GLU and ASP).

3.3.5 Ranking

The vast number of potential candidates for the backbone parameters (see table 3.2) and the multi-dimensional optimization space make the identification of one single best combination highly unlikely. Thus, we instead aimed for a good compromise between contrary optimization goals to ensure applicability in different contexts. Our selection protocol relied on the prediction of the J-value and the propensities for every combination based on the 54A7 trajectories. This task was performed by an R library developed for this purpose. Multidimensional optimization invariably require a weighting of the various properties. Here, we define an overall deviation, Δ , as reported in equation 3.4 where w is the respective weight and Δ_J , ΔP_α , ΔP_β and $\Delta P_{P_{II}}$ are the deviations for the J-value (in Hz) and the α , β and P_{II} propensities, respectively. Unless stated otherwise, we used $w_J = 1 \text{ Hz}^{-1}$ and $w_\alpha = w_\beta = w_{P_{II}} = 1$.

$$\Delta = w_J * |\Delta J| + w_\alpha * |\Delta P_\alpha| + w_\beta * |\Delta P_\beta| + w_{P_{II}} * |\Delta P_{P_{II}}| \quad (3.4)$$

The candidate combinations of dihedral angle parameters were subsequently sorted to identify a promising subset of the first (approximately) 100 results. The mean values of Δ over multiple amino acids were taken when appropriate. These have been plotted in a scatter plot (exemplified by figure S3.9) and the ones not showing concerted performance in both dimensions (J-values and summed up propensity deviations) have been removed. Usually, an "arch" is obtained as no combination perfectly reproduces both experiments simultaneously. In addition, we tried to find solutions that optimized all amino acids within a subgroup using barplots with detailed information (see figure 3.3). For the remaining candidates, the potential-energy functions have been plotted (see figure S3.5) to identify and exclude those with very high energy barriers (roughly over $20 \frac{\text{kJ}}{\text{mol}}$). The candidate sets have been further analyzed by predicting a low resolution Ramachandran distribution (with $15 \times 15 = 225$ bins) using equation 3.2 in order to visualize any uncommon peaks or very narrow distributions (figure S3.10). In our opinion, low and smooth energy surfaces and broader distributions are generally preferable for these backbone parameters, because they allow for a certain freedom compared to those "locking" the backbone very tightly into a few narrow minima. Therefore, in cases where our other criteria matched (almost) equally well, we opted for these alternatives.

There are two special amino acids, that should be considered more closely: glycine and proline. For glycine, the experimental propensities are not necessarily representative of the real distribution because of its known broad sampling of the Ramachandran plot [30]. Hence, its renormalized propensity values differ tremendously from the calculated ones. Moreover, glycine is known to show a strong bias towards the left-helical region of the Ramachandran plot. In this case, we aimed for a rather homogeneous distribution of the ϕ - ψ -angles, in agreement with distributions observed in the PDB (see reference [30]). Since no data was available for proline, we decided to apply the parameters retrieved for the common amino acids and check the resulting distribution afterwards, i.e. proline was not included in the optimization procedure. Furthermore, histidine was excluded from the ranking procedure, because the experimental data was reported to be rather uncertain [2, 3]. Including histidine in the selection only slightly shifted the overall results (not shown).

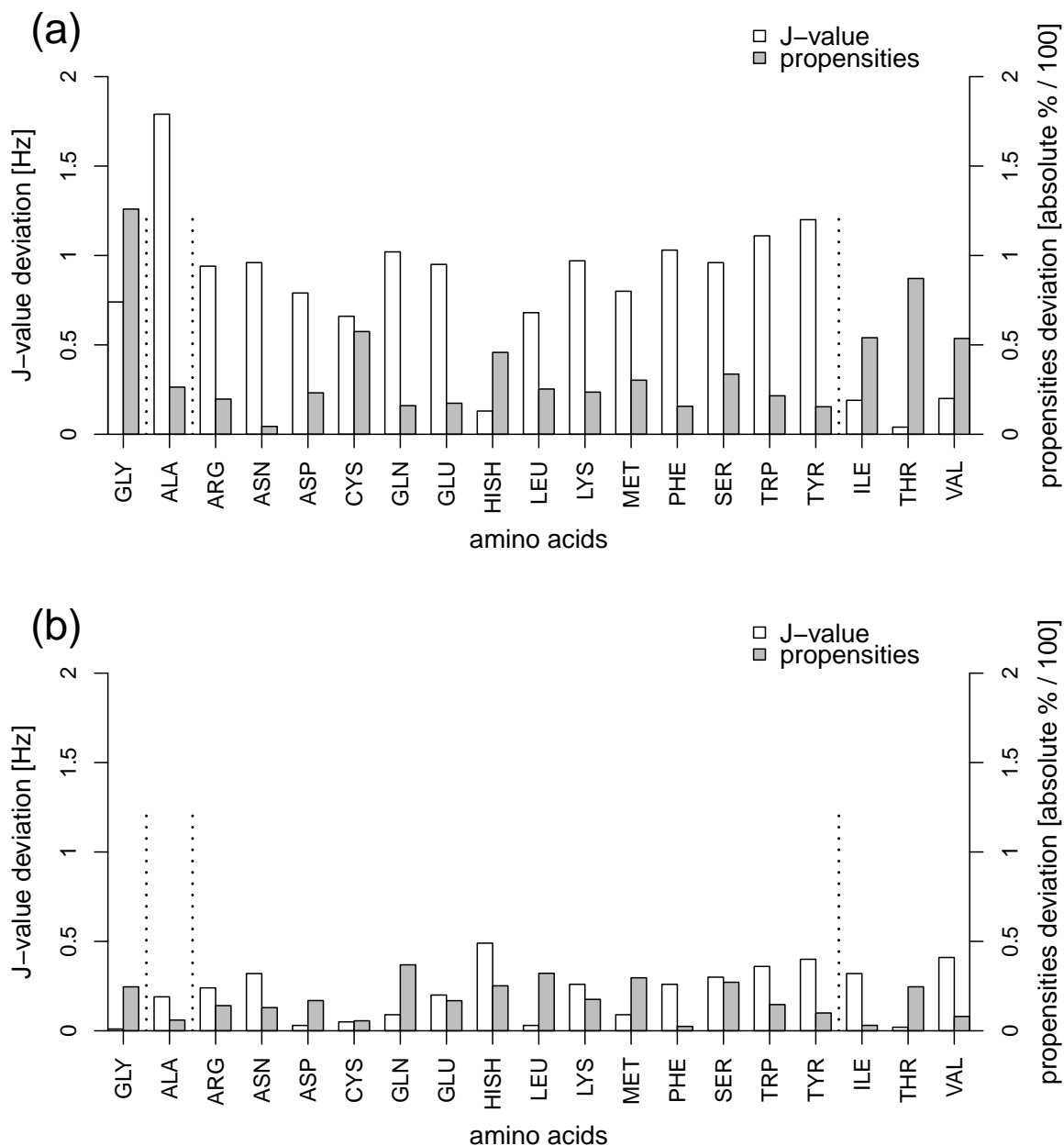


Figure 3.3: Final deviations between simulated and experimental data using 54A7 parameters (a) and our suggested set (b). White bars indicate the deviation of the J-value (left axis) and shaded bars indicate deviations in the secondary structure propensities (right axis). Dotted lines indicate the subgroups used in the optimization.

3.4 Results and discussion

Our analysis shows, that the GROMOS force field parameter set 54A7 fails to reproduce the experimental studies in terms of $^3J(\text{HN}, \text{H}_\alpha)$ and secondary structure element propensities in the context of blocked amino acids (see figure 3.3 (A) and the lines marked with 54A7 in tables 3.4 and 3.3).

Table 3.4: Summary of the averaged performance of the different sets in terms of agreement with experimental data. The absolute values of the individual deviation have been averaged and are reported together with their standard deviation. For the propensities, the renormalized data is reported.

subset	combination	description	$\langle \Delta J\text{-value} \rangle$ [Hz]	$\langle \Delta \alpha \rangle$	propensities [%/100] $\langle \Delta \beta \rangle$	$\langle \Delta P_H \rangle$	Δ^a $\langle \text{abs.} \rangle$ $\langle \text{ren.} \rangle$
glycine	54A7 ^b	see reference [1]	0.74 \pm 0.00	0.646 \pm 0.000	0.121 \pm 0.000	0.144 \pm 0.000	1.651 \pm 0.000 1.976 \pm 0.000
	#81883	suggested	0.04 \pm 0.00	0.450 \pm 0.000	0.058 \pm 0.000	0.188 \pm 0.000	0.736 \pm 0.000 0.270 \pm 0.000
	#81883 ^b	simulated	0.01 \pm 0.00	0.439 \pm 0.000	0.062 \pm 0.000	0.190 \pm 0.000	0.701 \pm 0.000 0.256 \pm 0.000
	#5623	common ^c	1.41 \pm 0.00	0.658 \pm 0.000	0.082 \pm 0.000	0.212 \pm 0.000	2.362 \pm 0.000 3.076 \pm 0.000
alanine	54A7 ^b	see reference [1]	1.79 \pm 0.00	0.001 \pm 0.000	0.098 \pm 0.000	0.156 \pm 0.000	2.045 \pm 0.000 2.048 \pm 0.000
	#12572	suggested	0.24 \pm 0.00	0.004 \pm 0.000	0.031 \pm 0.000	0.008 \pm 0.000	0.283 \pm 0.000 0.283 \pm 0.000
	#12572 ^b	simulated	0.14 \pm 0.00	0.017 \pm 0.000	0.051 \pm 0.000	0.003 \pm 0.000	0.211 \pm 0.000 0.224 \pm 0.000
	#5623	common ^c	1.08 \pm 0.00	0.071 \pm 0.000	0.240 \pm 0.000	0.225 \pm 0.000	1.616 \pm 0.000 1.623 \pm 0.000
common	54A7 ^b	see reference [1]	0.87 \pm 0.26	0.093 \pm 0.045	0.098 \pm 0.072	0.052 \pm 0.037	1.115 \pm 0.212 1.103 \pm 0.198
	#5623	suggested	0.25 \pm 0.15	0.024 \pm 0.019	0.083 \pm 0.062	0.058 \pm 0.038	0.416 \pm 0.170 0.425 \pm 0.179
	#5623 ^b	simulated	0.22 \pm 0.15	0.018 \pm 0.014	0.092 \pm 0.053	0.067 \pm 0.054	0.401 \pm 0.153 0.410 \pm 0.156
	table 3.5	individual	0.03 \pm 0.04	0.019 \pm 0.023	0.087 \pm 0.081	0.065 \pm 0.068	0.201 \pm 0.152 0.131 \pm 0.066
C _{β} -branched	54A7 ^b	see reference [1]	0.14 \pm 0.09	0.193 \pm 0.159	0.334 \pm 0.094	0.102 \pm 0.061	0.772 \pm 0.103 0.795 \pm 0.105
	#86516	suggested	0.20 \pm 0.15	0.011 \pm 0.004	0.066 \pm 0.090	0.036 \pm 0.034	0.316 \pm 0.038 0.309 \pm 0.025
	#86516 ^b	simulated	0.25 \pm 0.20	0.015 \pm 0.009	0.066 \pm 0.091	0.040 \pm 0.006	0.371 \pm 0.115 0.369 \pm 0.113
	#5623	common ^c	0.67 \pm 0.07	0.072 \pm 0.078	0.405 \pm 0.057	0.306 \pm 0.037	1.450 \pm 0.172 1.469 \pm 0.184
	table 3.5	individual	0.02 \pm 0.02	0.006 \pm 0.003	0.079 \pm 0.035	0.056 \pm 0.025	0.164 \pm 0.085 0.165 \pm 0.085

^a The overall Δ is the sum of the absolute values of the deviation in Hz for the J-value and the discrepancies in % for the propensities, see equation 3.4. Column $\langle \text{abs.} \rangle$ refers to the deviation when using the absolute occurrences of the three secondary structure classes while column $\langle \text{ren.} \rangle$ refers to the average deviations when the propensities are first renormalized to 100%.

^b These values have been computed from the respective actual simulations and not been predicted by reweighting.

^c For comparison, the values projected for the common set are reported for the other subgroups as well. It is clear, that #5623 does not perform well for the other groups.

The average deviation in J-values amounts to 0.8 Hz and the propensities are off by a total of 0.34 (34 %). Even if the relatively high uncertainty is considered, when comparisons of experimental and Karplus-derived J-values is undertaken, the results for 54A7 are quite poor. Half of the amino acids are close or more than 1 Hz off the respective target values (see table 3.3). If plotted with a linear regression model, the R^2 for 54A7 is 0.404 and if alanine, as the major outlier, is removed, this improves only to 0.606 (see figure S3.11). The match of secondary structure propensities is better and deviations arise mainly from a shifted ratio between the strongly populated β and P_{II} basins and a few cases, where artificially high α -helical propensities also play a significant role (e.g. threonine or cysteine). This mismatch is hardly surprising, given that 54A7 was not parametrized against such data and uses one set of protein backbone potentials for all amino acids for the sake of simplicity.

As outlined in the *Methods* section, we have subsequently embarked on a reparametrization effort. Because of the large number of potential combinations (table 3.2), a predictive method was used to estimate the average J-values and propensities, rather than trying to simulate all of the dihedral-angle potential-energy combinations (Hamiltonian reweight, see methods). The accuracy of this approach was ensured by testing against data obtained from actual simulations for three different, randomly chosen combinations (see figure S3.8) and for the di-alanine peptide by performing simulations for all of the finally suggested combinations for all amino acids. The predicted and simulated values are reported in table 3.3 and show a quantitative match in both the J-values and the propensities. As elaborated in the methods section, our selection was biased towards combinations with wide distributions in the respective basins. For asparagine and glutamine, however, the predictions may be slightly biased, because in 54A7, both amino acids show no α -population at all (experimental estimates amount to 2 and 8%, respectively). Since no configurations are available to be reweighted, our prediction is off in this respect. This shows the general limitation of our approach: if there is no significant overlap of the ensembles of the reference and target state, the prediction is naturally very poor. However, it is noteworthy, that the actual simulation of glutamine using the suggested combination #5623 leads to a better agreement. The other amino acids in this subgroup are sufficient to drive the conformational ensemble of glutamine towards the α -population. In figure S3.10 the prediction of serine using a coarse 15x15 bin resolution for several combinations is shown as an example. As long as the number of configurations is large enough, it is clearly possible to predict the sampling of the backbone angles with a higher resolution than just three secondary structure basins.

It appeared difficult to identify a single combination of dihedral angle parameters that could uniformly satisfy the experimental data for all amino acids. This suggests that different potential energy terms may be required for different amino acids. In our opinion, this is hardly surprising as the size, shape and polarity of the side-chains will most likely have an effect on the torsional potential of the backbone and thus need to be taken into account. On the other hand, while it is theoretically possible to optimize the amino acids individually, that approach leads to a complex and potentially over-fitted result. Instead, we chose a somewhat intermediate solution: we identified subgroups based on the differences in the experimental data and the possibilities to reproduce these subgroups with common potential energy functions. The subgroups nicely appeared to correspond to the substitution pattern at the C_β side-chain atom. Concretely, the identified commonalities between outliers which lead to the separate optimization of glycine, alanine, the "common" amino acids and a subgroup with a CH-group at C_β -branched amino acids, valine,

isoleucine and threonine. Proline, for which the backbone degrees of freedom are much restricted was excluded from the optimizations.

The rationale for this subdivision of the amino acids is likely to be found in their respective side-chain characteristics: Glycine does not have a real side-chain and this lack of certain steric hindrances allows a broad sampling of angles (see figure 3.4 and reference [30]). Alanine is the only amino acid that has no extension on C_β . The common amino acids have one non-hydrogen extension at C_β and the β -branching in valine, isoleucine and threonine leads to special rotational profiles. In order to illustrate that, we report the values predicted by using combination #5623 (the suggested solution for the common amino acids) in table 3.3 for alanine, glycine and the C_β -branched amino acids. Given the experimental data we aimed for and the vast set of possibilities which has been screened, it might be expected that a solution adequate for all amino acids simultaneously would have been found if it was possible. Instead, we were not able to optimize all canonical amino acids together. Glycine is described best by low potentials, granting it its natural extraordinary freedom in sampling the Ramachandran space. The score obtained for alanine using the "common" parameters (#5623) shows a deviation of 1 Hz in the J-value and a strong bias towards the α -helical basin. The three β -branched amino acids also do not perform well with this set and in all three cases have a summed deviation much worse than even 54A7, arising from both the J-value and the propensities. For threonine, valine and isoleucine alike, the common parameters lead to an extreme over-emphasis of the P_{II} basin at the cost of β -conformations. Accordingly, the optimized torsional energy profiles are quite different for both ϕ and ψ .

The optimal set of parameters for the four subgroups are given in table 3.3 and a graphical comparison of the potential energy contributions is given in figure 3.5. Note, that these figures are somewhat misleading, because the main contribution to the potential energy landscape is determined by the intramolecular non-bonded interactions. This is exemplified by figure 3.6 where the nonbonded energy of the alanine dipeptide is drawn, resulting from a systematic scan of ϕ and ψ , starting from a minimized conformation in vacuum. The torsional dihedral angle profiles of figure 3.5, merely modulate the intrinsic potential energy surface, leading to the appropriate shifts in J-value and secondary structure propensities. It is not surprising that there are many alternative combinations that could lead to similar shifts in the conformational preference. To ensure that the seemingly high barriers in figure 3.5 do not lead to artificially locking of molecules in local minima we have computed the mean residence times in a certain basin for one representative amino acid of all subgroups (table S3.9). Compared to 54A7 the residence times in the helical conformation seems to be reduced, while residence times in β and P_{II} are slightly increased, with the most extreme increment observed for the P_{II} conformation of alanine from 2.1 ps to 13.7 ps. With currently available simulation times, we do not consider this a significant reduction of the dynamic behaviour of the molecule.

In general, it appears that matching of the J-value alone is not sufficient to ensure a representative distribution (for example, the J-value for threonine in 54A7 matches perfectly but in terms of propensities, the agreement with the experimental values is rather poor; reweighting combinations #33268 and #75480 for glycine share the J-value but differ 11% in propensities, data not shown). The concomitant optimization of both the J-value and the secondary structure propensities reduces the number of combinations to be considered: although the J-values have been used for calibration of the Raman spectroscopy experiments, the latter holds additional information.

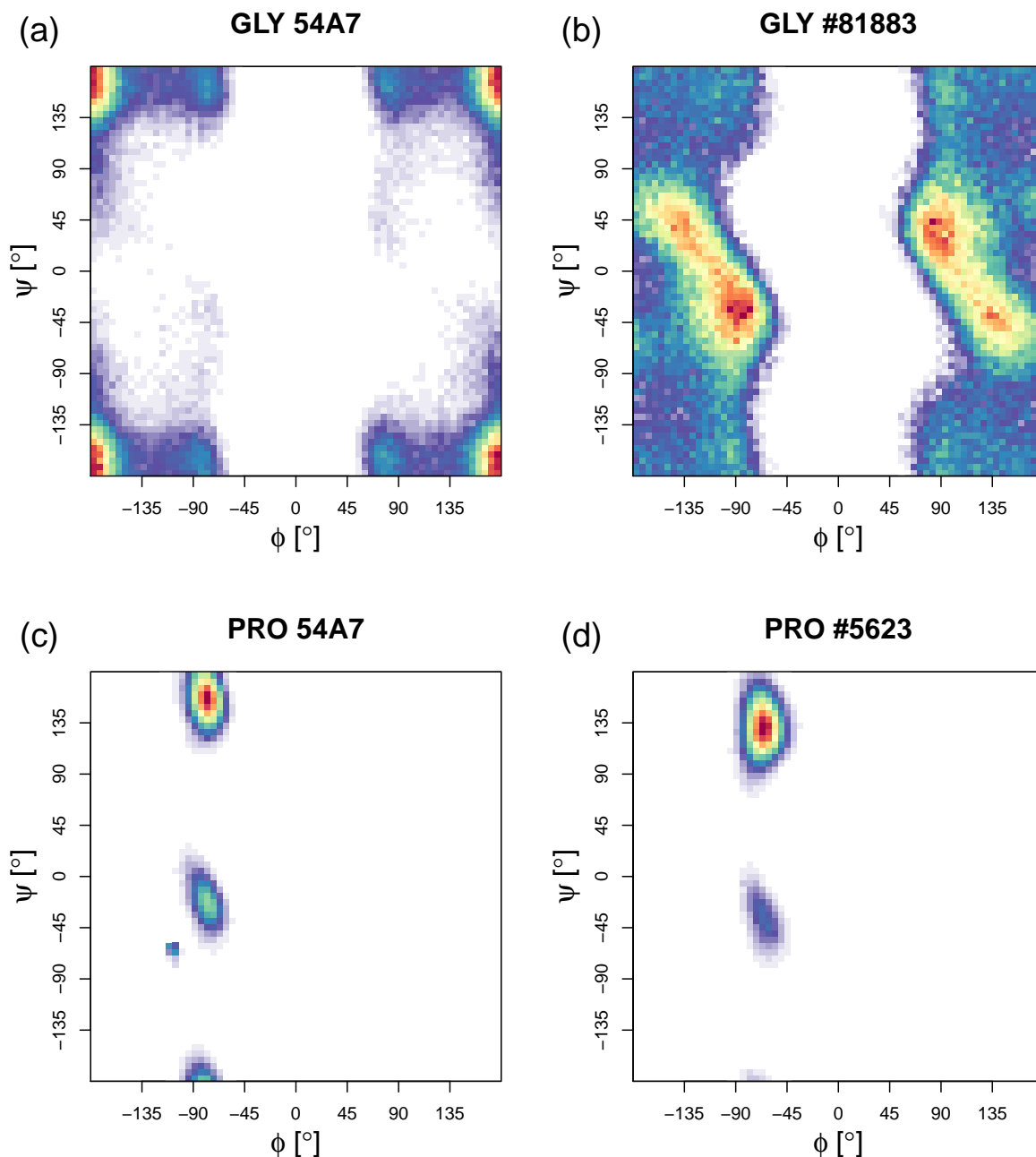


Figure 3.4: Ramachandran plot of glycine and proline with different parameters. As is shown in (a), 54A7 fails to reproduce a distribution that is in agreement with literature (see reference [30]), while the distribution for #81883 in (b) seems to be in better agreement. Moreover, proline in 54A7 shows an unexpected peak at a ϕ and ψ of approximately -107° and -65° , respectively (c). For our combination #5623, which is suggested for the common amino acids, we observe (d) a significant shift towards the P_{II} conformation, leaving only a minor fraction in the helical basin.

Performing the step-wise selection of potential candidates (see methods section), we were able to provide a set of suggested parameters (table 3.4) that performs significantly better than the GROMOS 54A7 parameters in terms of agreement with experimentally determined J-values and propensities (figures 3.3 and S3.11). In addition to these values, we also sought out the best parameter set for every amino acid individually. Table S3.5

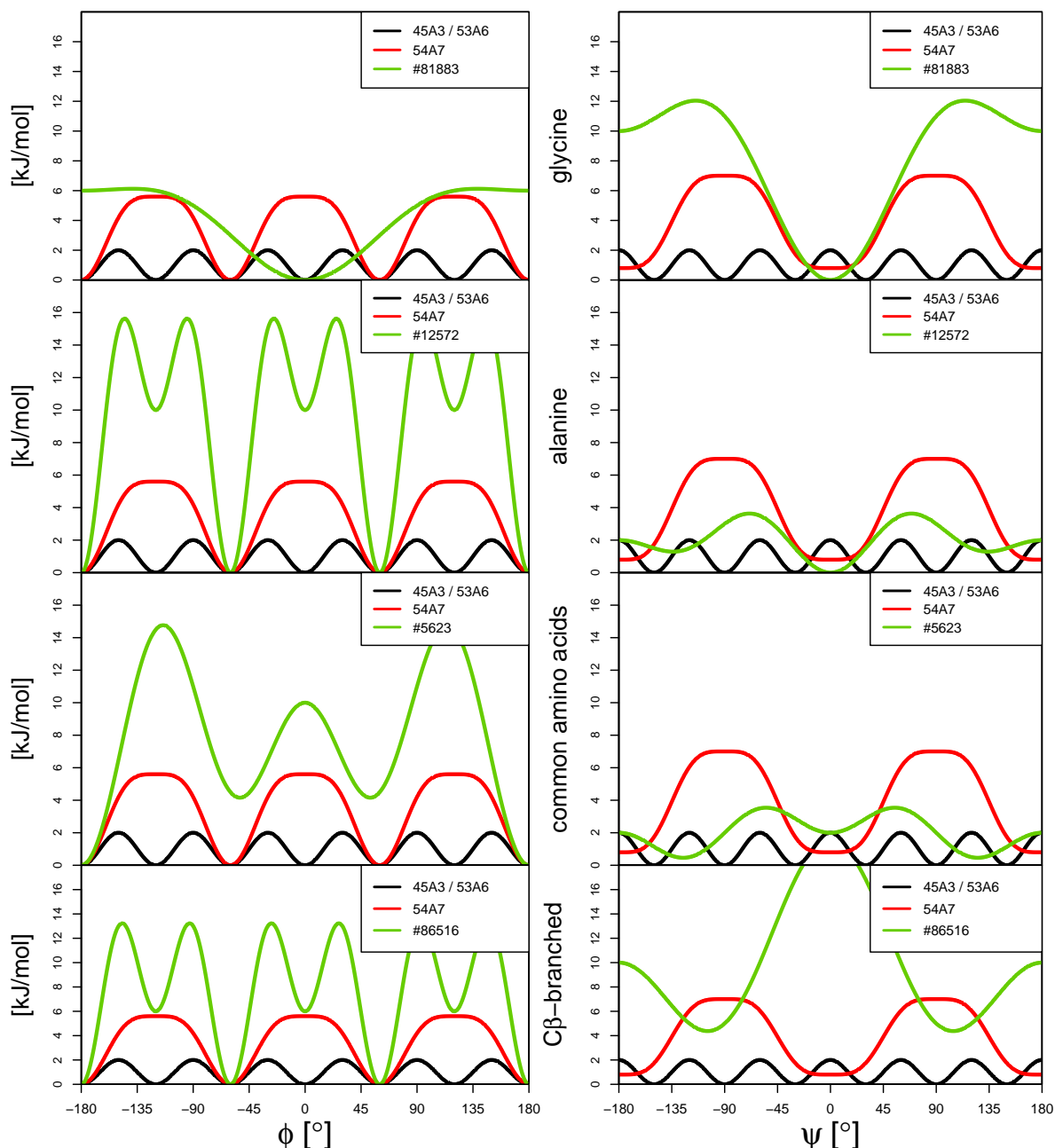


Figure 3.5: Potential-energy terms of table 3.1. The first row represents the suggested parameters for glycine, the second for alanine, the third for the common amino acids' subset and the last for the C_{β} -branched amino acids, respectively. The left column shows the ϕ , the right the ψ angle.

shows the individually optimized combinations. In general, these perform significantly better than those, optimized for the subgroups (see table 3.1), suggesting that our set of parameters contains enough variety to account for the special features of all 20 canonical amino acids.

Of the four subgroups, the β -branched one is the most likely to be further improved. For a group of only three members, the deviation is still quite high for our suggested set of parameters.

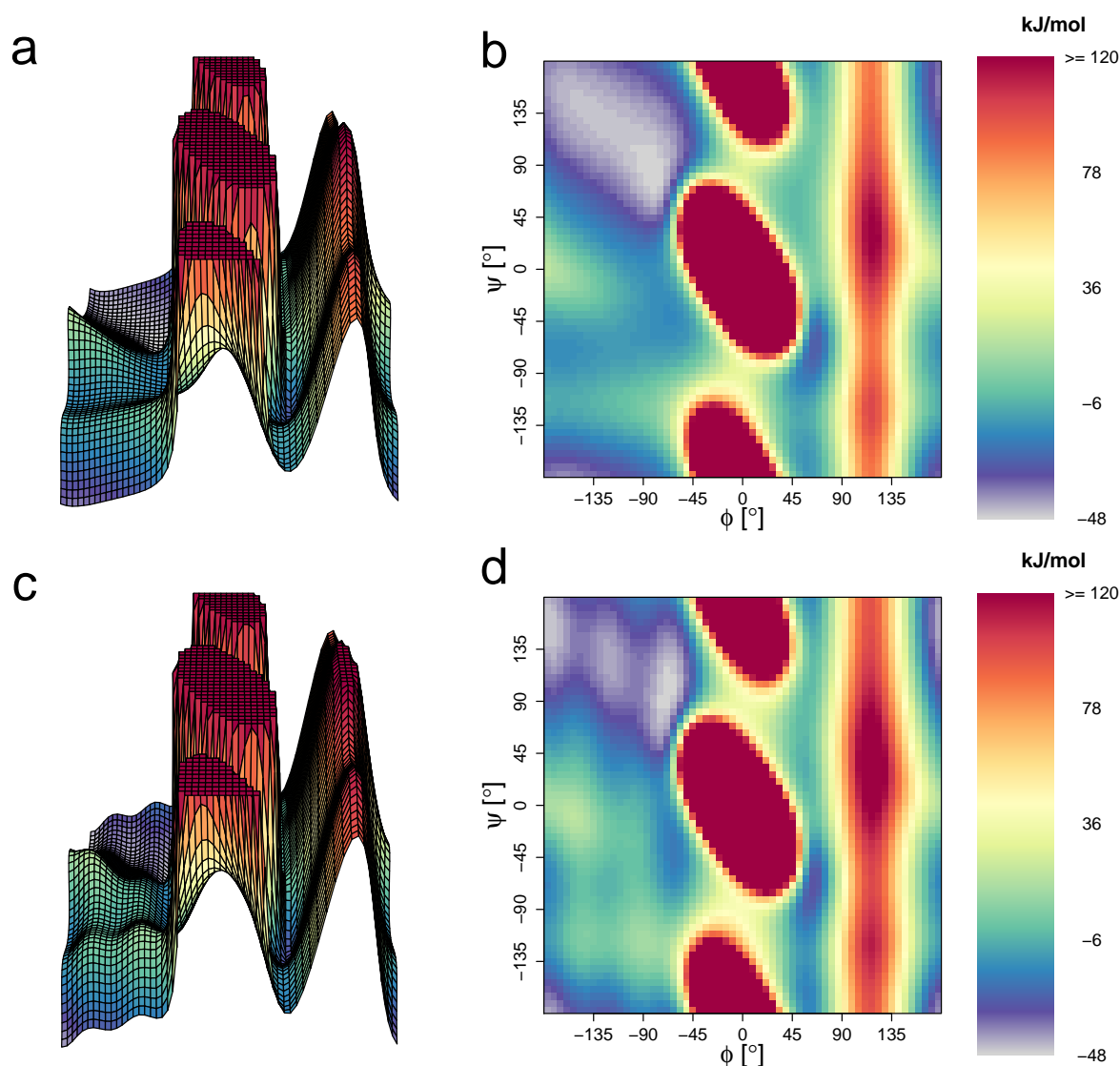


Figure 3.6: Backbone energy surface for Ac-A-NHMe (using parameter combination #12572) when rotating the ϕ - and ψ angles. In (a) and (b), the energy surface arising from the non-bonded interactions only is shown in 3D and 2D representations, respectively. It is clear, that the addition of the dihedral angle potentials contributes only by a limited, but still crucial, amount as shown in (c) and (d). This modulation of the energy landscape particularly leads to a P_{II} separation and forms a small α -helical basin, which is populated by 12.7% (estimated by simulation).

It appears from the results shown in figure 3.3, that all three amino acids contribute to the same extent and their best individual hits share the ϕ -potential, but it is threonine that is significantly different in ψ (table S3.5) and prohibits a further subgroup-wise optimization. Indeed, for many of the best hits of this subgroup, we identified threonine to be contributing most to the deviations.

Among the 20 canonical amino acids in this study, there are two that require special consideration: glycine and proline. Since the propensity values for glycine are most likely only comparable when the crucial effect of the normalization in this case is taken into account, we propose a parameter set for this special amino acid, that samples wide areas of the Ramachandran plot, as also observed in the PDB structures [30]. Figure 3.4 shows the obtained distribution for 54A7 and the selected parameter set (#81883). For

proline, no experimental values were available, so we adopted the "common" parameters (set #5623) for this amino acid leading to a more pronounced sampling of the P_{II} basin and the removal of a surprising (artificial) peak at $\phi \approx -107^\circ$ and $\psi \approx -76^\circ$ (see figure 3.4).

Further tests could include other small compound series (e.g. it has been reported in an earlier study [31] that the $^3J(\text{HN}, \text{H}_\alpha)$ increases for alanine inserted in the XAO peptide; see studies mentioned before as well). As a preliminary test of the parameters in a protein environment, we have performed four independent 50-ns simulations of hen egg-white lysozyme (HEWL; initial structure taken from PDB entry 4B0D [32]) using both the GROMOS 54A8 parameter set and a parameter set based on 54A8 with the suggested backbone dihedral parameters. The overall structure seems to be equally well maintained in terms of both root-mean-square deviation (figure 3.7) and the persistence of secondary structure elements.

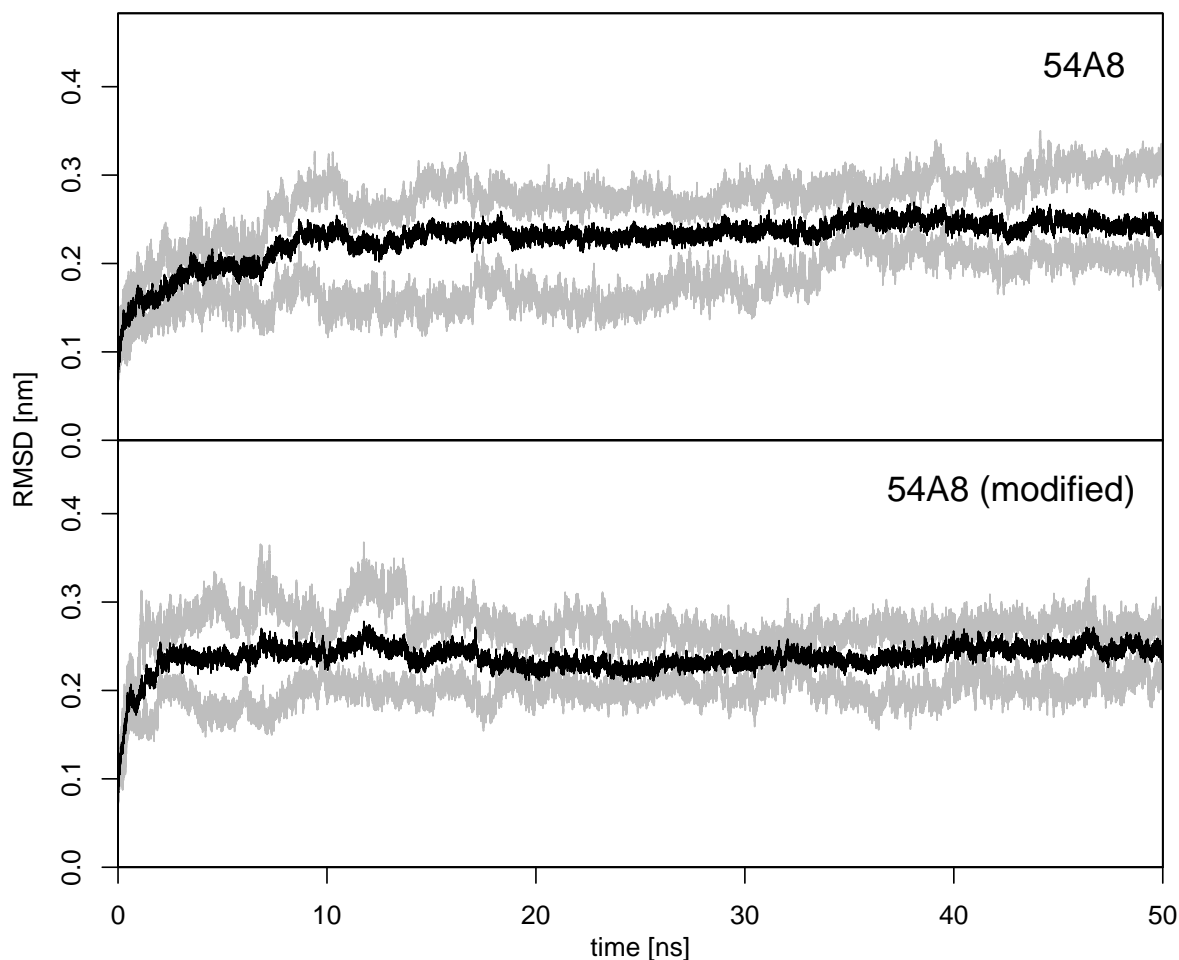


Figure 3.7: Average root-mean-square deviation (RMSD) with respect to the original structure for four independent simulations of HEWL using parameter set 54A8 (upper panel) and 54A8 with our suggested dihedral parameters (lower panel), respectively. The RMSD has been calculated based on the backbone atoms C, N and O exclusively. Both parameter sets show a stable structure over 50 ns. The plot has been generated using the R package MDplot, the black curve denotes the mean and the grey ones the respective minimum and maximum values at every time point.

While the simulations with 54A8 show helical conformations of $36.6 \pm 1.1\%$ (averaged over sequence, time and individual simulations), this amounts to $37.6 \pm 2.8\%$ for the simulations with the modified backbone dihedral angle parameters. Note, that this is merely a preliminary confirmation that the updated set of parameters does not disrupt protein structure. Future analyses will involve detailed comparisons to NMR observables (NOE distance restraints, J-values) and a more extensive set of proteins. It is not unlikely that further refinements will be necessary at the protein level. It can be expected, that this kind of analysis will have to be performed for every major iteration of the force field's (non-bonded) parameter set because of the inter-dependence of parameters mentioned before.

3.5 Conclusion

In this publication, we focused on finding optimized parameters for the backbone dihedral angles of amino acids. By screening a vast library of potential combinations using Hamiltonian reweighting, a set of parameters can be provided, that optimizes the experimental target data. We have proven Hamiltonian reweighting being a useful tool in the parametrization process of molecular dynamics force fields and its general accuracy for small to medium changes in the Hamiltonian. Our optimization procedure might serve as an useful basis for further optimizations of other force fields, especially in the range from very small to medium size peptides and intrinsically unstructured regions in proteins. Sets of parameters have been suggested for four subsets of amino acids that differ in the substitution at C_β , but various other potential parameter sets are available for further evaluation. Future studies will concentrate on the reproduction of experimental data for other small systems and the performance of the selected parameter sets in proteins, and the agreement against e.g. NMR data for such simulations.

3.6 Appendix / Supplementary material

Table 3.5: Combinations for the individually optimized amino acids.

	backbone angle energy potentials						description
	ϕ			ψ			
	K [$\frac{\text{kJ}}{\text{mol}}$]	shift [$^\circ$]	mult.	K [$\frac{\text{kJ}}{\text{mol}}$]	shift [$^\circ$]	mult.	
#67656	5.0	0.0	3	3.0	0.0	2	arginine
	5.0	180.0	6	5.0	0.0	2	
#96795	5.0	0.0	3	5.0	180.0	2	asparagine
	5.0	180.0	2	5.0	180.0	2	
#31541	3.0	180.0	2	1.0	180.0	2	aspartate
	5.0	0.0	3	3.0	180.0	3	
#50989	3.0	180.0	2	1.0	180.0	3	cysteine
	5.0	0.0	3	5.0	180.0	3	
#83856	5.0	0.0	3	3.0	180.0	2	glutamine
	5.0	180.0	6	5.0	180.0	6	
#87099	5.0	0.0	3	5.0	0.0	2	glutamate
	5.0	180.0	6	5.0	0.0	2	
#89861	3.0	180.0	2	5.0	0.0	2	histidine
	5.0	0.0	3	5.0	0.0	6	
#9315	5.0	0.0	3	1.0	0.0	6	leucine
	5.0	180.0	2	1.0	180.0	3	
#80612	5.0	0.0	3	3.0	180.0	3	lysine
	5.0	180.0	6	5.0	0.0	2	
#28751	5.0	0.0	3	1.0	180.0	6	methionine
	5.0	180.0	2	3.0	0.0	1	
#25538	5.0	0.0	3	1.0	180.0	3	phenylalanine
	5.0	180.0	6	3.0	0.0	2	
#67652	5.0	0.0	3	3.0	0.0	1	serine
	5.0	180.0	6	5.0	0.0	2	
#93571	5.0	0.0	3	5.0	0.0	2	tryptophane
	5.0	180.0	6	5.0	180.0	3	
#6094	5.0	0.0	3	1.0	0.0	1	tyrosine
	5.0	180.0	6	1.0	180.0	6	
#28191	3.0	0.0	3	1.0	180.0	6	isoleucine
	5.0	180.0	6	3.0	0.0	1	
#92991	3.0	0.0	3	5.0	0.0	2	threonine
	5.0	180.0	6	5.0	180.0	3	
#24957	3.0	0.0	3	1.0	180.0	3	valine
	5.0	180.0	6	3.0	0.0	1	

Table 3.6: Parameters used for blocking for the Ac-X-NHMe simulations.

Acetyl-			
	Integer Atom Code (IAC)	Masscode	Charge
CM	16	5	0.00
C	12	12	0.45
O	1	16	-0.45
N-Methyl-			
	Integer Atom Code (IAC)	Masscode	Charge
C	12	12	0.45
O	1	16	-0.45
N	7	14	-0.31
H	21	1	0.31
CT	15	5	0.00

Table 3.7: List of the target values extracted from the studies of Avbelj et al. [2] and Grdadolnik et al. [3]. The propensities have been retrieved from Raman spectroscopy data calibrated by the J-values of the former study.

	target values			
	J-value	α	β	P_{II}
ALA	6.06	0.11	0.60	0.29
GLY	5.85	0.66 ^a	0.12 ^a	0.22 ^a
ARG	6.85	0.07	0.54	0.39
ASN	7.45	0.02	0.58	0.40
ASP	6.93	0.05	0.46	0.49
CYS	7.31	0.03	0.54	0.43
GLU	6.63	0.05	0.36	0.59
GLN	7.14	0.08	0.48	0.44
HIS	7.89 ^b	0.04 ^b	0.58 ^b	0.38 ^b
ILE	7.33	0.02	0.52	0.46
LEU	6.88	0.10	0.35	0.55
LYS	6.83	0.04	0.41	0.55
MET	7.02	0.03	0.47	0.50
PHE	7.18	0.06	0.49	0.45
SER	7.02	0.04	0.47	0.49
THR	7.35	0.03	0.58	0.39
TRP	6.91	0.02	0.44	0.54
TYR	7.13	0.07	0.47	0.46
VAL	7.30	0.02	0.51	0.47

^a Although the special distribution of the dihedral angles in the case of glycine may lead to a great over-emphasis of the defined three basins, we apply the same renormalization procedure as in the other cases (see text for justification).

^b The values of histidine have not been taken into account in the ranking protocol.

Table 3.8: Listing of the dihedral angle areas used for classification. This table holds the basins [17], which were used for the classification with DISICL [17, 18] for all compounds investigated. The original values were reported by Hollingsworth et al. [19]. A structure with a pair of ϕ/ψ backbone dihedrals, that falls within one of these regions is assigned to belong to the secondary structure element given in the first column. All remaining structures are considered to be unclassified. The experimental comparison was based on helical_R , β and P_{II} .

	definitions			
type	ϕ_1 [°]	ϕ_2 [°]	ψ_1 [°]	ψ_2 [°]
helical_R	-95	-40	-70	-32
	-107	-40	-32	-12
	-165	-95	-70	-32
	-107	-40	-12	8
	-150	-67	8	40
	-150	-107	-32	8
β	-135	-100	95	150
	-175	-135	95	136
	-180	-135	136	180
	-135	-105	150	180
	-180	-135	-180.1	-160
helical_L	38	140	-25	75
P_{II}	-100	-30	95	180
	-100	-60	-180.1	-150

Table 3.9: Comparison of dynamical parameters of different combinations. On the left of each column, the number of visits per basin is reported, while on the right the average residence time per basin is given in picoseconds. Some combinations in our suggested set increase the torsional energy barriers (especially in the ϕ -angle, see figure 3.5), which in turn leads to higher residence times. Still, a sufficient number of transitions between the basins is observed. All values are reported for trajectories of 100 ns length.

system	parameters	# α	<time> [ps]	# β	<time> [ps]	#P _{II}	<time> [ps]
GLY	54A7	737	2.0	15158	1.6	3385	1.8
GLY	#81883	7116	3.1	3650	1.6	1845	1.6
ALA	54A7	681	16.3	16233	2.8	15183	2.1
ALA	#12572	914	11.3	5614	4.5	4448	13.7
ALA	#12572 (334 K)	1369	9.1	7466	3.5	6089	9.3
ALA	#12572 (354 K)	1839	8.0	8502	3.2	6948	7.6
CYS	54A7	732	24.5	14684	3.1	12952	2.1
CYS	#5623	560	6.8	9962	4.9	7673	5.6
VAL	54A7	368	33.6	16183	2.1	17247	2.5
VAL	#86516	113	2.9	9712	5.4	7238	5.9

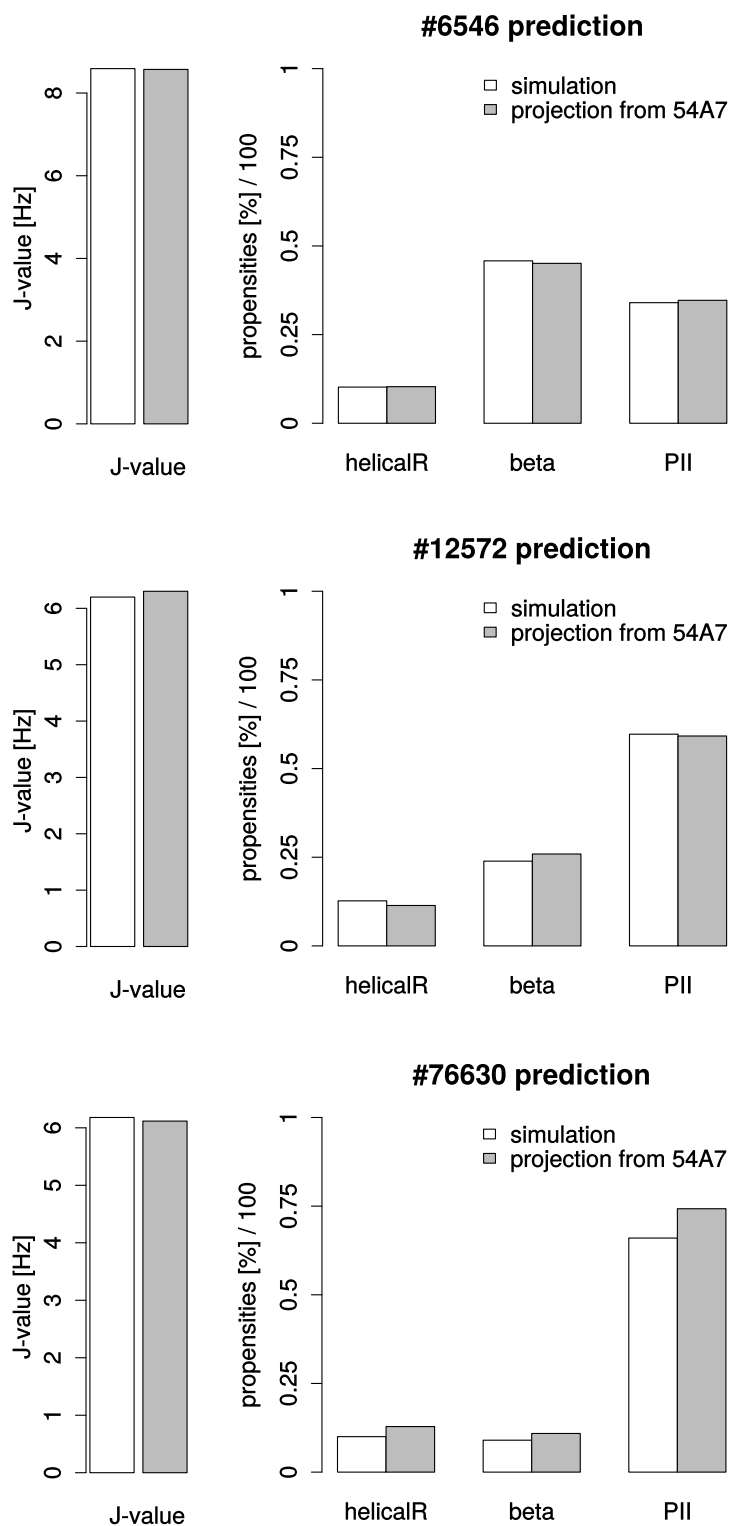


Figure 3.8: Shows the accuracy of the projections based on the 54A7 trajectories for alanine for three different sets of backbone dihedral potentials. Note, that the projection is quite accurate even when different trends (e.g. propensities of #76630 versus those of #6546) are observed.

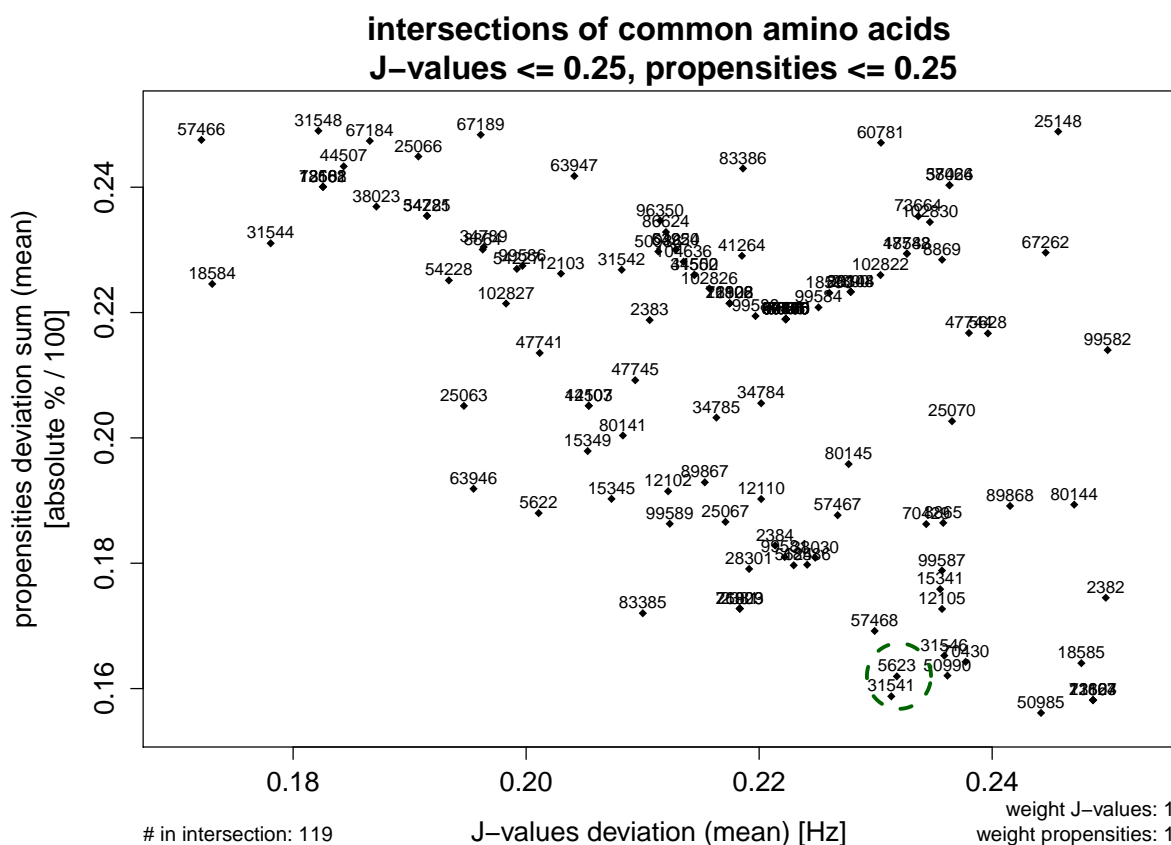


Figure 3.9: Example of scatter plot showing the deviation in J-value reproduction (x-axis) versus the summed up deviation in propensities (y-axis) for a selection of combinations from our screening set. In principle all combinations in this scatter plot could lead to reasonable agreement with experimental data. The combination #5623, which has been proposed for the description of the "common" amino acids (see main manuscript), is marked with a green dashed circle.

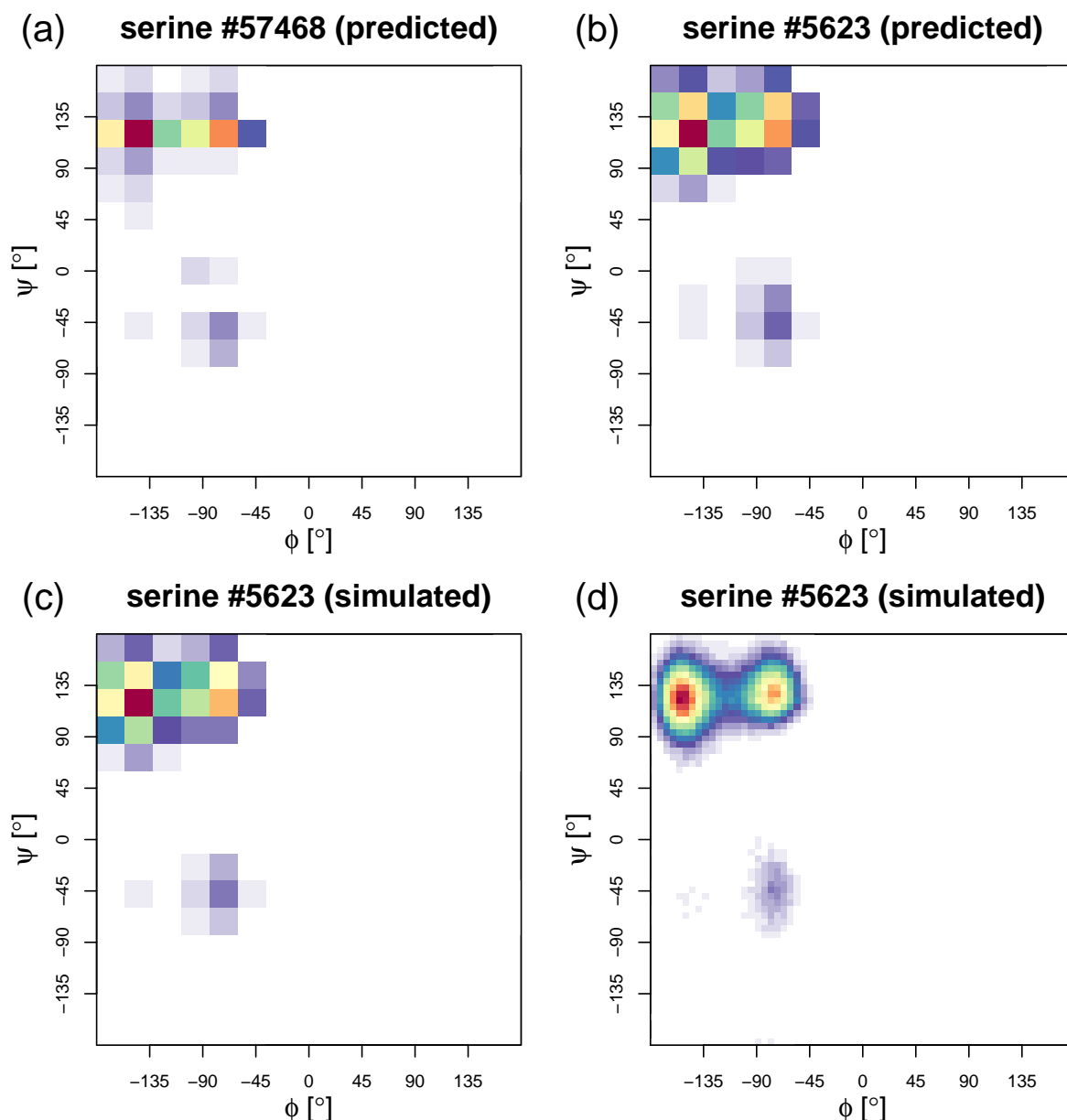


Figure 3.10: Prediction accuracy for the 15x15 bin Ramachandran plots on the example of serine. For combination #57468, which produced very good results in terms of matching the experimental results, a very narrow distribution within the basins is observed (a). Combination #5623 performs only slightly worse in the agreement with the experimental data, but leads to a wider sampling (b). The reliability of the prediction is shown by comparison to actual simulation data in panel (c). In (d), the simulated result with a higher resolution is shown.

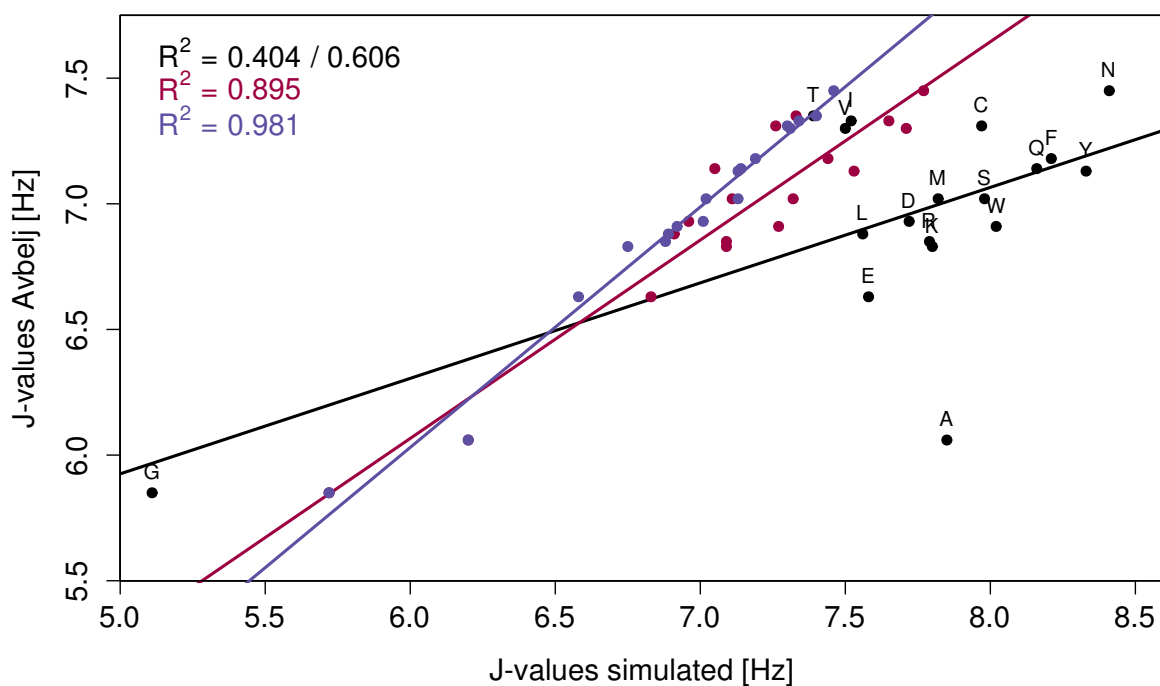


Figure 3.11: Correlation plot of experimental J-values with the calculated ones, based on 54A7 (black), our suggested set (red) and the individually optimized (blue) parameters. Since alanine is a strong outlier for 54A7, we also report the correlation coefficient for the remaining residues only which leads to a slight improvement (see second value). The slopes are 0.380, 0.397, 0.789 and 0.957, and the intercepts are 4.025, 3.952, 1.331 and 0.289 for 54A7, 54A7 without alanine, our suggested set and the individually optimized set, respectively. The markers for Ala and Gly are positioned identically in the proposed and the individually optimized sets.

References

- [1] Nathan Schmid, Andreas P. Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E. Mark, and Wilfred F. van Gunsteren. “Definition and testing of the GROMOS force-field versions 54A7 and 54B7”. In: *European Biophysics Journal* 40.7 (July 1, 2011), pp. 843–856. DOI: [10.1007/s00249-011-0700-9](https://doi.org/10.1007/s00249-011-0700-9).
- [2] Franc Avbelj, Simona Golig Grdadolnik, Joze Grdadolnik, and Robert L. Baldwin. “Intrinsic backbone preferences are fully present in blocked amino acids”. In: *Proceedings of the National Academy of Sciences of the United States of America* 103.5 (Jan. 2006), pp. 1272–1277. DOI: [10.1073/pnas.0510420103](https://doi.org/10.1073/pnas.0510420103).
- [3] J. Grdadolnik, V. Mohacek-Grosov, R. L. Baldwin, and F. Avbelj. “Populations of the three major backbone conformations in 19 amino acid dipeptides”. en. In: *Proceedings of the National Academy of Sciences* 108.5 (Feb. 2011), pp. 1794–1798. DOI: [10.1073/pnas.1017317108](https://doi.org/10.1073/pnas.1017317108).
- [4] Robert B. Best, Xiao Zhu, Ji Hyun Shim, Pedro E. M. Lopes, Jeetain Mittal, Michael Feig, and Alexander D. MacKerell. “Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone ϕ , ψ and side-chain χ_1 and χ_2 dihedral angles”. In: *Journal of chemical theory and computation* 8.9 (Sept. 2012), pp. 3257–3273. DOI: [10.1021/ct300400x](https://doi.org/10.1021/ct300400x).
- [5] Chen-Yang Zhou, Fan Jiang, and Yun-Dong Wu. “Residue-Specific Force Field Based on Protein Coil Library. RSFF2: Modification of AMBER ff99SB”. In: *The Journal of Physical Chemistry B* 119.3 (2014), pp. 1035–1047. DOI: [10.1021/jp5064676](https://doi.org/10.1021/jp5064676).
- [6] Siobhan Toal, Derya Meral, Daniel Verbaro, Brigita Urbanc, and Reinhard Schweitzer-Stenner. “pH-Independence of Trialanine and the Effects of Termini Blocking in Short Peptides: A Combined Vibrational, NMR, UVCD, and Molecular Dynamics Study”. In: *The Journal of Physical Chemistry B* 117.14 (Apr. 2013), pp. 3689–3706. DOI: [10.1021/jp310466b](https://doi.org/10.1021/jp310466b).
- [7] Nathan Schmid, Clara D. Christ, Markus Christen, Andreas P. Eichenberger, and Wilfred F. van Gunsteren. “Architecture, implementation and parallelisation of the GROMOS software for biomolecular simulation”. In: *Computer Physics Communications* 183.4 (Apr. 2012), pp. 890–903. DOI: [10.1016/j.cpc.2011.12.014](https://doi.org/10.1016/j.cpc.2011.12.014).
- [8] Markus Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: [10.1002/jcc.20303](https://doi.org/10.1002/jcc.20303).
- [9] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81.8 (Oct. 15, 1984), pp. 3684–3690. DOI: [10.1063/1.448118](https://doi.org/10.1063/1.448118).
- [10] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J. C. Berendsen. “Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes”. In: 23.3 (Mar. 1977), pp. 327–341. DOI: [10.1016/0021-9991\(77\)90098-5](https://doi.org/10.1016/0021-9991(77)90098-5).
- [11] Ilario G. Tironi, René Sperb, Paul E. Smith, and Wilfred F. van Gunsteren. “A generalized reaction field method for molecular dynamics simulations”. In: *The Journal of Chemical Physics* 102.13 (Apr. 1, 1995), pp. 5451–5459. DOI: [10.1063/1.469273](https://doi.org/10.1063/1.469273).

- [12] Tim N. Heinz, Wilfred F. van Gunsteren, and Philippe H. Hünenberger. “Comparison of four methods to compute the dielectric permittivity of liquids from molecular dynamics simulations”. In: *The Journal of Chemical Physics* 115.3 (July 15, 2001), pp. 1125–1136. DOI: [10.1063/1.1379764](https://doi.org/10.1063/1.1379764).
- [13] Lukas D. Schuler, Xavier Daura, and Wilfred F. van Gunsteren. “An improved GROMOS96 force field for aliphatic hydrocarbons in the condensed phase”. In: *Journal of Computational Chemistry* 22.11 (Aug. 2001), pp. 1205–1218. DOI: [10.1002/jcc.1078](https://doi.org/10.1002/jcc.1078).
- [14] Chris Oostenbrink, Alessandra Villa, Alan E. Mark, and Wilfred F. van Gunsteren. “A biomolecular force field based on the free enthalpy of hydration and solvation: the GROMOS force-field parameter sets 53A5 and 53A6”. In: *Journal of Computational Chemistry* 25.13 (Oct. 2004), pp. 1656–1676. DOI: [10.1002/jcc.20090](https://doi.org/10.1002/jcc.20090).
- [15] W. Kabsch and C. Sander. “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features”. In: *Biopolymers* 22.12 (Dec. 1983), pp. 2577–2637. DOI: [10.1002/bip.360221211](https://doi.org/10.1002/bip.360221211).
- [16] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. “Stereochemistry of polypeptide chain configurations”. In: *Journal of Molecular Biology* 7 (July 1963), pp. 95–99.
- [17] Gabor Nagy and Chris Oostenbrink. “Dihedral-based segment identification and classification of biopolymers I: proteins”. eng. In: *Journal of Chemical Information and Modeling* 54.1 (Jan. 2014), pp. 266–277. DOI: [10.1021/ci400541d](https://doi.org/10.1021/ci400541d).
- [18] Gabor Nagy and Chris Oostenbrink. “Dihedral-based segment identification and classification of biopolymers II: polynucleotides”. eng. In: *Journal of Chemical Information and Modeling* 54.1 (Jan. 2014), pp. 278–288. DOI: [10.1021/ci400542n](https://doi.org/10.1021/ci400542n).
- [19] Scott A. Hollingsworth, Matthew C. Lewis, Donald S. Berkholz, Weng-Keen Wong, and P. Andrew Karplus. “ (ϕ, ψ) -motifs: a purely conformation-based, fine-grained enumeration of protein parts at the two-residue level”. In: *Journal of Molecular Biology* 416.1 (Feb. 2012), pp. 78–93. DOI: [10.1016/j.jmb.2011.12.022](https://doi.org/10.1016/j.jmb.2011.12.022).
- [20] Andrew Hagarman, Thomas J. Measey, Daniel Mathieu, Harald Schwalbe, and Reinhard Schweitzer-Stenner. “Intrinsic Propensities of Amino Acid Residues in GxG Peptides Inferred from Amide I Band Profiles and NMR Scalar Coupling Constants”. In: *Journal of the American Chemical Society* 132.2 (2010), pp. 540–551. DOI: [10.1021/ja9058052](https://doi.org/10.1021/ja9058052).
- [21] Hoang T. Tran, Xiaoling Wang, and Rohit V. Pappu. “Reconciling Observations of Sequence-Specific Conformational Propensities with the Generic Polymeric Behavior of Denatured Proteins”. In: *Biochemistry* 44.34 (Aug. 2005), pp. 11369–11380. DOI: [10.1021/bi0501961](https://doi.org/10.1021/bi0501961).
- [22] Silvia Pizzanelli, Claudia Forte, Susanna Monti, Giorgia Zandomenighi, Andrew Hagarman, Thomas J. Measey, and Reinhard Schweitzer-Stenner. “Conformations of Phenylalanine in the Tripeptides AFA and GFG Probed by Combining MD Simulations with NMR, FTIR, Polarized Raman, and VCD Spectroscopy”. In: *The Journal of Physical Chemistry B* 114.11 (2010), pp. 3965–3978. DOI: [10.1021/jp907502n](https://doi.org/10.1021/jp907502n).

- [23] David A. C. Beck, Darwin O. V. Alonso, Daigo Inoyama, and Valerie Daggett. “The intrinsic conformational propensities of the 20 naturally occurring amino acids and reflection of these propensities in proteins”. In: *Proceedings of the National Academy of Sciences* 105.34 (Aug. 2008), pp. 12259–12264. DOI: [10.1073/pnas.0706527105](https://doi.org/10.1073/pnas.0706527105).
- [24] A. Pardi, M. Billeter, and K. Wüthrich. “Calibration of the angular dependence of the amide proton-C alpha proton coupling constants, 3JHN alpha, in a globular protein. Use of 3JHN alpha for identification of helical secondary structure”. In: *Journal of Molecular Biology* 180.3 (Dec. 1984), pp. 741–751.
- [25] Urs Stocker and Wilfred F. van Gunsteren. “Molecular dynamics simulation of hen egg white lysozyme: A test of the GROMOS96 force field against nuclear magnetic resonance data”. In: *Proteins: Structure, Function, and Bioinformatics* 40.1 (July 2000), pp. 145–153.
- [26] T. A. Soares, X. Daura, C. Oostenbrink, L. J. Smith, and W. F. van Gunsteren. “Validation of the GROMOS force-field parameter set 45Alpha3 against nuclear magnetic resonance data of hen egg lysozyme”. In: *Journal of biomolecular NMR* 30.4 (Dec. 2004), pp. 407–422.
- [27] Robert B. Best, Nicolae-Viorel Buchete, and Gerhard Hummer. “Are Current Molecular Dynamics Force Fields too Helical?” In: *Biophysical Journal* 95.1 (July 2008), pp. L07–L09. DOI: [10.1529/biophysj.108.132696](https://doi.org/10.1529/biophysj.108.132696).
- [28] G. M. Torrie and J. P. Valleau. “Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling”. In: *Journal of Computational Physics* 23.2 (Feb. 1977), pp. 187–199. DOI: [10.1016/0021-9991\(77\)90121-8](https://doi.org/10.1016/0021-9991(77)90121-8).
- [29] Zhixiong Lin, Chris Oostenbrink, and Wilfred F. van Gunsteren. “On the use of one-step perturbation to investigate the dependence of NOE-derived atom-atom distance bound violations of peptides upon a variation of force-field parameters”. In: *European Biophysics Journal* 43.2-3 (Feb. 2014), pp. 113–119. DOI: [10.1007/s00249-014-0943-3](https://doi.org/10.1007/s00249-014-0943-3).
- [30] Simon C. Lovell, Ian W. Davis, W. Bryan Arendall, Paul I. W. de Bakker, J. Michael Word, Michael G. Prisant, Jane S. Richardson, and David C. Richardson. “Structure validation by C geometry: ϕ, ψ and $C\beta$ deviation”. In: *Proteins: Structure, Function, and Bioinformatics* 50.3 (Feb. 2003), pp. 437–450. DOI: [10.1002/prot.10286](https://doi.org/10.1002/prot.10286).
- [31] Zhengshuang Shi, C. Anders Olson, George D. Rose, Robert L. Baldwin, and Neville R. Kallenbach. “Polyproline II structure in a sequence of seven alanine residues”. In: *Proceedings of the National Academy of Sciences* 99.14 (July 2002), pp. 9190–9195. DOI: [10.1073/pnas.112193999](https://doi.org/10.1073/pnas.112193999).
- [32] Florent Cipriani, Martin Röwer, Christophe Landret, Ulrich Zander, Franck Felisaz, and Jose Antonio Marquez. “CrystalDirect: a new method for automated crystal harvesting based on laser-induced photoablation of thin films”. In: *Acta Crystallogr. Sect. D* 68.10 (Oct. 2012), pp. 1393–1399. DOI: [10.1107/S0907444912031459](https://doi.org/10.1107/S0907444912031459).

Post-translational modifications: effects on the backbone

4.1 Abstract

It is common knowledge that the structure and dynamics of a protein is almost exclusively determined by the sequence of its amino acids. Considerable effort has been put into the search for correlations between the physico-chemical properties of amino acids and features such as secondary structure and stability. However, these studies focused on the 20 canonical amino acids, neglecting the vast enrichment in the chemical diversity of the side-chains introduced by post-translational modifications (PTMs). In this study, we investigate the intrinsic preferences for secondary structure elements of 45 of these post-translational modifications, which commonly occur. We observe a broad range of preferences which depend on the chemical nature of the side-chains. The outliers, that were observed in the results for the 54A7 parameter set, however, disappear when projecting to our recently established backbone dihedral angle parameters described in chapter 3. We have classified the investigated PTMs in the same way as the canonical amino acids, applying one of four groups of parameters depending on the C_β -structure of the side-chain. To our knowledge, this is the first study including a representative set of modified amino acids, which provides a qualitative prediction of the effects PTMs may have on the amino acids' intrinsic structural preferences, helping to rationalize local structural changes induced by them.

4.2 Introduction

For decades, scientists have tried to assign secondary structure propensities to amino acids in order to predict in a bottom-up approach the folding behaviour of peptide and protein sequences [1–10]. Many scales were developed by host-guest experiments, in which one or multiple positions in a (short) peptide sequence are mutated into all 20 canonical amino acids or an amino acid subset [1–3, 7, 11–13]. The analysis is typically done using circular dichroism (CD) spectroscopy [13–15] for peptides of about 7–25 residues or nuclear magnetic resonance (NMR) measurements to access even smaller model systems, typically of a size of five or less amino acids [12, 16–21]. The most recent approaches use vibrational spectroscopy (VS) [5, 22] which is supposed to yield additional insight because of the smaller time scales of measurements compared to NMR [5]. However, all these methods do not measure the propensities directly, which is the main reason why

combinations of methods are often used [12, 21, 23, 24]. On the other hand, molecular dynamics simulations, which directly provide the backbone angle distributions, were shown to often over-predict certain Ramachandran regions (especially the helical type) for small systems [25]. Commonly used model systems include (blocked and unblocked) single amino acids such as Acetyl-X-NHMe (Ac-X-NHMe) [6, 21, 22, 26–28], the tripeptides Ac-AXA-NHMe [23, 24, 29–31], Ac-GXG-NHMe [12, 24], Ac-VXV-NHMe [23, 31] and Ac-PXP-NHMe [4], pentapeptides Ac-GGXGG-NHMe [17, 32, 33] and Ac-PPXPP-NHMe [8] as well as peptides purely based on alanines with varying length [34]. These systems may be used to estimate the intrinsic propensity value of every amino acid for given secondary structure elements because the context-dependent effects are (mostly) excluded.

In contrast to experiments on short peptides, other authors searched databases for proteins with annotated secondary structures and counted occurrences of given amino acids in secondary structure elements, which should reflect, in turn, the likelihood to find a certain amino acid in a distinct secondary structure motif. The observed preferences are subsequently used to calculate a score for new sequences to predict their folding state [35–37]. More recent approaches focus only on the conformations of amino acids that are not assigned to a specific secondary structure motif, but lie in a random coil region leading to so-called "coil libraries" [38–40]. It has been proposed, that those libraries are better suited to derive intrinsic propensities, because they exclude the strong bias towards properly ordered motifs which leads, for example, to an overestimation of the fraction in the right-handed α -helical conformational basin. However, they depend strongly on the definition of the coil state and vary with respect to the used data set and are hence not fully context-independent [12].

To our knowledge, the analysis of post-translational modifications in this respect has been widely neglected. These chemical changes often completely alter the nature of an amino acid, so much so that the differences between them in propensities are likely to approach or even surpass those between the canonical amino acids. Therefore, we performed simulations of 45 commonly occurring post-translationally modified amino acids. We used capped ends to eliminate the effect of changed termini and the backbone dihedral angle parameters established and validated in chapter 3. These PTMs were analyzed in comparison to one another as well as to their respective precursor amino acids, elucidating both commonalities shared by them and the effects they have on the J-value ($^3J(\text{H}_\text{N}, \text{H}_\alpha)$) and on the secondary structure propensities.

4.3 Methods

The simulation protocol and the parameters are identical to the common amino acids' simulations in chapter 3, while the new phosphate parameters used for residues S1P and T1P have been taken from chapter 2. The trajectory lengths were 100 ns and the numerous simulations have been performed using the PROMETHEUS interface (chapter 6). The simulations used parameter set 54A7 for the canonical amino acids [41] and PTMs [42, 43], with the exception of the indicated phosphates. The results were subsequently predicted for our new backbone dihedral angle parameters using Hamiltonian reweighting, as outlined in chapter 3.

4.3.0.1 J-values

The J-values used for the comparison between canonical and post-translationally modified amino acids and the 54A7 and new backbone parameters, respectively, depend on the ϕ -backbone torsional angle and have been calculated by using Pardis parameters [44]. For a detailed description on the calculation, see chapter 3.

4.3.0.2 Classification

There are two ways to define and distinguish secondary structure elements in peptides and proteins. First, the hydrogen bond pattern of the backbone is often typical and is used for classification, for example, by the widely used Define Secondary Structure of Proteins (DSSP) [45] algorithm. This approach requires, however, a minimum fragment length of at least four, which is needed in order to recognize e.g. an α -helical conformation. For this reason, this approach cannot be used for the small compounds investigated in this study. Instead, we applied the algorithm described in chapter 3, making use of the ϕ and ψ dihedral angles in the backbone. Certain areas in the Ramachandran plot indicate backbone conformations associated with secondary structure elements [46].

4.4 Results

Primary data on the effects of post-translational modifications (PTMs) on secondary structure preferences is rare [15]. Table 4.1 shows the J-values and secondary structure propensities for 45 amino acid variants, when describing the backbone dihedral angle potential energy with the 54A7 force field and with the parameters suggested in chapter 3. To the best of our knowledge, table 4.1 provides the most detailed set of conformational preferences for common PTMs. First of all, this table allows us to compare the J-values and secondary structure propensities between the 54A7 parameter set and the updated backbone dihedral angle parameters. The simulated (canonical amino acids) and predicted (post-translationally modified amino acids) data for the latter parameter set show similar patterns. One of the largest shifts in terms of secondary structure for the regular amino acids was observed for threonine, where the unusually high α fraction (40.7%) was reduced to only 5.2% using the updated parameters. It is striking that similar trends could be observed for beta-hydroxyleucine (L3H), beta-hydroxyvaline (V3H) and the modified variants of threonine itself (phosphothreonine, T1P and 2-amino-ketobutyric acid, TOX) with reductions of the α fraction up to 43.6% (L3H). This indicates that inappropriately high helical fraction of threonine was not an artifact of the particular simulation, but rather of the 54A7 force field. While the post-translationally modified amino acids were not part of the parameterization efforts described in chapter 3, the new parameters do seem to be able to correct the surprisingly large α fraction for these compounds as well.

Table 4.1: J-values (in Hz) and secondary structure propensities (in %/100) obtained for commonly found post-translational modifications and their canonical amino-acid precursors. Both the values using the 54A7 backbone parameters and the ones reported in chapter 3 as well as versions of S1P and T1P using the old phosphate parameters and the ones derived in chapter 2, are reported. The values in brackets denote the change in respect to the 54A7 parameters.

		J-value [Hz]				propensities [%/100]	
		α		β		P_{II}	
CYS ^a	54A7 ^b	8.0	0.179	0.381	0.393		
	#5623 ^b	7.3 (-0.7)	0.038 (-0.141)	0.488 (0.107)	0.427 (0.034)		
CSA	54A7 ^b	7.6	0.239	0.274	0.423		
	#5623	6.9 (-0.7)	0.113 (-0.126)	0.341 (0.067)	0.462 (0.039)		
CSE	54A7 ^b	8.5	0.104	0.529	0.299		
	#5623	7.8 (-0.7)	0.021 (-0.083)	0.590 (0.061)	0.308 (0.009)		
CSN	54A7 ^b	8.1	0.122	0.386	0.432		
	#5623	7.3 (-0.8)	0.044 (-0.078)	0.429 (0.043)	0.472 (0.040)		
CYH	54A7 ^b	7.9	0.124	0.350	0.471		
	#5623	7.2 (-0.7)	0.047 (-0.077)	0.383 (0.033)	0.521 (0.050)		
CYM	54A7 ^b	8.1	0.170	0.423	0.358		
	#5623	7.4 (-0.7)	0.044 (-0.126)	0.550 (0.127)	0.351 (-0.007)		
ASP ^a	54A7 ^b	7.7	0.093	0.324	0.517		
	#5623 ^b	7.0 (-0.7)	0.056 (-0.037)	0.354 (0.030)	0.533 (0.016)		
D1P	54A7 ^b	8.2	0.129	0.419	0.394		
	#5623	7.4 (-0.8)	0.044 (-0.085)	0.448 (0.029)	0.455 (0.061)		
GLU ^a	54A7 ^b	7.6	0.126	0.264	0.559		
	#5623 ^b	6.8 (-0.8)	0.046 (-0.08)	0.269 (0.005)	0.652 (0.093)		
ECA	54A7 ^b	6.9	0.189	0.109	0.663		
	#5623	6.1 (-0.8)	0.099 (-0.090)	0.078 (-0.031)	0.800 (0.137)		
PHE ^a	54A7 ^b	8.2	0.133	0.430	0.381		
	#5623 ^b	7.4 (-0.8)	0.046 (-0.087)	0.467 (0.037)	0.441 (0.06)		
F23	54A7 ^b	8.4	0.119	0.498	0.323		
	#5623	7.7 (-0.7)	0.039 (-0.080)	0.514 (0.016)	0.394 (0.071)		
F2H	54A7 ^b	7.9	0.126	0.368	0.451		
	#5623	7.1 (-0.8)	0.056 (-0.070)	0.377 (0.009)	0.526 (0.075)		
F3H	54A7 ^b	8.3	0.113	0.471	0.358		
	#5623	7.5 (-0.8)	0.040 (-0.073)	0.494 (0.023)	0.418 (0.060)		
ARG ^a	54A7 ^b	7.8	0.156	0.291	0.497		
	#5623 ^b	7.1 (-0.7)	0.080 (-0.076)	0.303 (0.012)	0.565 (0.068)		
GSA	54A7 ^b	7.8	0.110	0.301	0.523		
	#5623	7.1 (-0.7)	0.045 (-0.065)	0.292 (-0.009)	0.582 (0.059)		
R0P	54A7 ^b	7.8	0.131	0.295	0.519		
	#5623	7.1 (-0.7)	0.056 (-0.075)	0.302 (0.007)	0.599 (0.080)		
HIS ^a	54A7 ^b	8.0	0.211	0.331	0.397		
	#5623 ^b	7.4 (-0.6)	0.060 (-0.151)	0.421 (0.090)	0.446 (0.049)		
H2X	54A7 ^b	8.1	0.136	0.422	0.386		
	#5623	7.4 (-0.7)	0.054 (-0.082)	0.462 (0.040)	0.439 (0.053)		

^a The precursor amino acids are given prior to the modified ones. The first letter of the post-translationally modified residues usually is the original amino acids' single-letter code. For a complete list of names, see <http://vienna-ptm.univie.ac.at>

^b These values have been simulated, not projected.

		J-value [Hz]	propensities [%/100]		
			α	β	P_{II}
LYS ^a	54A7 ^b	7.8	0.153	0.295	0.496
	#5623 ^b	7.1 (-0.7)	0.071 (-0.082)	0.297 (0.002)	0.554 (0.058)
K1P	54A7 ^b	7.6	0.152	0.258	0.538
	#5623	6.9 (-0.7)	0.069 (-0.083)	0.264 (0.006)	0.627 (0.089)
K2C	54A7 ^b	7.8	0.161	0.287	0.498
	#5623	7.1 (-0.7)	0.067 (-0.094)	0.301 (0.014)	0.584 (0.086)
K3C	54A7 ^b	7.8	0.135	0.301	0.510
	#5623	7.1 (-0.7)	0.057 (-0.078)	0.310 (0.009)	0.588 (0.078)
KAC	54A7 ^b	7.7	0.134	0.281	0.531
	#5623	7.0 (-0.7)	0.060 (-0.074)	0.282 (0.001)	0.617 (0.086)
KAL	54A7 ^b	7.7	0.127	0.288	0.532
	#5623	7.0 (-0.7)	0.053 (-0.074)	0.298 (0.010)	0.610 (0.078)
KCA	54A7 ^b	7.7	0.129	0.267	0.553
	#5623	7.0 (-0.7)	0.057 (-0.072)	0.270 (0.003)	0.632 (0.079)
KMC	54A7 ^b	6.8	0.000	0.265	0.676
	#5623	5.9 (-0.9)	0.000 (0.000)	0.297 (0.032)	0.578 (-0.098)
KPH	54A7 ^b	7.8	0.158	0.307	0.479
	#5623	7.1 (-0.7)	0.067 (-0.091)	0.321 (0.014)	0.564 (0.085)
LEU ^a	54A7 ^b	7.6	0.174	0.214	0.562
	#5623 ^b	6.9 (-0.7)	0.047 (-0.127)	0.231 (0.017)	0.683 (0.121)
L3H	54A7 ^b	7.1	0.486	0.074	0.412
	#86516	6.7 (-0.4)	0.050 (-0.436)	0.285 (0.211)	0.581 (0.169)
L4H	54A7 ^b	7.2	0.181	0.173	0.607
	#5623	6.5 (-0.7)	0.075 (-0.106)	0.156 (-0.017)	0.734 (0.127)
L5H	54A7 ^b	7.6	0.171	0.231	0.548
	#5623	7.0 (-0.6)	0.070 (-0.101)	0.257 (0.026)	0.629 (0.081)
LNO	54A7 ^b	7.7	0.135	0.280	0.532
	#5623	7.0 (-0.7)	0.058 (-0.077)	0.285 (0.005)	0.615 (0.083)
MET ^a	54A7 ^b	7.8	0.152	0.288	0.471
	#5623 ^b	7.1 (-0.7)	0.044 (-0.108)	0.307 (0.019)	0.603 (0.132)
MSX	54A7 ^b	7.8	0.129	0.294	0.520
	#5623	7.1 (-0.7)	0.056 (-0.073)	0.277 (-0.017)	0.625 (0.105)
MXS	54A7 ^b	7.8	0.146	0.285	0.490
	#5623	7.1 (-0.7)	0.056 (-0.090)	0.259 (-0.026)	0.564 (0.074)
ASN ^a	54A7 ^b	8.4	0.000	0.590	0.387
	#5623 ^b	7.8 (-0.6)	0.000 (0.000)	0.619 (0.029)	0.341 (-0.046)
N3H	54A7 ^b	8.0	0.179	0.371	0.413
	#86516	8.7 (0.7)	0.008 (-0.171)	0.712 (0.341)	0.211 (-0.202)
NME	54A7 ^b	8.3	0.139	0.460	0.343
	#5623	7.6 (-0.7)	0.038 (-0.101)	0.511 (0.051)	0.389 (0.046)
PRO	54A7 ^b	5.9	0.256	0.002	0.729
	#5623	6.3 (0.4)	0.141 (-0.115)	0.001 (-0.001)	0.841 (0.112)
HY2	54A7 ^b	6.5	0.175	0.002	0.816
	#86516	5.4 (-1.1)	0.013 (-0.162)	0.003 (0.001)	0.975 (0.159)
PGA	54A7 ^b	6.6	0.027	0.001	0.968
	#86516	5.7 (-0.9)	0.003 (-0.024)	0.002 (0.001)	0.992 (0.024)

^a The precursor amino acids are given prior to the modified ones. The first letter of the post-translationally modified residues usually is the original amino acids' single-letter code. For a complete list of names, see <http://vienna-ptm.univie.ac.at>

^b These values have been simulated, not projected.

		J-value [Hz]	propensities [%/100]		
			α	β	P_{II}
PH5	54A7 ^b	5.5	0.070	0.000	0.917
	#86516	4.8 (-0.7)	0.004 (-0.066)	0.000 (0.000)	0.958 (0.041)
GLN ^a	54A7 ^b	8.2	0.000	0.489	0.491
	#5623 ^b	7.0 (-1.2)	0.064 (0.064)	0.295 (-0.194)	0.597 (0.106)
Q4H	54A7 ^b	7.5	0.154	0.237	0.560
	#5623	6.9 (-0.6)	0.062 (-0.092)	0.239 (0.002)	0.655 (0.095)
QME	54A7 ^b	7.9	0.130	0.312	0.501
	#5623	7.1 (-0.8)	0.055 (-0.075)	0.321 (0.009)	0.575 (0.074)
SER ^a	54A7 ^b	8.0	0.195	0.389	0.370
	#5623 ^b	7.3 (-0.7)	0.043 (-0.152)	0.564 (0.175)	0.333 (-0.037)
S1P	54A7 ^b	7.6	0.169	0.306	0.480
	#5623	6.9 (-0.7)	0.069 (-0.100)	0.385 (0.079)	0.504 (0.024)
S1P ^c	54A7 ^b	7.2	0.342	0.230	0.398
	#5623	6.7 (-0.5)	0.128 (-0.214)	0.347 (0.117)	0.485 (0.087)
THR ^a	54A7 ^b	7.4	0.407	0.137	0.421
	#86516 ^b	7.3 (-0.1)	0.052 (-0.355)	0.409 (0.272)	0.434 (0.013)
T1P	54A7 ^b	7.3	0.257	0.191	0.488
	#86516	7.1 (-0.2)	0.045 (-0.212)	0.341 (0.150)	0.419 (-0.069)
T1P ^c	54A7 ^b	7.4	0.324	0.218	0.395
	#86516	7.0 (-0.4)	0.050 (-0.274)	0.325 (0.107)	0.463 (0.068)
TOX	54A7 ^b	8.0	0.219	0.486	0.231
	#86516	8.3 (0.3)	0.007 (-0.212)	0.708 (0.222)	0.145 (-0.086)
VAL ^a	54A7 ^b	7.5	0.124	0.233	0.601
	#86516 ^b	7.7 (0.2)	0.003 (-0.121)	0.527 (0.294)	0.428 (-0.173)
V3H	54A7 ^b	7.5	0.319	0.182	0.467
	#86516	7.4 (-0.1)	0.032 (-0.287)	0.435 (0.253)	0.469 (0.002)
TRP ^a	54A7 ^b	8.0	0.121	0.401	0.424
	#5623 ^b	7.3 (-0.7)	0.041 (-0.08)	0.472 (0.071)	0.449 (0.025)
W2H	54A7 ^b	8.1	0.136	0.419	0.394
	#5623	7.4 (-0.7)	0.050 (-0.086)	0.469 (0.050)	0.438 (0.044)
W4H	54A7 ^b	7.2	0.165	0.239	0.549
	#5623	6.6 (-0.6)	0.083 (-0.082)	0.284 (0.045)	0.591 (0.042)
W5H	54A7 ^b	8.1	0.134	0.414	0.401
	#5623	7.3 (-0.8)	0.054 (-0.080)	0.484 (0.070)	0.424 (0.023)
W6H	54A7 ^b	8.1	0.143	0.399	0.404
	#5623	7.3 (-0.8)	0.056 (-0.087)	0.462 (0.063)	0.440 (0.036)
W7H	54A7 ^b	8.0	0.125	0.412	0.413
	#5623	7.3 (-0.7)	0.052 (-0.073)	0.465 (0.053)	0.447 (0.034)
TYR ^a	54A7 ^b	8.3	0.114	0.473	0.356
	#5623 ^b	7.5 (-0.8)	0.031 (-0.083)	0.498 (0.025)	0.429 (0.073)
Y1P	54A7 ^b	8.2	0.124	0.443	0.380
	#5623	7.5 (-0.7)	0.042 (-0.082)	0.471 (0.028)	0.442 (0.062)
HTY	54A7 ^b	8.4	0.108	0.490	0.344
	#5623	7.6 (-0.8)	0.036 (-0.072)	0.516 (0.026)	0.402 (0.058)

^a The precursor amino acids are given prior to the modified ones. The first letter of the post-translationally modified residues usually is the original amino acids' single-letter code. For a complete list of names, see <http://vienna-ptm.univie.ac.at>

^b These values have been simulated, not projected.

^c Using new phosphate-parameters (chapter 2).

Next, the data in table 4.1 offers the unprecedented opportunity to estimate the effect of post-translational modifications on the J-value and the intrinsic secondary structure preferences of the amino acids. In the following, we only consider the data that was obtained for the updated parameter sets. In figure 4.1, we describe the average effect post-translational modifications have over the complete set of 45 modifications (labeled #5623 and #86516) as well as a separation into 41 modifications that do not have a second branch at C_β (labeled as #5623) and 4 modifications that are further branched at C_β (labeled as #86516). For the unbranched amino acids and PTMs the general trend is that a significant loss in the β fraction is split between the α and P_{II} fractions, respectively. Accordingly, the J-value is reduced, due to the shift in preferred ϕ dihedral angle. The opposite seems to be the case for the C_β -branched amino acids and PTMs, although the low number of data points and large spread in the data ask for caution.

We could not observe any general trends, e.g. correlating the distance or hydrophobicity of the modifications to the shifts in propensities. While the average modifications may seem moderate, individual PTMs do show large shifts in terms of J-values and propensities. The average absolute shift in J-value amounts to about 0.3 Hz, coming from individual shifts in the range of -1.2 Hz (methyllysine, KMC) to +1 Hz (TOX). The largest changes in propensities are observed for TOX (increase of the β fraction by 30%) and phosphoserine (decrease of the β fraction by 22%). Other prominent examples are a decrease of the β fraction by 19% for 4-carboxyglutamate and 4-hydroxytryptophan and marked increases of 7-8% in the α fractions of KMC and cysteine sulfinic acid (CSA). Bielska and Zondlo [19] have proposed, that the phosphorylation of both serine and threonine residues in the context of tau protein fragments could potentially act as the main source for a change of the conformation towards the P_{II} basin. The hyperphosphorylated form of tau is one of the main constituents of the protein plaques causing Alzheimers disease [19] and the phosphorylation of more than 30 sites (mainly in proline-rich regions) in this protein is associated with structural changes. Indeed, we observe a shift from β to P_{II} (see table 4.1) for both the phosphorylation of serine (compare SER vs. SIP) and threonine (compare THR vs. TIP) of about 15% and 3%, respectively, using the latest force field parameters (backbone dihedrals and phosphate partial charges). Therefore, the observed change in the polypeptide chain secondary structure equilibrium is indeed likely to be at least partially explained by the change of the modified amino acids' local propensities.

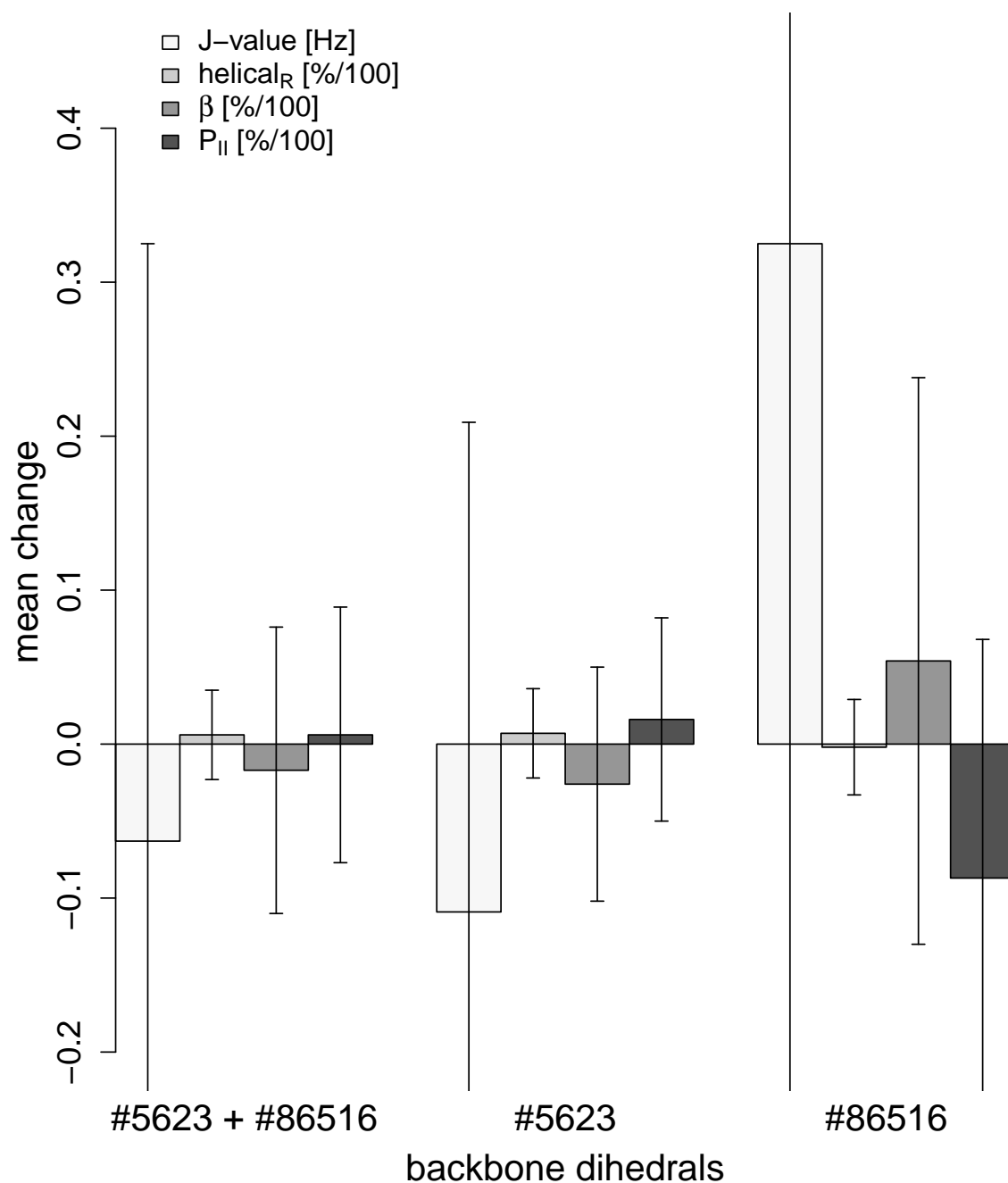


Figure 4.1: Shows the mean effect of the post-translational modifications as compared to their precursors (both using the updated backbone parameters, either #5623 or #86516). The group using #5623 amounts to 41, the one using #86516 to 4 members, respectively. Note, that for L3H and N3H the modifications lead to a change in the used backbone parameters because of an addition to their C_β atoms.

4.5 Conclusion

Based on the optimized parameters described in chapter 3, we were able to predict the J-values and conformational propensities of 45 post-translationally modified blocked amino acids. To the best of our knowledge, this is the most extensive and detailed dataset for PTMs that is currently available. The fact that it remains difficult to identify general trends in the (significant) shifts that were observed, emphasizes that the connection between chemical structure and conformational preference is highly complex and cannot be guessed from simple principles. Apparently, PTMs do not only broaden the chemical properties that may be incorporated into proteins, but may also directly influence overall protein structure and stability. While experimental production of the specific post-translationally modified amino acids is rather complicated, computational methods allow us to directly assess the effect of (small) chemical changes on the local structure. Considering the poorly understood effects of PTMs, this is a highly relevant field of research that should be investigated in more detail in the future. Most concretely, the predictions for the PTMs in table 4.1 should be confirmed by performing the actual MD simulations. Furthermore, this study suggests that experiments on selected post-translationally modified blocked amino acids (which may be accessible synthetically) would be worthwhile.

References

- [1] A. Chakrabartty, T. Kortemme, and R. L. Baldwin. “Helix propensities of the amino acids measured in alanine-based peptides without helix-stabilizing side-chain interactions.” In: *Protein Science : A Publication of the Protein Society* 3.5 (May 1994), pp. 843–852.
- [2] C. N. Pace and J. M. Scholtz. “A helix propensity scale based on experimental studies of peptides and proteins.” In: *Biophysical Journal* 75.1 (July 1998), pp. 422–427.
- [3] Melissa A. Kelly, Brian W. Chellgren, Adam L. Rucker, Jerry M. Troutman, Michael G. Fried, Anne-Frances Miller, and Trevor P. Creamer. “Host-Guest Study of Left-Handed Polyproline II Helix Formation”. In: *Biochemistry* 40.48 (2001), pp. 14376–14383. DOI: [10.1021/bi011043a](https://doi.org/10.1021/bi011043a).
- [4] Adam L. Rucker, Cara T. Pager, Margaret N. Campbell, Joseph E. Qualls, and Trevor P. Creamer. “Host-guest scale of left-handed polyproline II helix formation”. In: *Proteins: Structure, Function, and Bioinformatics* 53.1 (2003), pp. 68–75. DOI: [10.1002/prot.10477](https://doi.org/10.1002/prot.10477).
- [5] Jože Grdadolnik, Simona Golic Grdadolnik, and Franc Avbelj. “Determination of Conformational Preferences of Dipeptides Using Vibrational Spectroscopy”. In: *The Journal of Physical Chemistry B* 112.9 (Mar. 2008), pp. 2712–2718. DOI: [10.1021/jp7096313](https://doi.org/10.1021/jp7096313).
- [6] J. Grdadolnik, V. Mohacek-Grosev, R. L. Baldwin, and F. Avbelj. “Populations of the three major backbone conformations in 19 amino acid dipeptides”. en. In: *Proceedings of the National Academy of Sciences* 108.5 (Feb. 2011), pp. 1794–1798. DOI: [10.1073/pnas.1017317108](https://doi.org/10.1073/pnas.1017317108).
- [7] Robert B. Best, David de Sancho, and Jeetain Mittal. “Residue-specific α -helix propensities from molecular simulation”. In: *Biophysical Journal* 102.6 (Mar. 2012), pp. 1462–1467. DOI: [10.1016/j.bpj.2012.02.024](https://doi.org/10.1016/j.bpj.2012.02.024).
- [8] Alaina M. Brown and Neal J. Zondlo. “A Propensity Scale for Type II Polyproline Helices (PPII): Aromatic Amino Acids in Proline-Rich Sequences Strongly Disfavor PPII Due to ProlineAromatic Interactions”. In: *Biochemistry* 51.25 (June 2012), pp. 5041–5051. DOI: [10.1021/bi3002924](https://doi.org/10.1021/bi3002924).
- [9] Kwang-Im Oh, Kyung-Koo Lee, Eun-Kyung Park, Youngae Jung, Geum-Sook Hwang, and Minhaeng Cho. “A comprehensive library of blocked dipeptides reveals intrinsic backbone conformational propensities of unfolded proteins”. In: *Proteins: Structure, Function, and Bioinformatics* 80.4 (2012), pp. 977–990. DOI: [10.1002/prot.24000](https://doi.org/10.1002/prot.24000).
- [10] Anthony Hazel, Christophe Chipot, and James C. Gumbart. “Thermodynamics of Deca-alanine Folding in Water”. In: *Journal of Chemical Theory and Computation* 10.7 (2014), pp. 2836–2844. DOI: [10.1021/ct5002076](https://doi.org/10.1021/ct5002076).
- [11] David A. C. Beck, Darwin O. V. Alonso, Daigo Inoyama, and Valerie Daggett. “The intrinsic conformational propensities of the 20 naturally occurring amino acids and reflection of these propensities in proteins”. In: *Proceedings of the National Academy of Sciences* 105.34 (Aug. 2008), pp. 12259–12264. DOI: [10.1073/pnas.0706527105](https://doi.org/10.1073/pnas.0706527105).

- [12] Andrew Hagarman, Thomas J. Measey, Daniel Mathieu, Harald Schwalbe, and Reinhard Schweitzer-Stenner. “Intrinsic Propensities of Amino Acid Residues in GxG Peptides Inferred from Amide I Band Profiles and NMR Scalar Coupling Constants”. In: *Journal of the American Chemical Society* 132.2 (2010), pp. 540–551. DOI: [10.1021/ja9058052](https://doi.org/10.1021/ja9058052).
- [13] Robert J. Moreau, Christian R. Schubert, Khaled A. Nasr, Marianna Török, Justin S. Miller, Robert J. Kennedy, and Daniel S. Kemp. “Context-Independent, Temperature-Dependent Helical Propensities for Amino Acid Residues”. In: *Journal of the American Chemical Society* 131.36 (2009), pp. 13107–13116. DOI: [10.1021/ja904271k](https://doi.org/10.1021/ja904271k).
- [14] S. M. Kelly and N. C. Price. “The use of circular dichroism in the investigation of protein structure and function”. In: *Current Protein & Peptide Science* 1.4 (Dec. 2000), pp. 349–384.
- [15] Charles D. Andrew, Jim Warwicker, Gareth R. Jones, and Andrew J. Doig. “Effect of phosphorylation on alpha-helix stability as a function of position”. eng. In: *Biochemistry* 41.6 (Feb. 2002), pp. 1897–1905.
- [16] Arno Bundi and Kurt Wüthrich. “¹H-nmr parameters of the common amino acid residues measured in aqueous solutions of the linear tetrapeptides H-Gly-Gly-X-L-Ala-OH”. In: *Biopolymers* 18.2 (1979), pp. 285–297. DOI: [10.1002/bip.1979.360180206](https://doi.org/10.1002/bip.1979.360180206).
- [17] Liang Ding, Kang Chen, Paul A. Santini, Zhengshuang Shi, and Neville R. Kallenbach. “The Pentapeptide GGAGG Has PII Conformation”. In: *Journal of the American Chemical Society* 125.27 (July 2003), pp. 8092–8093. DOI: [10.1021/ja035551e](https://doi.org/10.1021/ja035551e).
- [18] Jurgen Graf, Phuong H. Nguyen, Gerhard Stock, and Harald Schwalbe. “Structure and Dynamics of the Homologous Series of Alanine Peptides: A Joint Molecular Dynamics NMR Study”. In: *Journal of the American Chemical Society* 129.5 (Feb. 2007), pp. 1179–1189. DOI: [10.1021/ja0660406](https://doi.org/10.1021/ja0660406).
- [19] Agata A. Bielska and Neal J. Zondlo. “Hyperphosphorylation of Tau Induces Local Polyproline II Helix”. In: *Biochemistry* 45.17 (May 2006), pp. 5527–5537. DOI: [10.1021/bi052662c](https://doi.org/10.1021/bi052662c).
- [20] Kyle A. Beauchamp, Yu-Shan Lin, Rhiju Das, and Vijay S. Pande. “Are Protein Force Fields Getting Better? A Systematic Benchmark on 524 Diverse NMR Measurements”. In: *Journal of chemical theory and computation* 8.4 (Apr. 2012), pp. 1409–1414. DOI: [10.1021/ct2007814](https://doi.org/10.1021/ct2007814).
- [21] Siobhan Toal, Derya Meral, Daniel Verbaro, Brigita Urbanc, and Reinhard Schweitzer-Stenner. “pH-Independence of Tri-alanine and the Effects of Termini Blocking in Short Peptides: A Combined Vibrational, NMR, UVCD, and Molecular Dynamics Study”. In: *The Journal of Physical Chemistry B* 117.14 (Apr. 2013), pp. 3689–3706. DOI: [10.1021/jp310466b](https://doi.org/10.1021/jp310466b).
- [22] Andreja Mirtič, Franci Merzel, and Jože Grdadolnik. “The amide III vibrational circular dichroism band as a probe to detect conformational preferences of alanine dipeptide in water”. In: *Biopolymers* 101.7 (July 2014), pp. 814–818. DOI: [10.1002/bip.22460](https://doi.org/10.1002/bip.22460).
- [23] Fatma Eker, Xiaolin Cao, Laurence Nafie, and Reinhard Schweitzer-Stenner. “Tripeptides Adopt Stable Structures in Water. A Combined Polarized Visible Raman, FTIR, and VCD Spectroscopy Study”. In: *Journal of the American Chemical Society* 124.48 (Dec. 2002), pp. 14330–14341. DOI: [10.1021/ja027381w](https://doi.org/10.1021/ja027381w).

- [24] Silvia Pizzanelli, Claudia Forte, Susanna Monti, Giorgia Zandomenoghi, Andrew Hagarman, Thomas J. Measey, and Reinhard Schweitzer-Stenner. “Conformations of Phenylalanine in the Tripeptides AFA and GFG Probed by Combining MD Simulations with NMR, FTIR, Polarized Raman, and VCD Spectroscopy”. In: *The Journal of Physical Chemistry B* 114.11 (2010), pp. 3965–3978. DOI: [10.1021/jp907502n](https://doi.org/10.1021/jp907502n).
- [25] Robert B. Best, Nicolae-Viorel Buchete, and Gerhard Hummer. “Are Current Molecular Dynamics Force Fields too Helical?” In: *Biophysical Journal* 95.1 (July 2008), pp. L07–L09. DOI: [10.1529/biophysj.108.132696](https://doi.org/10.1529/biophysj.108.132696).
- [26] A. G. Anderson and J. Hermans. “Microfolding: conformational probability map for the alanine dipeptide in water from molecular dynamics simulations”. In: *Proteins* 3.4 (1988), pp. 262–265. DOI: [10.1002/prot.340030408](https://doi.org/10.1002/prot.340030408).
- [27] Franc Avbelj, Simona Golic Grdadolnik, Joze Grdadolnik, and Robert L. Baldwin. “Intrinsic backbone preferences are fully present in blocked amino acids”. In: *Proceedings of the National Academy of Sciences of the United States of America* 103.5 (Jan. 2006), pp. 1272–1277. DOI: [10.1073/pnas.0510420103](https://doi.org/10.1073/pnas.0510420103).
- [28] Victor L. Cruz, Javier Ramos, and Javier Martinez-Salazar. “Assessment of the Intrinsic Conformational Preferences of Dipeptide Amino Acids in Aqueous Solution by Combined Umbrella Sampling/MBAR Statistics. A Comparison with Experimental Results”. In: *The Journal of Physical Chemistry B* 116.1 (2011), pp. 469–475. DOI: [10.1021/jp206757j](https://doi.org/10.1021/jp206757j).
- [29] Fatma Eker, Kai Griebenow, Xiaolin Cao, Laurence A. Nafie, and Reinhard Schweitzer-Stenner. “Preferred peptide backbone conformations in the unfolded state revealed by the structure analysis of alanine-based (AXA) tripeptides in aqueous solution”. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.27 (July 2004), pp. 10054–10059. DOI: [10.1073/pnas.0402623101](https://doi.org/10.1073/pnas.0402623101).
- [30] Andrea Motta, Meital Rechtes, Lucia Pappalardo, Giuseppina Andreotti, and Ehud Gazit. “The preferred conformation of the tripeptide Ala-Phe-Ala in water is an inverse gamma-turn: implications for protein folding and drug design”. In: *Biochemistry* 44.43 (Nov. 2005), pp. 14170–14178. DOI: [10.1021/bi050658v](https://doi.org/10.1021/bi050658v).
- [31] Reinhard Schweitzer-Stenner. “Distribution of Conformations Sampled by the Central Amino Acid Residue in Tripeptides Inferred From Amide I Band Profiles and NMR Scalar Coupling Constants”. In: *The Journal of Physical Chemistry B* 113.9 (2009), pp. 2922–2932. DOI: [10.1021/jp8087644](https://doi.org/10.1021/jp8087644).
- [32] Kevin W. Plaxco, Craig J. Morton, Shaun B. Grimshaw, Jonathan A. Jones, Maureen Pitkeathly, Iain D. Campbell, and Christopher M. Dobson. “The effects of guanidine hydrochloride on the random coil conformations and NMR chemical shifts of the peptide series GGXGG”. In: *Journal of Biomolecular NMR* 10.3 (Oct. 1997), pp. 221–230. DOI: [10.1023/A:1018340217891](https://doi.org/10.1023/A:1018340217891).
- [33] Zhengshuang Shi, Kang Chen, Zhigang Liu, Angela Ng, W. Clay Bracken, and Neville R. Kallenbach. “Polyproline II propensities from GGXGG peptides reveal an anticorrelation with β -sheet scales”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.50 (Dec. 2005), pp. 17964–17968. DOI: [10.1073/pnas.0507124102](https://doi.org/10.1073/pnas.0507124102).

- [34] Reinhard Schweitzer-Stenner, Thomas Measey, Lazaros Kakalis, Frank Jordan, Silvia Pizzanelli, Claudia Forte, and Kai Griebenow. “Conformations of Alanine-Based Peptides in Water Probed by FTIR, Raman, Vibrational Circular Dichroism, Electronic Circular Dichroism, and NMR Spectroscopy”. In: *Biochemistry* 46.6 (Feb. 2007), pp. 1587–1596. DOI: [10.1021/bi0622241](https://doi.org/10.1021/bi0622241).
- [35] Peter Y. Chou and Gerald D. Fasman. “Prediction of protein conformation”. In: *Biochemistry* 13.2 (1974), pp. 222–245. DOI: [10.1021/bi00699a002](https://doi.org/10.1021/bi00699a002).
- [36] J. Kyngas and J. Valjakka. “Unreliability of the Chou-Fasman parameters in predicting protein secondary structure”. eng. In: *Protein Engineering* 11.5 (May 1998), pp. 345–348.
- [37] Susan Costantini, Giovanni Colonna, and Angelo M. Facchiano. “Amino acid propensities for secondary structures are influenced by the protein structural class”. In: *Biochemical and Biophysical Research Communications* 342.2 (Apr. 2006), pp. 441–451. DOI: [10.1016/j.bbrc.2006.01.159](https://doi.org/10.1016/j.bbrc.2006.01.159).
- [38] Lauren L. Perskie, Timothy O. Street, and George D. Rose. “Structures, basins, and energies: a deconstruction of the Protein Coil Library”. eng. In: *Protein Science: A Publication of the Protein Society* 17.7 (July 2008), pp. 1151–1161. DOI: [10.1110/ps.035055.108](https://doi.org/10.1110/ps.035055.108).
- [39] Mark B. Swindells, Malcolm W. MacArthur, and Janet M. Thornton. “Intrinsic ϕ, ψ propensities of amino acids, derived from the coil regions of known structures”. In: *Nature Structural & Molecular Biology* 2.7 (July 1995), pp. 596–603. DOI: [10.1038/nsb0795-596](https://doi.org/10.1038/nsb0795-596).
- [40] Abhishek K. Jha, Andres Colubri, Muhammad H. Zaman, Shohei Koide, Tobin R. Sosnick, and Karl F. Freed. “Helix, Sheet, and Polyproline II Frequencies and Strong Nearest Neighbor Effects in a Restricted Coil Library”. In: *Biochemistry* 44.28 (July 2005), pp. 9691–9702. DOI: [10.1021/bi0474822](https://doi.org/10.1021/bi0474822).
- [41] Nathan Schmid, Andreas P. Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E. Mark, and Wilfred F. van Gunsteren. “Definition and testing of the GROMOS force-field versions 54A7 and 54B7”. In: *European Biophysics Journal* 40.7 (July 1, 2011), pp. 843–856. DOI: [10.1007/s00249-011-0700-9](https://doi.org/10.1007/s00249-011-0700-9).
- [42] Christian Margreitter, Drazen Petrov, and Bojan Zagrovic. “Vienna-PTM web server: a toolkit for MD simulations of protein post-translational modifications”. In: *Nucleic Acids Research* 41.Web Server issue (July 2013), W422–426. DOI: [10.1093/nar/gkt416](https://doi.org/10.1093/nar/gkt416).
- [43] Drazen Petrov, Christian Margreitter, Melanie Grandits, Chris Oostenbrink, and Bojan Zagrovic. “A Systematic Framework for Molecular Dynamics Simulations of Protein Post-Translational Modifications”. In: *PLoS Computational Biology* 9.7 (July 2013), e1003154. DOI: [10.1371/journal.pcbi.1003154](https://doi.org/10.1371/journal.pcbi.1003154).
- [44] A. Pardi, M. Billeter, and K. Wüthrich. “Calibration of the angular dependence of the amide proton-C alpha proton coupling constants, 3JHN alpha, in a globular protein. Use of 3JHN alpha for identification of helical secondary structure”. In: *Journal of Molecular Biology* 180.3 (Dec. 1984), pp. 741–751.
- [45] W. Kabsch and C. Sander. “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features”. In: *Biopolymers* 22.12 (Dec. 1983), pp. 2577–2637. DOI: [10.1002/bip.360221211](https://doi.org/10.1002/bip.360221211).

- [46] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. “Stereochemistry of polypeptide chain configurations”. In: *Journal of Molecular Biology* 7 (July 1963), pp. 95–99.

Anti-body humanization guided by molecular dynamics simulations

Adapted from:

Margreitter C., Mayrhofer P., Kunert R., Oostenbrink C., *Anti-body humanization guided by molecular dynamics simulations*, J. Mol. Recogn. (2015), 29, 266-275, doi: <https://dx.doi.org/10.1002/jmr.2527>

Author contributions:

CM performed and analyzed all simulations, PM performed and analyzed all wet-lab experiments, RK and CO supervised the work and assisted in writing the manuscript.

5.1 Abstract

Monoclonal antibodies represent the fastest growing class of biotherapeutic proteins. However, as they are often initially derived from rodent organisms, there is a severe risk of immunogenic reactions, hampering their applicability. The humanization of these antibodies remains a challenging task in the context of rational drug design. "Superhumanization" describes the direct transfer of the complementarity determining regions to a human germline framework, but this humanization approach often results in loss of binding affinity. In this study, we present a new approach for predicting promising backmutation sites using molecular dynamics simulations of the model antibody Ab2/3H6. The simulation method was developed in close conjunction with novel specificity experiments. Binding properties of mAb variants were evaluated directly from crude supernatants and confirmed using established binding affinity assays for purified antibodies. Our approach provides access to the dynamical features of the actual binding sites of an antibody, based solely on the antibody sequence. Thus we do not need structural data on the antibody-antigen complex and circumvent cumbersome methods to assess binding affinities.

5.2 Introduction

Monoclonal antibodies (mAbs) represent the fastest growing class of biotherapeutic proteins, with US sales reaching \$24.6 billion in 2012 [1]. As of January 2015, IMGT®[2], the international ImMunoGeneTics information system (<http://www.imgt.org>) listed 36 monoclonal antibodies approved by the FDA or EMEA for human therapeutic use [3].

Only 12 of the approved antibodies are of human origin, whereas the majority represents rodent (3), chimeric (7) or humanized antibodies (14), all containing non-human sequences. However, antibodies derived from non-human organisms can lead to the HAMA response (human anti-mouse antibody) when applied as remedies, due to their foreign characteristics, leading to adverse and potentially harmful side-effects due to altered efficacy and pharmacokinetics [4–6]. This indicates the importance for techniques to reduce immunogenicity of antibodies by making them more human-like. Traditional methods to reduce the risk of severe immunogenic responses to therapeutic antibodies are based on chimerization [7–9] or complementarity-determining region (CDR)-grafting [10–12]. Further advanced protocols include resurfacing [13], framework shuffling [14], human content optimization [15], superhumanization [16, 17], screening of human antibody libraries [18–20] or immunization of transgenic mice [21–24]. The primary purpose of all of these methods is to keep the risk of adverse side-effects [25] at an absolute minimum. Applied to human subjects, the engineered therapeutic antibody must not trigger any critical human anti-mouse [4–6] or human anti-chimeric [26] antibody responses, while the full biological function should be maintained, quantified by a high binding affinity. However, extensive sequence modifications within the framework regions (FR) during the trial and error based humanization process often result in reduced or even lost binding affinities [27, 28]. This effect may be attributed to critical framework positions within the antibody framework sequence, which stabilize the overall protein structure or the VH/VL interface [29], contact the antigen directly [30] or establish the Vernier zone [31] by providing a suitable physico-chemical environment for a proper conformational ensemble of the CDR loops. In the first step of humanization, the non-human framework regions are replaced by carefully selected, appropriate human framework sequences. Afterwards, several critical positions within the human framework have to be backmutated to the non-human wild-type. Currently, no universally applicable humanization protocol is available that allows the straightforward, concurrent maintenance of the binding affinity and reduction of the risk for immunogenic responses, i.e. the lowest number of backmutations necessary. These choices often have to be made based on empirical knowledge gained from iterative rounds of antibody design, expression and in-vitro binding evaluation on a case-by-case basis, making antibody humanization an unpredictable, time-consuming and costly undertaking. It would hence be highly advantageous to predict the effect of potential backmutations on the binding affinity of mutants, not only because the mere number of potential candidates is tremendous but also because there is an urgent need to understand the underlying physico-chemical mechanisms.

Yet, the assessment of binding affinities (i.e. the free energy of binding), which are usually determined experimentally, by computational tools remains a very demanding task. Docking lacks accuracy (mainly because of the imposed rigidity of bigger molecules), while free energy calculations using molecular dynamics (MD) simulations require structural data on the complex and are far from being readily applied to interactions involving a large molecular interface. Nevertheless, in-silico techniques may prove to be a useful addition to the humanization process. In this study, we perform molecular dynamics (MD) simulations to analyse and predict CDR conformations in the humanization process of a mAb. By providing in-silico knowledge from molecular dynamics simulations, proper decisions about critical backmutations can be made, before testing the designed variants on the bench. In such a prospective design cycle, many different humanized variants, containing different backmutations, might be assessed in-silico. The dynamic behaviour of the CDRs is monitored by simulation and expressed in a score that represents the similarity to the wild-type, the known binder. The most promising mutants are then selected for

expression and measurement of binding affinities by experiments. With our technique, we allow for pre-selection of various humanized variants, thus reducing the amount of required experiments during humanization significantly.

In this study the anti-idiotypic antibody Ab2/3H6 directed against the broadly neutralizing anti-HIV-1 antibody 2F5 [32, 33] was used as model protein. It was developed from mouse hybridoma [34] and subsequently chimerized [35] or humanized by CDR-grafting, resurfacing or superhumanization [36]. Although not eliciting HIV-1 neutralizing antibodies in first prime/boost studies in BALB/c mice [37] or rabbits [38], it served as a template for different humanization approaches and molecular dynamics simulations [39] based on the resolved crystal structure [40]. The superhumanized variant, su3H6, has lost the binding affinity completely and is therefore suited to be the negative control for the simulation [36]. An antibody panel consisting of several humanized 3H6 mutants was tested for binding in-vitro to refine a similarity score, quantifying the similarity to the original wild-type antibody (wt3H6). The optimized in-silico system was then tested to predict the influence of backmutations on the binding affinity in superhumanized variants.

5.3 Methods

5.3.1 Expression of mAbs

Cell cultures were cultivated in vented 125 mL Erlenmeyer flasks (Corning) on a climo-shaker ISF1-XC (Kuhner) at 140 rpm, 37°C, 7% CO₂ and 85% humidity. mAb variants used for training of the MD system (TR01-TR06) were expressed using stable transfected cell pools of a serum-free adapted host cell line CHO-K1 (ATCC CCL-61) cultivated in ProCHO5 medium (Lonza, Cat. No. BE12-766Q) supplemented with 4 mM L-glutamine (Biochrom, Cat. No. K0302), 15 mg/L phenol red (Sigma, Cat. No. P0290) and 0.5 mg/mL G418 (Biochrom, Cat. No. A2912).

For studying the effect of backmutations (BM) in heavy chain framework regions of the superhumanized Ab2/3H6 variants, transient expression was performed in HEK293-6E cells (NRC biotechnology Research Institute) [41] cultured in F17 medium (Invitrogen, Life Technologies, Cat. No. A13835-01) supplemented with 4 mM L-glutamine (Biochrom, Cat. No. K0302), 0.1% Kolliphor P188 (Sigma-Aldrich, Cat. No. K4894), 15 mg/L phenol red (Sigma-Aldrich, Cat. No. P0290) and 25 µg/mL G418 (Biochrom, Cat. No. A2912). Transfection of pCEP4 vector (Invitrogen, Cat. No. V044-50) was mediated by polyethylenimine (PEI) transfection using linear 25kDa PEI (Polysciences, Cat. No. 23966).

Culture supernatants were subjected to concentration by either using Amicon Ultra Centrifugal Filters (0.5 mL, NMWCO 10 kDa, Millipore, Cat. No. UFC501096) or Millipore-Labscale TFF system (Millipore, equipped with a 30 kDa Pellicon cassette (Millipore, Cat. No. PXB030A50) followed by antibody purification by protein A affinity chromatography using the ÄKTA Purifier Station (GE Healthcare) equipped with a HiTrap MabSelect SuRe protein A column (GE Healthcare, Cat. No. 29-0491-04).

5.3.2 Preparation of mAb variants

The mAb variants used for training of the MD system were expressed in CHO-K1 cells with stable cell pools and purified by protein A chromatography (TR01 - TR06). For studying

backmutations (BM) in framework regions of the heavy chain, transient expression was performed in HEK293-6E cells for different superhumanized Ab2/3H6 variants to allow assessment of re-established binding affinities. After a seven-day production phase, crude culture supernatants were concentrated and diluted in ForteBio kinetics buffer.

5.3.3 Affinity evaluation of mAb variants

All binding studies based on bio-layer interferometry were performed on a ForteBio Octet QK system (Pall ForteBio) equipped with streptavidin (Pall ForteBio, Cat. No. 18-5019) or protein A biosensors (Pall ForteBio, Cat. No. 18-5010). For immobilization to streptavidin biosensors purified antibody 2F5 was biotinylated using the EZ-Link NHS-PEG4-Biotin kit (Thermo Scientific, Cat. No. 21329). Samples and biosensors were equilibrated in kinetics buffer (ForteBio). The streptavidin biosensors were loaded with 20 $\mu\text{g/mL}$ biotinylated 2F5 antibody, before k_{on} and k_{off} were measured of purified mAb variants by monitoring association and dissociation in kinetics buffer.

Capturing monoclonal antibodies from crude culture supernatants for quantification already has been proven as an established and robust procedure [42]. In this study we also demonstrate the use of protein A biosensors for determining binding affinity of the immobilized antibodies to its antigen (anti-idiotypic antibody). Protein A sensors were equilibrated in kinetics buffer for 60 seconds before transiently expressed Ab2/3H6 variants were immobilized from the crude and concentrated culture supernatants for 1200 seconds (figure 5.1 A). This time period for capturing antibodies from crude culture supernatants was considered as a good trade-off between reaching a suitable sensor saturation level and overall assay time. To block possible free protein A binding sites a blocking step of 1200 seconds was introduced by submersing the loaded protein A biosensors in high concentration (100 $\mu\text{g/mL}$) of purified mAb su3H6, showing no interaction with 2F5. Following a baseline / washing step for 120 seconds, the association of purified target antibody mAb 2F5 (100 $\mu\text{g/mL}$) to protein A-immobilized Ab2/3H6 variants was measured followed by a dissociation step in kinetics buffer only. mAb 2F5 binding to anti-idiotypic mutants showed a high response for wt3H6 samples of about 3.5 nm in the baseline corrected sensogram (figure 5.1 B). Both crude and pure wt3H6 preparations at high concentrations gave similar binding curves. Additionally, also with a very low initial wt3H6 concentration (1.5 $\mu\text{g/mL}$) a high binding response could be observed reaching a response level of 3.4 nm. These results demonstrate that it is possible to distinguish mAb 2F5-binders (i.e. wt3H6) from non-binders (i.e. su3H6), the latter of which gave only a very low non-specific response level probably resulting from minimal residual free protein A binding sites. From the experimentally obtained K_d values, the average binding free energies were calculated to be compared to the score obtained from the simulations. To eliminate outliers, data sets have been excluded for which the fit to the theoretical signal curve was calculated to have a correlation coefficient (R^2) below or equal to 0.8. Afterwards, binding free energies were calculated using $\Delta G_{\text{binding}} = k_B T * \ln(K_d)$ for all remaining measurements. Measurements, deviating more than 5.6 kJ/mol from the average (factor 10 in K_d) were iteratively excluded from the calculation. From the remaining values (at least 3 measurements for each variant), the average binding free energy is reported.

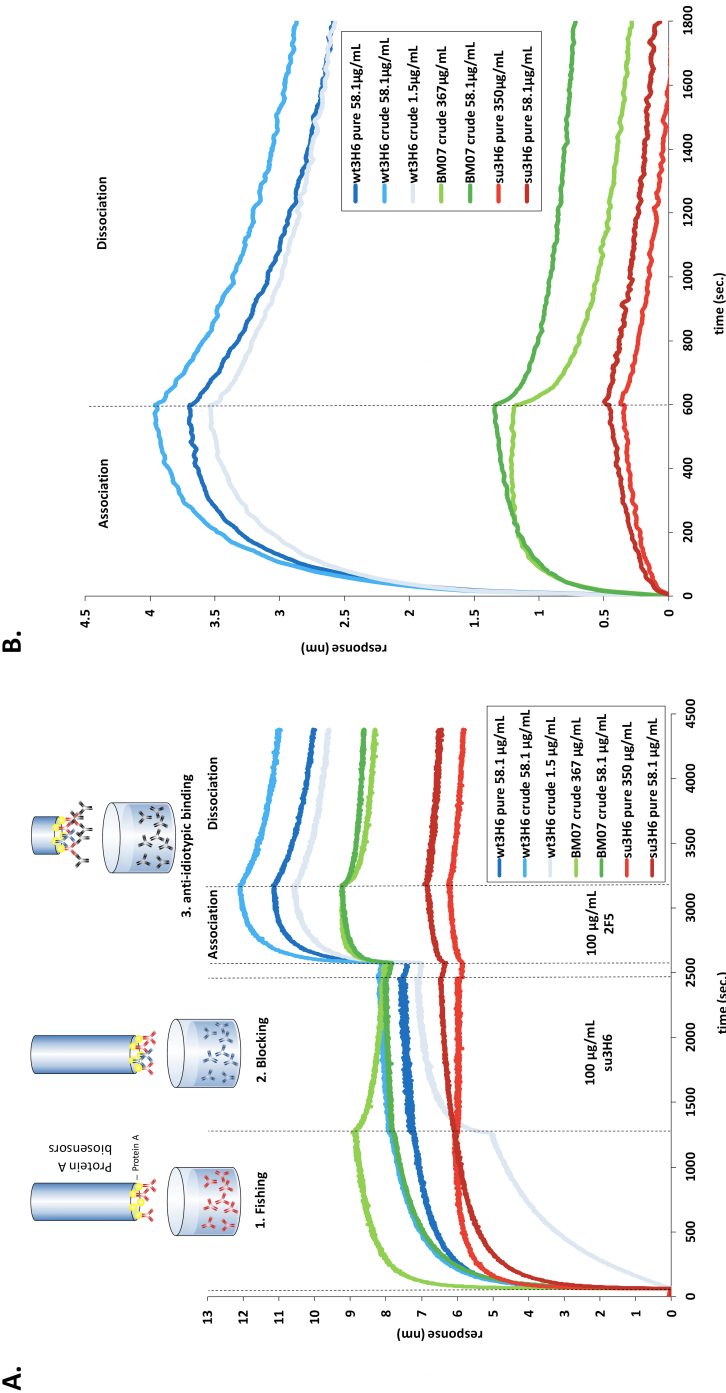


Figure 5.1: Protein A fishing from crude culture supernatants by bio-layer interferometry. The ForteBio Octet system was equipped with protein A biosensors to immobilize transiently expressed humanized Ab2/3H6 mutants from concentrated and crude culture supernatants. (A) Real-time sensorgram of wt3H6, su3H6 and BM07 at different concentrations. Assay-step times were as follows: 60-s baseline in kinetics buffer, 1. Fishing: 1200-s immobilization of antibodies from crude culture supernatants, 2. Blocking with 100 µg/ml purified mAb su3H6 for 1200 s, followed by 120-s baseline/washing in kinetics buffer, 3. Binding measurements: 600-s mAb 2 F5 association (100 µg/ml) with immobilized Ab2/3H6 variants, followed by 1200-s dissociation in kinetics buffer only. (B) Association and dissociation curves extracted from raw data and aligned to baseline by the fortebio software.

5.3.4 In-silico score calculation

The approach presented here relies solely on the structural and dynamic information retrieved from the monoclonal antibody, as shown in figure 5.2. From multiple simulations on the murine antibody, we obtain the most prominent conformations of the CDRs, represented by the central member structures (CMS) of conformational clusters. Subsequently, variants are simulated and the reproduction of the wild-type reference conformations (CMS) is expressed through a similarity score. It is based on time series of the root-mean-square deviation (RMSD) of the CDR atoms (fitted to the flanking framework backbone; see below) with respect to the wild-type CMS. This means that the score is higher for variants, which are closer to the original rodent antibody in terms of their structural ensembles. The score is calculated according to,

$$\text{score} = \frac{100}{s \cdot m \cdot a} \sum_{i=1}^a \sum_{j=1}^s \sum_{k=1}^n \sum_{x=1}^m \begin{cases} 1, & \text{RMSD}_{x, \text{CMS}_i} \leq c_k \\ 0, & \text{else} \end{cases} \quad (5.1)$$

where c is a vector of thresholds used, $\text{RMSD}_{x, \text{CMS}}$, is calculated for a given configuration-central member structure pair (in nanometers), m is the number of configurations in a trajectory, n is the number of thresholds considered, a is the number of significant clusters and s is the number of replicate simulations for this very variant. In an initial training round, the scores are compared to the binding free energy, experimentally determined by affinity measurements for some variants, to estimate a cutoff of the similarity score.

In the second stage, backmutations of the superhumanized variant are simulated until a candidate with a score above the cutoff is identified, which can then be further optimized. Our approach is based on the assumption that mutants with comparable structures and dynamics as the original monoclonal antibody also show significant binding. Obviously, the reverse statement is not necessarily true as other conformations / ensembles may bind as well or even better but are disregarded in our approach since they were not present in the murine reference. Furthermore, we assume that induced fit effects upon binding play a minor role and the relevant pre-binding conformations will be sufficiently sampled in the MD simulations, following the conformational selection paradigm [43, 44]. We have applied this workflow to the murine / wild-type anti-idiotypic Ab2/3H6 antibody, which is directed against the broadly neutralizing anti-HIV-1 antibody 2F5 and has been studied by our groups earlier [32, 34, 36, 39]. The binding to mAb 2F5 is mainly facilitated by the third CDR loop of the heavy chain (Ab2/3H6), which simplifies the analysis. In order to cover cases that require multiple loops for proper binding, the above equation can readily be extended. We defined six training variants based on critical framework positions (supplementary table 5.6) of the murine / wild-type Ab2/3H6 (TR01 to TR06; figure 5.3), to train the scoring function and eleven backmutation variants based on the non-binding su3H6 antibody (BM01 to BM11; figure 5.4), with mutation sites in the FRs selected based on their location in the X-ray structure, vicinity to the Vernier zone, or being in the VH/VL interface region.

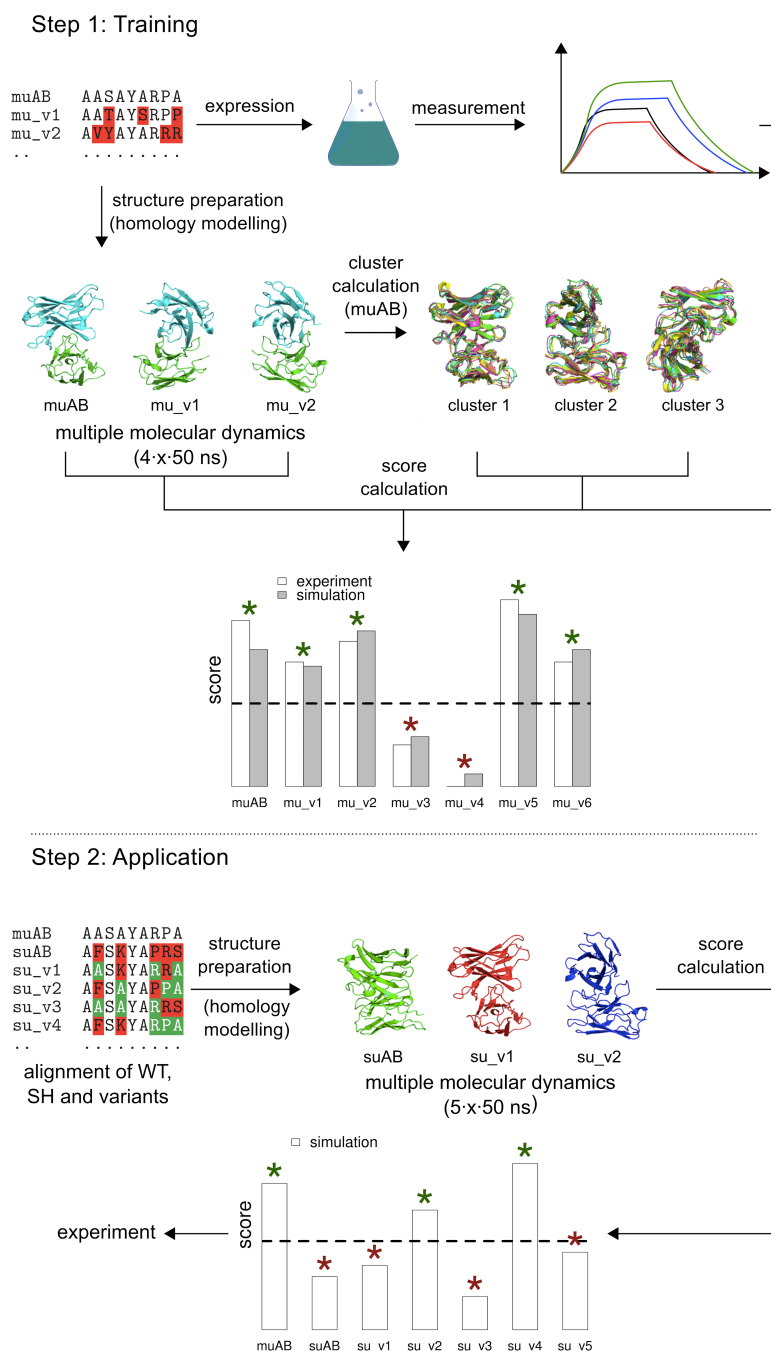


Figure 5.2: Workflow of the simulation assisted humanization approach. In the training step (above), molecular dynamics simulations of the murine derived wild-type and selected mutant variants are performed and assessed in terms of a score (see equation 5.1), representing similarity to the wild-type loop conformations. The same set is expressed, and affinities are measured experimentally, allowing for the identification of the qualitative boundary separating binders from non-binders (the latter marked with a red asterisk). The second step (below) starts with the superhumanized antibody variant, and a set with selected backmutations. With the same procedure as before, a computational score can be calculated holding qualitative information on the expected binding affinities. The cutoff determined in the previous step can now be used to classify the results. Note, that this is only an illustrative diagram; the actual values obtained throughout this study are reported in figures 5.5 and 5.6 and table 5.1.

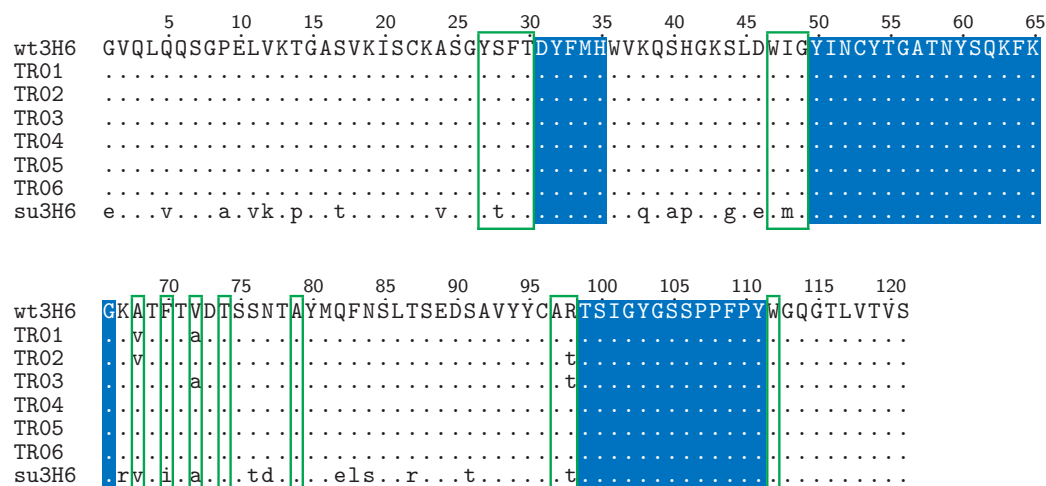
A**B**

Figure 5.3: Sequence alignment of the heavy (A) and light (B) chain of the wild-type (wt3H6), the training (TR01-TR06) and the superhumanized (su3H6) variant. The CDR regions as defined by Kabat were conserved during the humanization process and kept constant in all variants throughout this study (blue background). Identical amino acids compared to the murine/wild-type template sequence are indicated as dots. Vernier zone residues of the heavy chain defined by Foote and Winter are marked by green frames.

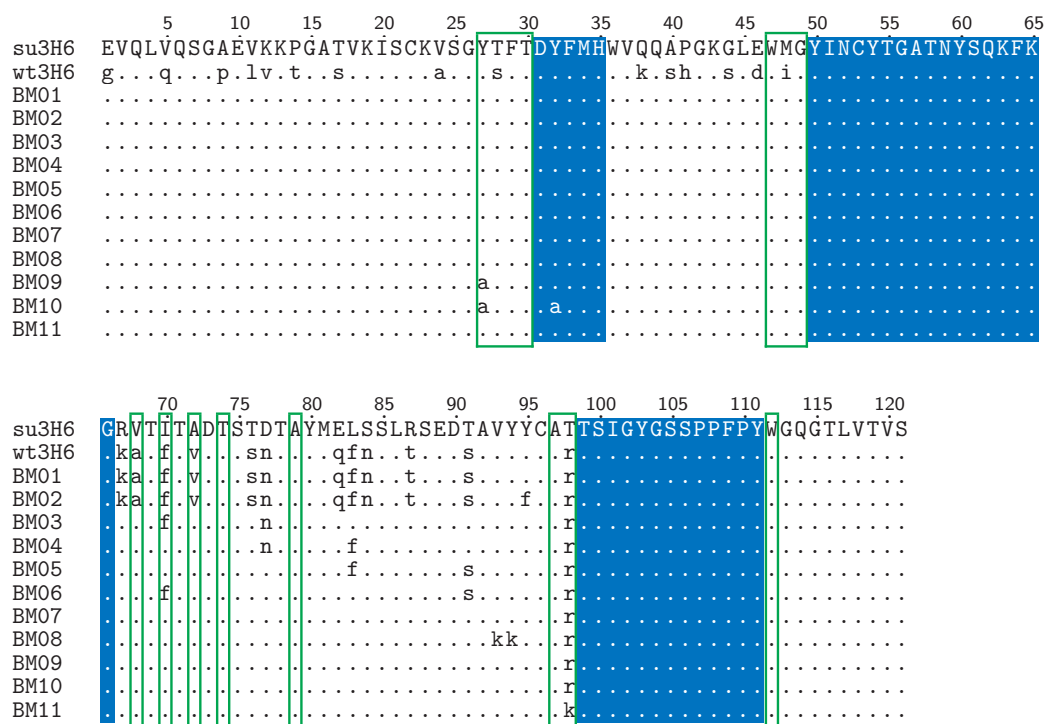


Figure 5.4: Sequence alignment of the heavy chain of the wild-type (wt3H6), the superhumanized (su3H6) and the simulated (predictive, BM01-BM11) variants. The CDR regions as defined by Kabat were conserved during the humanization process and kept constant in all variants (but BM10) throughout this study (blue background). Identical amino acids compared to the murine template sequence are indicated as dots. Vernier zone residues of the heavy chain defined by Foote and Winter are marked by green frames. No light chain mutations in respect to the superhumanized variant were applied to the backmutation variants.

5.3.5 Molecular dynamics simulations

For wt3H6 (the binding reference), six simulations (of 100 ns each) of the VH/VL complex were undertaken. For the compounds in the training set, four simulations (replicates) for each variant were performed with a trajectory length of 50 ns, respectively. The superhumanized version and its derivatives have been simulated five times each (50 ns). Initial coordinates were taken from the crystal structure of the 3H6-2F5 complex (PDB ID: 3BQU) [40], variants were modelled using the program MOE [45]. The simulations were performed without any restraints. All simulations were performed using the GRO-MOS [46, 47] simulation package with the 54A7 parameter set [48, 49] in a sufficiently large water-box (0.8 nm minimum solute to box-wall distance). Counter-ions were added (Na^+ and Cl^-) to neutralize the net charge in the box, up to a limit of 15 ions of each type, e.g. to a solute with net charge -9 e, 15 Na^+ and 6 Cl^- were added. The rectangular periodic simulation boxes (roughly 5.5 x 6.5 x 7.5 nm) contained approximately 27000 atoms (275 nm^3). Prior to the production runs, the systems were equilibrated from 60 K to 300 K in six discrete steps with a simulation length of 20 ps each and a weak thermostat-coupling with two baths for the solute and solvent (relaxation time of 0.1 ps) and weak barostat-coupling (relaxation time of 0.5 ps and an isothermal compressibility of $4.575 \cdot 10^{-4} (\text{kJmol}^{-1}\text{nm}^{-3})^{-1}$). In order to avoid artefacts originating from the same starting structure, only the last 40 ns of each trajectory have been analysed, while for cluster analysis the last 90 ns of the wt3H6 replicates have been used. All simulations in the training set were extended to 100 ns (90 ns analysed) without a significant change in the subsequent analyses (supplementary figure 5.9), therefore an additional replicate was preferred over longer simulation times in the prediction set. For consistency with the superhumanized and predicted variants, all the reported data is based on the last 40 ns of the first 50 ns of the training set simulations. Weak temperature and barostat-coupling [50] ensured a constant temperature of 300 K and a constant pressure of 1 atm, respectively. SHAKE [51] was used to maintain the bond distances at the energy minimum. The integration time step used was 2 fs. Interactions within 0.8 nm were calculated at every time step from a pairlist that was updated every five steps. Intermediate range interactions up to a distance of 1.4 nm were calculated at pairlist updates and kept constant between updates. Long range interactions were approximated with a reaction field contribution [52] to the energies and forces, accounting for a homogeneous medium with a relative dielectric constant [53] of 61 beyond the cut-off of 1.4 nm.

5.3.6 Fitting procedure

In order to compare the conformational ensembles generated by the molecular dynamics simulations to one another, the respective backbone atoms of framework regions 3 and 4 of the heavy chain (H:FR3 and H:FR4) were used for a roto-translational least-squares fit. The RMSD calculation afterwards (both for the clustering and the score calculation) was based on all atoms (including side-chain atoms) of complementary determining region 3 of the heavy chain (H:CDR3).

5.4 Results and Discussion

All MD simulations lead to stable trajectories with backbone atom-positional RMSD values to the initial structure of, on average, 0.3 (± 0.1) nm for the training set and 0.3 (± 0.1) nm for the superhumanized variants (supplementary figures 5.10 and 5.11). The secondary structure elements of the framework regions (anti-parallel β -sheets) were maintained throughout. Averaged over simulation time and the framework region residues, which are also in β -sheet conformation in the crystal structure, the occurrence of β -sheet conformations amounted to 88% ($\pm 2\%$) for the training set and 83% ($\pm 4\%$) for the superhumanized variants as determined by the DSSP (determine secondary structure of proteins) algorithm (supplementary figure 5.11). To calculate the reference structure(s), that are most representative of the conformational ensemble of H:CDR3 in wt3H6, we calculated the cross RMSD matrix of structures collected from all its replicates. A clustering algorithm was applied, using a cut-off of 0.2 nm [54]. The majority of configurations belonged to the first cluster (52%), while the others were only populated by small amounts (up to 15%), suggesting to use a single representative structure, i.e. $a = 1$ in equation 1. Without loss of generality, the subsequent analysis could have included additional clusters. Subsequently, the similarity score was computed for the training set, which is shown in figure 5.5 together with the experimentally determined binding free energies.

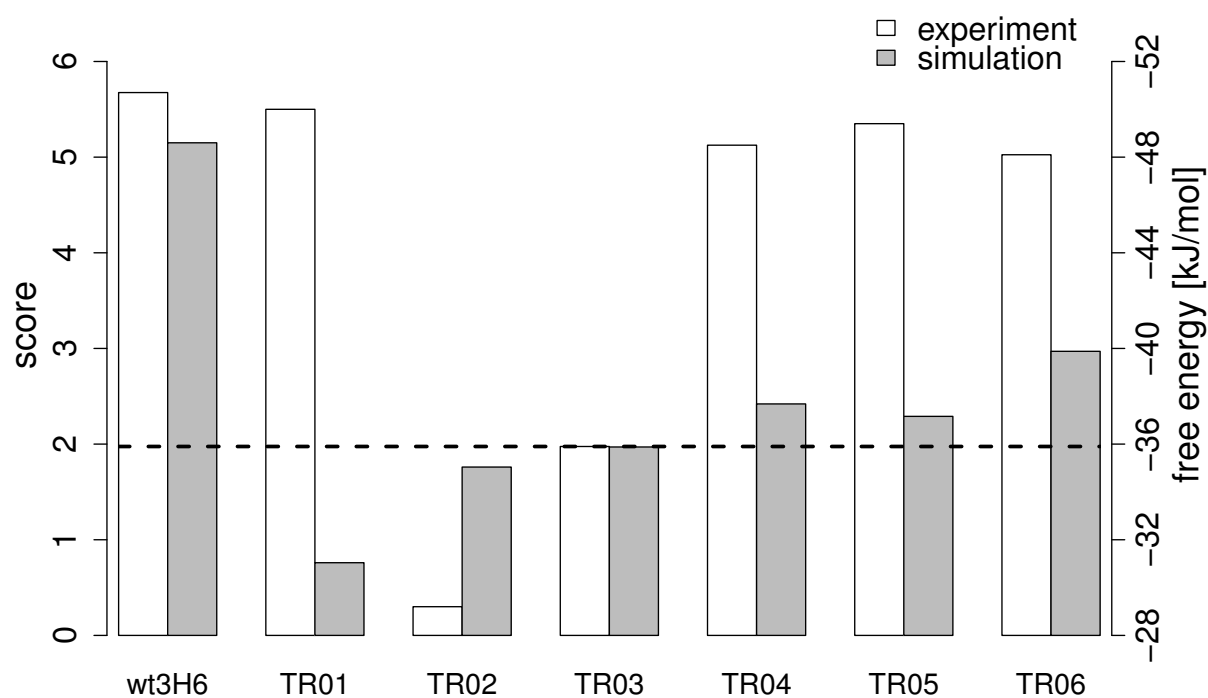


Figure 5.5: Similarity scores and free energies of binding for the variants. For the calculation of the score, see equation 5.1. The grey bars indicate a score for the similarity to the murine/wild-type 3H6 antibody (left axis), while the white bars indicate experimentally determined binding free energies (right axis). For TR01, the simulation score and the experimentally determined binding free energy are clearly disagreeing, see main text.

From the MD simulations, TR01, TR02 and TR03 were most dissimilar to the original wt3H6. Indeed, for TR02 and TR03 the binding affinity seems to be reduced by 15 - 20 kJ/mol, while the mutations applied to TR04 to TR06 do not seem to affect affinity. Only for TR01, there is no match between the similarity score and the measured binding affinity to 2F5, possibly because of alternative binding modes (see above). A superhumanized variant (su3H6) was described earlier, that lost binding affinity completely (figure 5.6) [36].

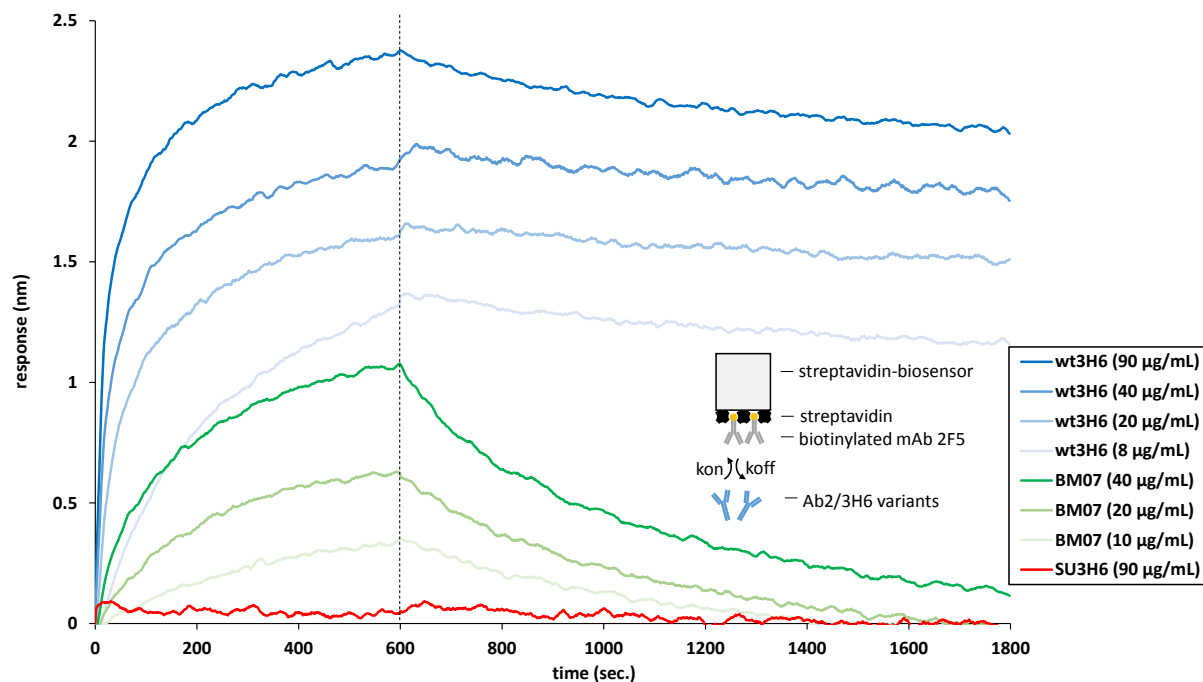


Figure 5.6: Real-time bio-layer interferometry sensorgrams for determination of anti-idiotypic binding affinities of purified Ab2/3H6 variants to its target antibody mAb 2F5. Streptavidin biosensors were loaded with biotinylated mAb 2 F5 (20 µg/ml) followed by a washing/baseline step. Association (600 s) and dissociation (1200 s) of different Ab2/3H6 variants were measured at different concentrations or buffer only, respectively.

Our MD simulations confirm that the CDR is significantly different (similarity score of 1.2). Eleven candidates with specific backmutations (BM01 to BM11) were proposed and are represented in figure 5.4. The associated similarity scores for these variants are given in table 5.1. Based on the results of the training simulations, the emphasis for the back-mutations was placed on the framework region 3 of the heavy chain. In the first variant, BM01, the entire framework region was backmutated to the murine / wild-type sequence. The similarity score was low, indicating that BM01 is not a variant that is to be tested experimentally. Interestingly, variant BM02, which contains the additional mutation Y95F, leads to a significantly higher similarity score. The side-chain of Y95 is pointing into the interface region between the VL and VH domains, suggesting that the packing of amino acids at the interface and in the hydrophobic core plays a crucial role. To maintain optimal packing with the human residues from the other framework regions, variants with fewer backmutations were proposed in the form of BM03 to BM07. The most interesting variant in this respect is variant BM07, in which only a single backmutation, T98R, was applied and for which a reasonably high similarity score was observed. Next, we proposed an artificial double mutation to lysine at positions 93 and 94 in variant BM08 to have

Variant	Score
su3H6	1.0
BM01	2.6
BM02	3.3
BM03	3.5
BM04	1.7
BM05	1.8
BM06	2.8
BM07	2.8
BM08	1.3
BM09	2.9
BM10	0.8
BM11	3.2

Table 5.1: *In-silico* score predictions for all simulated variants based on the (non-binding) su3H6 antibody.

a non-binding variant (negative control) that still contains the T98R mutation. Indeed, the similarity score dropped considerably. Variant BM07 was transiently expressed using the HEK293-6E system. Because of low transient expression titers the affinity to 2F5 was measured by a modified protein A fishing setup (see above): Protein A biosensors were immersed in the supernatant to bind the transiently expressed variant. After blocking the remaining binding sites of the sensors with inactive su3H6, the antigen 2F5 was added and binding and dissociation could be measured. This superhumanized variant, containing only a single backmutation (BM07) showed a significant improvement in anti-idiotypic binding affinity to mAb 2F5 resulting in final response levels of 1.4 nm at two different concentrations. Although the binding affinity of BM07 did not reach the full binding capabilities of wt3H6, it showed a significant increase with respect to its precursor molecule, i.e. the non-binding su3H6 antibody (figure 5.1 B). Based on these qualitative results we expressed BM07 on a larger scale followed by protein A chromatography purification and quantitative assessment of its binding properties using the streptavidin bio-assay setup (figure 5.6). The results of the two different methods were qualitatively comparable, indicating that our method for estimating binding affinities from the supernatant is reliable. Unfortunately, difficulties with expression efficiency have so far prevented the validation of additional superhumanized variants. Both our simulations and the experiments show, that the backmutation T98R is sufficient to restore binding affinity to a significant extent (figure 5.7).

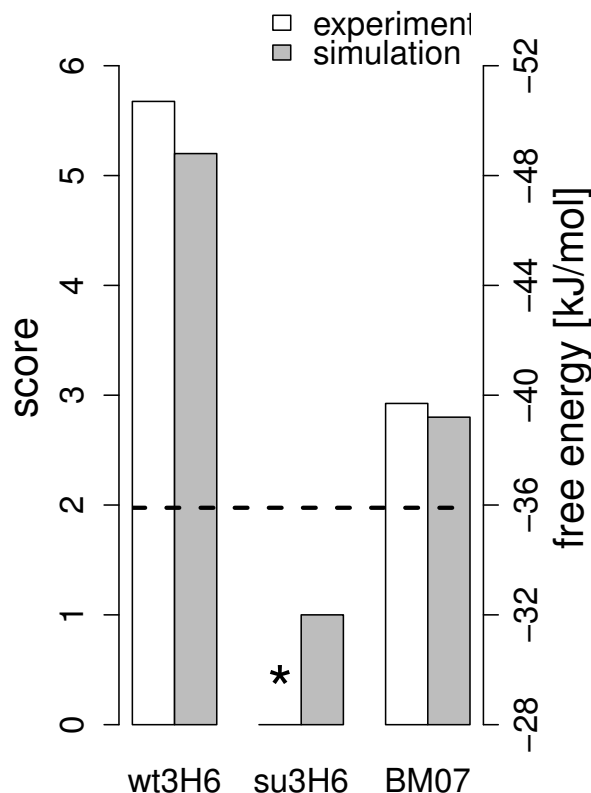


Figure 5.7: Similarity scores and free energies of binding for the wild-type, the superhumanized antibody and BM07. The experimental binding free energy of su3H6 was below the detection limit.

This confirms that a single backmutation can be sufficient to (partially) restore binding affinity for superhumanized variants and also indicates a crucial role for R98, which could be successfully predicted by our in-silico approach. In fact, 80% of the human heavy chain germline gene sequences retrieved from the IMGT/Gene-DB databank [2] have an Arg at this very position (supplementary table 5.3). By replacing this residue by threonine, as in su3H6, TR02 and TR03, binding affinity is severely reduced. From the simulations we observe that the charged side chain of R98 interacts with the Y27, Y32, Y111 side-chains (in some sort of "tyrosine-cage") and the T99 backbone (figure 5.8). It seems, that through the cation- π -interactions in this structural region, certain conformational restrictions are applied to the CDR3 loop (heavy chain), which are crucial for binding. To test this hypothesis, we also investigated other variants computationally (BM09 to BM11; table 5.1, figure 5.4) which will be the subject of future experimental validation studies. It is remarkable that the double mutation of Y27 and Y32 (BM10) to alanine is predicted to lead to a severe loss of binding, while the single mutation of Y27 (BM09) seems to make no difference. Moreover, a substitution of R98 by lysine (BM11), which also contains a positively charged moiety at a comparable distance to the backbone, seems to be able to function appropriately.

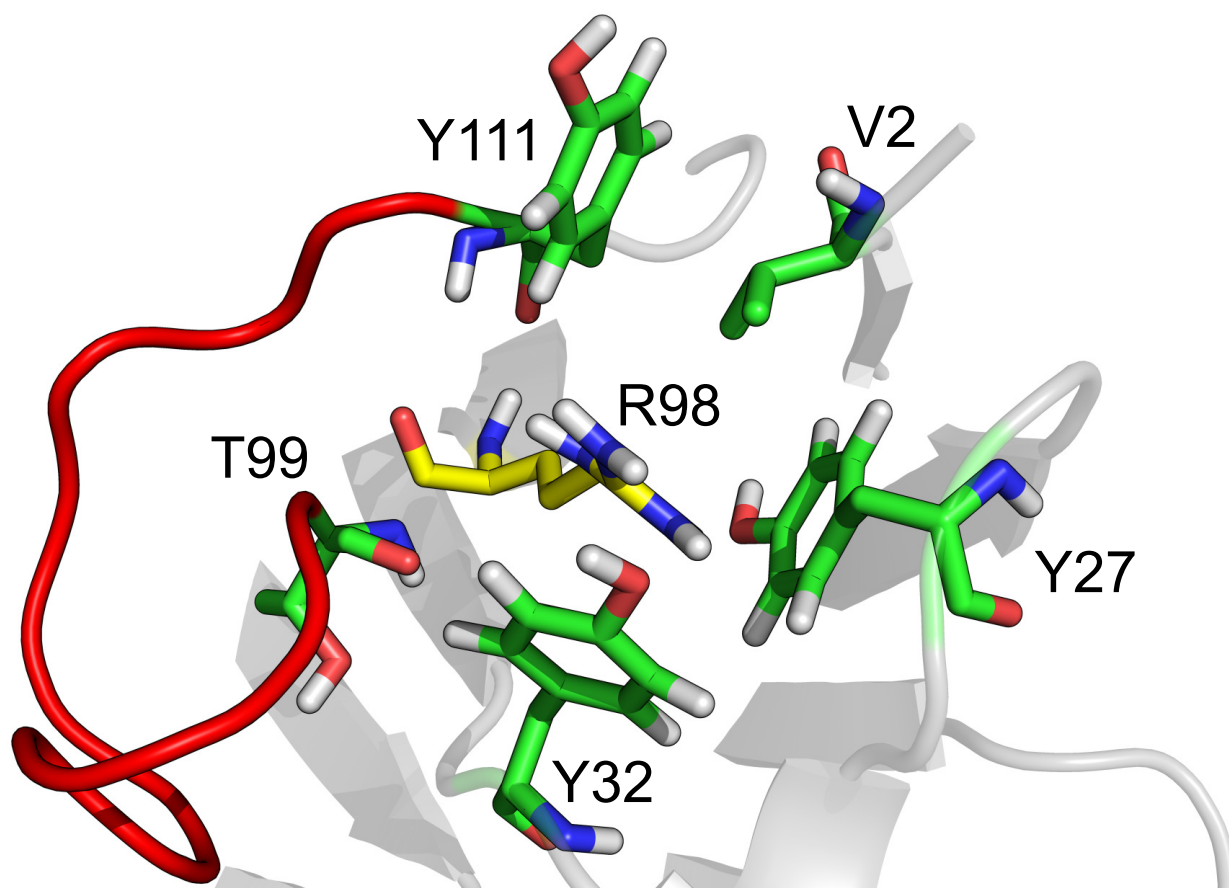


Figure 5.8: The importance of R98 is likely to be explained by its interaction with the surrounding tyrosines through cation-pi interactions, most notably with one located in CDR loop three (Y111). Therefore, it is of utmost importance that this position remains an arginine in order to retain binding affinity. The "tyrosine cage" at that position could, in conjunction with R98, lead to a more restricted local environment for the CDR loop, which is shown in red.

5.5 Conclusion

We conclude that the presented approach, calibrated with experimental data, allows for useful predictions of the effect that distinct backmutations have on the binding affinities of antibody variants. We have validated our predictions using well-established experimental techniques and also shown the qualitative agreement of these results with a newly developed efficient method, based on cell culture supernatant. Our computational workflow can be applied as an *ab-initio* protocol, however, additional information such as the relative importance of the respective CDR loops in binding, might be included. Moreover, simulations allow for a molecular rationalization of the observed differences, which may help to guide further rounds of compound design where necessary. The approach is readily applicable to different antibodies as only structural information on the original antibody is required and no explicit binding free energy calculations are performed. Presumably, iterative cycles of improvement will be necessary to re-establish a binding affinity, comparable to that of the wild-type, where promising combinations of the previous round are natural candidates.

References

- [1] Rob Aggarwal. “What’s fueling the biotech engine - 2012 to 2013”. In: *Nature Biotechnology* 32.1 (Jan. 2014), pp. 32–39. DOI: [10.1038/nbt.2794](https://doi.org/10.1038/nbt.2794).
- [2] Marie-Paule Lefranc et al. “IMGT, the international ImMunoGeneTics information system”. In: *Nucleic Acids Research* 33 (Database issue Jan. 2005), pp. D593–597. DOI: [10.1093/nar/gki065](https://doi.org/10.1093/nar/gki065).
- [3] Claire Poirion, Yan Wu, Chantal Ginestoux, Francois Ehrenmann, Patrice Duroux, and Marie-Paule Lefranc. “IMGT/mAb-DB: the IMGT database for therapeutic monoclonal antibodies.” In: *Journées Ouvertes Biol Inform Mathématiques* (2010).
- [4] R. W. Schroff, K. A. Foon, S. M. Beatty, R. K. Oldham, and A. C. Morgan. “Human anti-murine immunoglobulin responses in patients receiving monoclonal antibody therapy”. In: *Cancer Research* 45.2 (Feb. 1985), pp. 879–885.
- [5] C. Sgro. “Side-effects of a monoclonal antibody, muromonab CD3/orthoclone OKT3: bibliographic review”. In: *Toxicology* 105.1 (Dec. 20, 1995), pp. 23–29.
- [6] D. L. Shawler, R. M. Bartholomew, L. M. Smith, and R. O. Dillman. “Human immune response to multiple injections of murine monoclonal IgG”. In: *Journal of Immunology (Baltimore, Md.: 1950)* 135.2 (Aug. 1985), pp. 1530–1535.
- [7] G. L. Boulianne, N. Hozumi, and M. J. Shulman. “Production of functional chimaeric mouse/human antibody”. In: *Nature* 312.5995 (Dec. 1984), pp. 643–646.
- [8] S. L. Morrison, M. J. Johnson, L. A. Herzenberg, and V. T. Oi. “Chimeric human antibody molecules: mouse antigen-binding domains with human constant region domains”. In: *Proceedings of the National Academy of Sciences of the United States of America* 81.21 (Nov. 1984), pp. 6851–6855.
- [9] M. S. Neuberger, G. T. Williams, E. B. Mitchell, S. S. Jouhal, J. G. Flanagan, and T. H. Rabbitts. “A hapten-specific chimaeric IgE antibody with human physiological effector function”. In: *Nature* 314.6008 (Mar. 21, 1985), pp. 268–270.
- [10] P. T. Jones, P. H. Dear, J. Foote, M. S. Neuberger, and G. Winter. “Replacing the complementarity-determining regions in a human antibody with those from a mouse”. In: *Nature* 321.6069 (June 1986), pp. 522–525. DOI: [10.1038/321522a0](https://doi.org/10.1038/321522a0).
- [11] C. Queen et al. “A humanized antibody that binds to the interleukin 2 receptor”. In: *Proceedings of the National Academy of Sciences of the United States of America* 86.24 (Dec. 1989), pp. 10029–10033.
- [12] Lutz Riechmann, Michael Clark, Herman Waldmann, and Greg Winter. “Reshaping human antibodies for therapy”. In: *Nature* 332.6162 (Mar. 24, 1988), pp. 323–327. DOI: [10.1038/332323a0](https://doi.org/10.1038/332323a0).
- [13] M. A. Roguska et al. “Humanization of murine monoclonal antibodies through variable domain resurfacing”. In: *Proceedings of the National Academy of Sciences of the United States of America* 91.3 (Feb. 1, 1994), pp. 969–973.
- [14] William F. Dall’Acqua, Melissa M. Damschroder, Jingli Zhang, Robert M. Woods, Lusiana Widjaja, Julie Yu, and Herren Wu. “Antibody humanization by framework shuffling”. In: *Methods (San Diego, Calif.)* 36.1 (May 2005), pp. 43–60. DOI: [10.1016/j.ymeth.2005.01.005](https://doi.org/10.1016/j.ymeth.2005.01.005).

- [15] Greg A. Lazar, John R. Desjarlais, Jonathan Jacinto, Sher Karki, and Philip W. Hammond. “A molecular immunology approach to antibody humanization and functional optimization”. In: *Molecular Immunology* 44.8 (Mar. 2007), pp. 1986–1998. DOI: [10.1016/j.molimm.2006.09.029](https://doi.org/10.1016/j.molimm.2006.09.029).
- [16] William Ying Khee Hwang, Juan Carlos Almagro, Timothy N. Buss, Philip Tan, and Jefferson Foote. “Use of human germline genes in a CDR homology-based approach to antibody humanization”. In: *Methods (San Diego, Calif.)* 36.1 (May 2005), pp. 35–42. DOI: [10.1016/j.ymeth.2005.01.004](https://doi.org/10.1016/j.ymeth.2005.01.004).
- [17] Philip Tan, David A. Mitchell, Timothy N. Buss, Margaret A. Holmes, Claudio Anasetti, and Jefferson Foote. ““Superhumanized” antibodies: reduction of immunogenic potential by complementarity-determining region grafting with human germline sequences: application to an anti-CD28”. In: *Journal of Immunology (Baltimore, Md.: 1950)* 169.2 (July 15, 2002), pp. 1119–1125.
- [18] Andrew R. M. Bradbury, Sachdev Sidhu, Stefan Dübel, and John McCafferty. “Beyond natural antibodies: the power of in vitro display technologies”. In: *Nature Biotechnology* 29.3 (Mar. 2011), pp. 245–254. DOI: [10.1038/nbt.1791](https://doi.org/10.1038/nbt.1791).
- [19] N. M. Low, P. H. Holliger, and G. Winter. “Mimicking somatic hypermutation: affinity maturation of antibodies displayed on bacteriophage using a bacterial mutator strain”. In: *Journal of Molecular Biology* 260.3 (July 1996), pp. 359–368.
- [20] G. Winter, A. D. Griffiths, R. E. Hawkins, and H. R. Hoogenboom. “Making antibodies by phage display technology”. In: *Annual Review of Immunology* 12 (1994), pp. 433–455. DOI: [10.1146/annurev.iy.12.040194.002245](https://doi.org/10.1146/annurev.iy.12.040194.002245).
- [21] M. Brüggemann, C. Spicer, L. Buluwela, I. Rosewell, S. Barton, M. A. Surani, and T. H. Rabbitts. “Human antibody production in transgenic mice: expression from 100 kb of the human IgH locus”. In: *European Journal of Immunology* 21.5 (May 1991), pp. 1323–1326. DOI: [10.1002/eji.1830210535](https://doi.org/10.1002/eji.1830210535).
- [22] Nils Lonberg. “Human antibodies from transgenic animals”. In: *Nature Biotechnology* 23.9 (Sept. 2005), pp. 1117–1125. DOI: [10.1038/nbt1135](https://doi.org/10.1038/nbt1135).
- [23] M. J. Mendez et al. “Functional transplant of megabase human immunoglobulin loci recapitulates human antibody response in mice”. In: *Nature Genetics* 15.2 (Feb. 1997), pp. 146–156. DOI: [10.1038/ng0297-146](https://doi.org/10.1038/ng0297-146).
- [24] L. D. Taylor, C. E. Carmack, S. R. Schramm, R. Mashayekh, K. M. Higgins, C. C. Kuo, C. Woodhouse, R. M. Kay, and N. Lonberg. “A transgenic mouse that expresses a diversity of human sequence heavy and light chain immunoglobulins”. In: *Nucleic Acids Research* 20.23 (Dec. 11, 1992), pp. 6287–6295.
- [25] Trevor T. Hansel, Harald Kropshofer, Thomas Singer, Jane A. Mitchell, and Andrew J. T. George. “The safety and side effects of monoclonal antibodies”. In: *Nature Reviews. Drug Discovery* 9.4 (Apr. 2010), pp. 325–338. DOI: [10.1038/nrd3003](https://doi.org/10.1038/nrd3003).
- [26] M. B. Khazaeli, R. M. Conry, and A. F. LoBuglio. “Human immune response to monoclonal antibodies”. In: *Journal of Immunotherapy with Emphasis on Tumor Immunology: Official Journal of the Society for Biological Therapy* 15.1 (Jan. 1994), pp. 42–52.
- [27] Camellia W. Adams, David E. Allison, Kelly Flagella, Leonard Presta, Janet Clarke, Noel Dybdal, Kathleen McKeever, and Mark X. Sliwkowski. “Humanization of a recombinant monoclonal antibody to produce a therapeutic HER dimerization inhibitor, pertuzumab”. In: *Cancer immunology, immunotherapy: CII* 55.6 (June 2006), pp. 717–727. DOI: [10.1007/s00262-005-0058-x](https://doi.org/10.1007/s00262-005-0058-x).

- [28] L. G. Presta, S. J. Lahr, R. L. Shields, J. P. Porter, C. M. Gorman, B. M. Fendly, and P. M. Jardieu. “Humanization of an antibody directed against IgE”. In: *Journal of Immunology (Baltimore, Md.: 1950)* 151.5 (Sept. 1, 1993), pp. 2623–2632.
- [29] C. Chothia et al. “Conformations of immunoglobulin hypervariable regions”. In: *Nature* 342.6252 (Dec. 1989), pp. 877–883. DOI: [10.1038/342877a0](https://doi.org/10.1038/342877a0).
- [30] I. S. Mian, A. R. Bradwell, and A. J. Olson. “Structure, function and properties of antibody binding sites”. In: *Journal of Molecular Biology* 217.1 (Jan. 1991), pp. 133–151.
- [31] J. Foote and G. Winter. “Antibody framework residues affecting the conformation of the hypervariable loops”. In: *Journal of Molecular Biology* 224.2 (Mar. 1992), pp. 487–499.
- [32] R. Kunert, F. Rüker, and H. Katinger. “Molecular characterization of five neutralizing anti-HIV type 1 antibodies: identification of nonconventional D segments in the human monoclonal antibodies 2G12 and 2F5”. In: *AIDS research and human retroviruses* 14.13 (Sept. 1998), pp. 1115–1128.
- [33] T. Muster, F. Steindl, M. Purtscher, A. Trkola, A. Klima, G. Himmler, F. Rüker, and H. Katinger. “A conserved neutralizing epitope on gp41 of human immunodeficiency virus type 1”. In: *Journal of Virology* 67.11 (Nov. 1993), pp. 6642–6647.
- [34] Renate E. Kunert, Robert Weik, Boris Ferko, Gabriela Stiegler, and Hermann Katinger. “Anti-idiotypic antibody Ab2/3H6 mimics the epitope of the neutralizing anti-HIV-1 monoclonal antibody 2F5”. In: *AIDS (London, England)* 16.4 (Mar. 2002), pp. 667–668.
- [35] Johannes S. Gach, Heribert Quendler, Robert Weik, Hermann Katinger, and Renate Kunert. “Partial humanization and characterization of an anti-idiotypic antibody against monoclonal antibody 2F5, a potential HIV vaccine?” In: *AIDS research and human retroviruses* 23.11 (Nov. 2007), pp. 1405–1415. DOI: [10.1089/aid.2007.0089](https://doi.org/10.1089/aid.2007.0089).
- [36] A. Mader and R. Kunert. “Humanization strategies for an anti-idiotypic antibody mimicking HIV-1 gp41”. In: *Protein engineering, design & selection: PEDS* 23.12 (Dec. 2010), pp. 947–954. DOI: [10.1093/protein/gzq092](https://doi.org/10.1093/protein/gzq092).
- [37] Johannes S. Gach, Heribert Quendler, Stefanie Strobach, Hermann Katinger, and Renate Kunert. “Structural analysis and in vivo administration of an anti-idiotypic antibody against mAb 2F5”. In: *Molecular Immunology* 45.4 (Feb. 2008), pp. 1027–1034. DOI: [10.1016/j.molimm.2007.07.030](https://doi.org/10.1016/j.molimm.2007.07.030).
- [38] Renate Kunert and Alexander Mader. “Anti-idiotypic antibody Ab2/3H6 mimicking gp41: a potential HIV-1 vaccine?” In: *BMC Proceedings* 5 (Suppl 8 Nov. 22, 2011), P64. DOI: [10.1186/1753-6561-5-S8-P64](https://doi.org/10.1186/1753-6561-5-S8-P64).
- [39] Anita de Ruiter, Alexander Mader, Renate Kunert, and Chris Oostenbrink. “Molecular Simulations to Rationalize Humanized Ab2/3H6 Activity”. In: *Australian Journal of Chemistry* 64.7 (2011), pp. 900–909.
- [40] Steve Bryson, Jean-Philippe Julien, David E. Isenman, Renate Kunert, Hermann Katinger, and Emil F. Pai. “Crystal structure of the complex between the F(ab)’ fragment of the cross-neutralizing anti-HIV-1 antibody 2F5 and the F(ab) fragment of its anti-idiotypic antibody 3H6”. In: *Journal of Molecular Biology* 382.4 (Oct. 17, 2008), pp. 910–919. DOI: [10.1016/j.jmb.2008.07.057](https://doi.org/10.1016/j.jmb.2008.07.057).

- [41] Yves Durocher, Sylvie Perret, and Amine Kamen. “High-level and high-throughput recombinant protein production by transient transfection of suspension-growing human 293-EBNA1 cells”. In: *Nucleic Acids Research* 30.2 (Jan. 15, 2002). DOI: [10.1093/nar/30.2.e9](https://doi.org/10.1093/nar/30.2.e9).
- [42] Rene Tobias and Sriram Kumaraswamy. “Biomolecular binding kinetic assays on the octet platform”. In: *FortBio Application Note* 14 (2014), pp. 1–21.
- [43] Gregory M. Lee and Charles S. Craik. “Trapping Moving Targets with Small Molecules”. In: *Science* 324.5924 (Apr. 10, 2009), pp. 213–215. DOI: [10.1126/science.1169378](https://doi.org/10.1126/science.1169378).
- [44] Austin D. Vogt and Enrico Di Cera. “Conformational Selection Is a Dominant Mechanism of Ligand Binding”. In: *Biochemistry* 52.34 (Aug. 27, 2013), pp. 5723–5729. DOI: [10.1021/bi400929b](https://doi.org/10.1021/bi400929b).
- [45] MOE. *Molecular Operating Environment (MOE)*. 2014.
- [46] Markus Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: [10.1002/jcc.20303](https://doi.org/10.1002/jcc.20303).
- [47] Nathan Schmid, Clara D. Christ, Markus Christen, Andreas P. Eichenberger, and Wilfred F. van Gunsteren. “Architecture, implementation and parallelisation of the GROMOS software for biomolecular simulation”. In: *Computer Physics Communications* 183.4 (Apr. 2012), pp. 890–903. DOI: [10.1016/j.cpc.2011.12.014](https://doi.org/10.1016/j.cpc.2011.12.014).
- [48] Chris Oostenbrink, Alessandra Villa, Alan E. Mark, and Wilfred F. van Gunsteren. “A biomolecular force field based on the free enthalpy of hydration and solvation: the GROMOS force-field parameter sets 53A5 and 53A6”. In: *Journal of Computational Chemistry* 25.13 (Oct. 2004), pp. 1656–1676. DOI: [10.1002/jcc.20090](https://doi.org/10.1002/jcc.20090).
- [49] Nathan Schmid, Andreas P. Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E. Mark, and Wilfred F. van Gunsteren. “Definition and testing of the GROMOS force-field versions 54A7 and 54B7”. In: *European Biophysics Journal* 40.7 (July 1, 2011), pp. 843–856. DOI: [10.1007/s00249-011-0700-9](https://doi.org/10.1007/s00249-011-0700-9).
- [50] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81.8 (Oct. 15, 1984), pp. 3684–3690. DOI: [10.1063/1.448118](https://doi.org/10.1063/1.448118).
- [51] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J. C Berendsen. “Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes”. In: 23.3 (Mar. 1977), pp. 327–341. DOI: [10.1016/0021-9991\(77\)90098-5](https://doi.org/10.1016/0021-9991(77)90098-5).
- [52] Ilario G. Tironi, René Sperb, Paul E. Smith, and Wilfred F. van Gunsteren. “A generalized reaction field method for molecular dynamics simulations”. In: *The Journal of Chemical Physics* 102.13 (Apr. 1, 1995), pp. 5451–5459. DOI: [10.1063/1.469273](https://doi.org/10.1063/1.469273).
- [53] Tim N. Heinz, Wilfred F. van Gunsteren, and Philippe H. Hünenberger. “Comparison of four methods to compute the dielectric permittivity of liquids from molecular dynamics simulations”. In: *The Journal of Chemical Physics* 115.3 (July 15, 2001), pp. 1125–1136. DOI: [10.1063/1.1379764](https://doi.org/10.1063/1.1379764).
- [54] Xavier Daura, Wilfred F. van Gunsteren, and Alan E. Mark. “Foldingunfolding thermodynamics of a beta-heptapeptide from equilibrium simulations”. In: *Proteins* 34.3 (Feb. 15, 1999), pp. 269–280. DOI: [10.1002/\(SICI\)1097-0134\(19990215\)34:3<269::AID-PROT1>3.0.CO;2-3](https://doi.org/10.1002/(SICI)1097-0134(19990215)34:3<269::AID-PROT1>3.0.CO;2-3).

5.6 Appendix / Supplementary material

wt3H6	properties	su3H6	properties	possible function	in crystal structure
A68 ^{HC}	small, aliphatic, hydrophobic	V68 ^{HC}	aliphatic, hydrophobic	backmutated in GC3H6, part of Vernier zone	surface, facing towards CDR-H2/ β -sheet (stabilizing barrel?)
V72 ^{HC}	aliphatic, hydrophobic	A72 ^{HC}	small, aliphatic, hydrophobic	backmutated in GC3H6 and GA3H6, defining canonical CDR structure class, part of Vernier zone	surface, underneath CDR-H2
R98 ^{HC}	basic	T98 ^{HC}	polar, uncharged	backmutated in GC3H6 and GA3H6, defining canonical CDR structure class, part of Vernier zone	between CDR-H1 and CDRH3 (stabilizing Ag binding site)
V4 ^{LC}	aliphatic, hydrophobic	L4 ^{LC}	aliphatic, hydrophobic	backmutated in GC3H6, part of Vernier zone	surface, facing towards CDRL3/ β -sheet (stabilizing barrel?)
R45 ^{LC}	basic	I45 ^{LC}	aliphatic, hydrophobic	backmutated in GC3H6, buried in V _H /V _L interface, not part of Vernier zone	surface, interaction with CDR-L2?
D85 ^{LC}	acidic	Y85 ^{LC}	aromatic, hydrophobic	backmutated in GC3H6 and GA3H6, affects overall protein stability	interaction with Kappa constant region?

Table 5.2: Properties of amino acid side-chains selected for establishing wt3H6 double mutants used as a training panel (TR01-06) for MD simulations. The critical functions of these framework positions are described in Mader and Kunert, Protein Eng. Des. Sel. PEDS, 23, 947 - 954 (2010). Spatial positions in the crystal structure 3BQU are summarized.

residue	number of alleles	% of total
Arginine (R)	175	80%
Threonine (T)	10	5%
Lysine (K)	24	11%
Histidine (H)	7	3%
Alanine (A)	2	1%
Total	218	100%

Table 5.3: Conserved amino acid residues at the position equivalent to R98 in the human IGHV germline genes (IMGT/Gene-DB). Search parameters: homo sapiens (species), variable (gene type), functional (functionality), IGH (locus) and IG (molecular component).

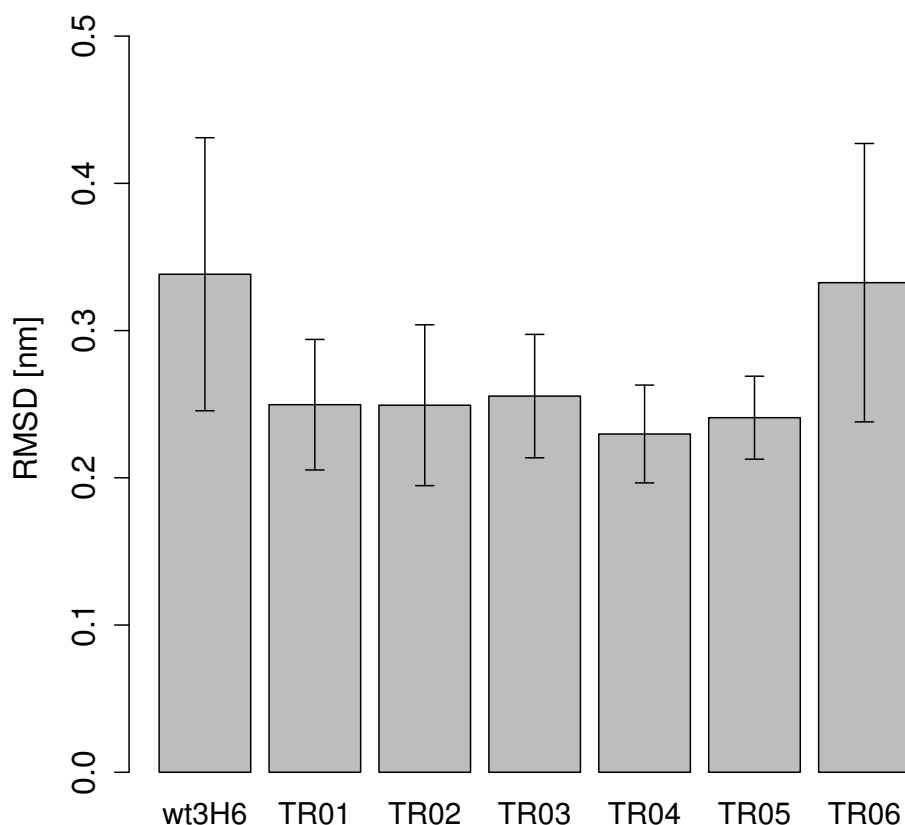


Figure 5.9: The average backbone atom root-mean-square deviation (RMSD) with respect to the crystal structure over the last 20 ns of all replicates for the training set. The RMSD is calculated for all backbone atoms of the framework regions after a least-squares fit on these atoms. The error bars indicate the standard deviation.

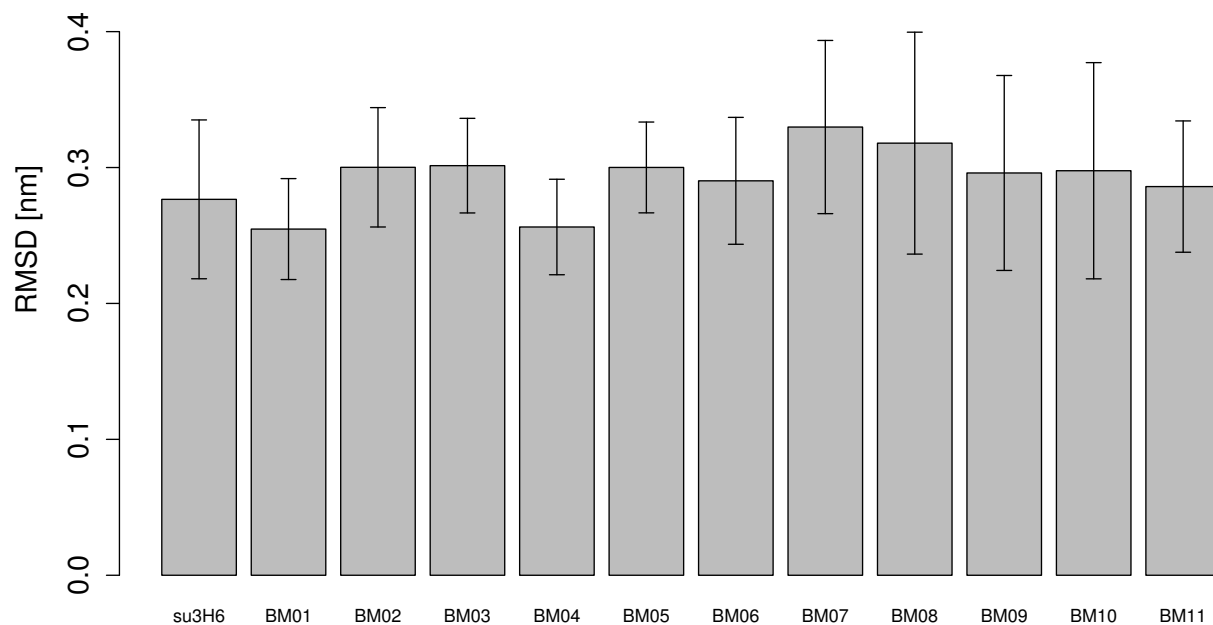


Figure 5.10: The average backbone atom root-mean-square deviation (RMSD) with respect to the crystal structure over the last 20 ns of all replicates for the superhumanized variants. The RMSD is calculated for all backbone atoms of the framework regions after a least-squares fit on these atoms. The error bars indicate the standard deviation.

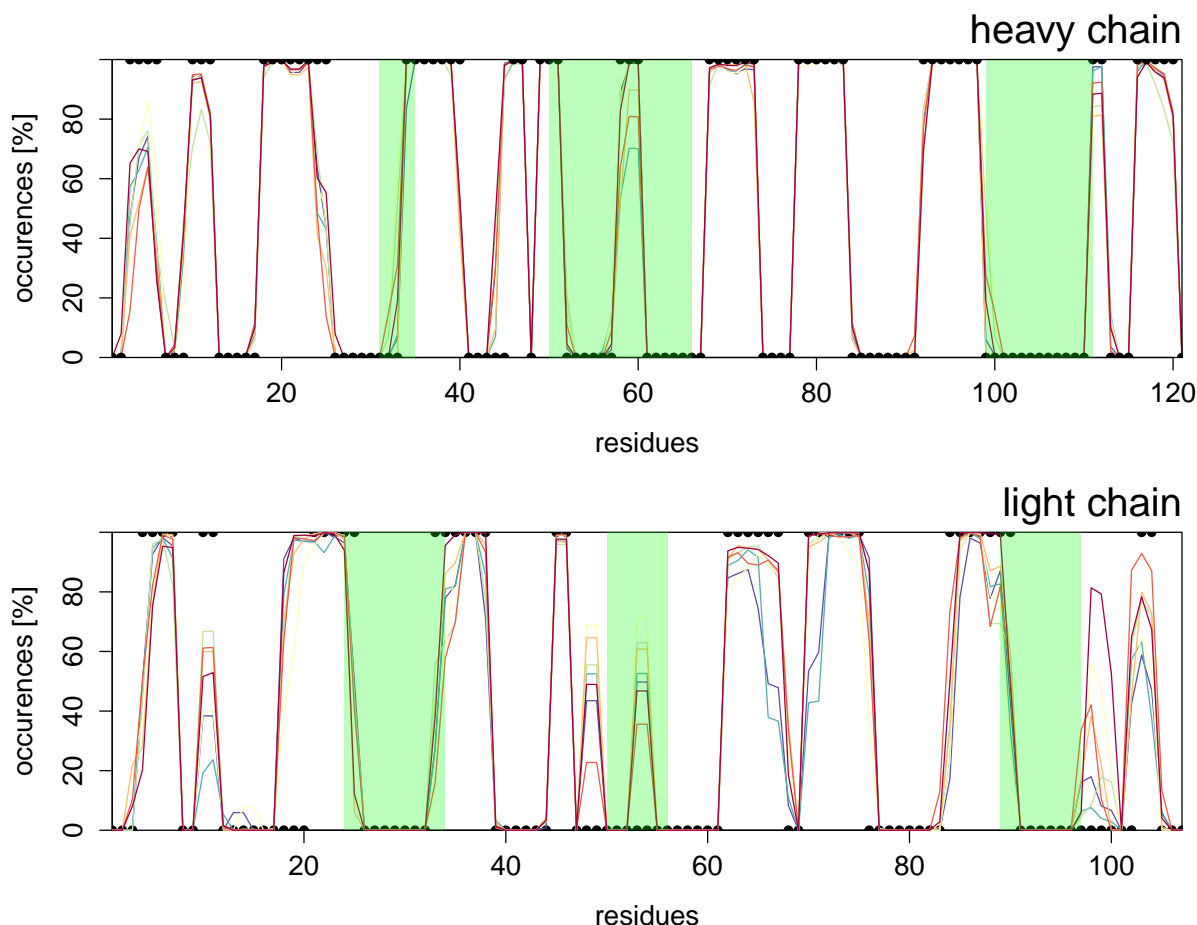


Figure 5.11: β -strand average occurrences of conformations β over the last 40 ns over all replicates for the training set as determined by the program DSSP for the framework and complementary-determining regions (CDRs; in green). The classification of the crystal structure is shown in black dots (either 100% or 0%), which agrees well with the mean values of the trajectories. This indicates a stable secondary fold throughout the simulation.

PROMETHEUS - automatization of molecular dynamics simulations

PROMETHEUS is a software package for the automated execution of molecular dynamics simulations and their subsequent analysis. It is particularly useful in cases, where (large) series of logically related simulations have to be performed, since these often contain highly repetitive steps. PROMETHEUS allows to formulate this processing logic in terms of workflow templates, which are instantiated for every member of a series automatically. The software consists of three different parts, the frontend (GUI), the backend (execution unit) and a database, to store workflow templates and facilitate a better user experience. The internal hierarchy of levels allows to structure projects, which might be accompanied by a respective folder structure, in a way that allows batch processing as well as documentation of calculations performed. In contrast to e.g. bash scripts, the framework provides tracking of the current status during execution in order to resume paused, cancelled or failed calculations from the last successfully applied step. These steps are the smallest unit of information in the workflow, implementing either program execution (internal or external mode) or process logic (iterations). The abstraction from an actual e.g. simulation to the general "simulation logic" requires the possibility to change and define key parameters, which is provided by the definition of (namespace) variables in the header of the instruction files. For convenience, the interaction with the cluster system is encapsulated and generally independent of the calculations performed. Currently, PROMETHEUS supports GROMOS file formats but may serve as a useful "common" workflow manager for other applications as well.

Glossary

- *backend*: collection of stand-alone programs, written in C++, executing part of PROMETHEUS, called by the frontend
- *frontend*: graphical user interface and execution preparation agent, written in PHP and JavaScript, has access to a MySQL database
- *metajob*: project-level organisational entity, owns a root directory, has a collection of asks and subjobs associated
- *task*: groups processes into logically similar or identical kinds (e.g. a RMSD calculation), usually has a template associated with it
- *process*: actual execution (typically of a subjob-task pair) e.g. molecule 3 (subjob) is subjected to a RMSD analysis (task)
- *subjob*: logical part of a project (e.g. a molecule), can be instantiated for a task to form a process
- *subjobgroup*: groups subjobs into sets (e.g. a set of replicates), can be used to facilitate batch processing (i.e. apply the same instantiation to all group members simultaneously)
- *templates*: a general workflow logic container, has placeholders for the instantiation (e.g. when a process is formed, the actual job directory is set), usually stored in the central database
- *instantiation*: the act of generating an instruction file for a distinct process from a suitable template
- *macros*: small functions executed by the backend (e.g. to retrieve an environment variable), typically substituting a placeholder with its return value
- *namespace*: construct (e.g. implemented in C++) to group certain parts of a source code together
- *variables*: are replaced by the `executor` binary while processing and allow e.g. global definitions, template sharing and iterations
- *instruction files*: are instantiated from a template for a certain process (usually with variable updates), processed by program `executor`, serves as workflow definition and progress monitoring container
- *workflow*: a defined sequence of actions, usually includes the meaning of "automatic" in the context of computers
- *root directory*: the top directory in a metajob, all other folders should be derived from it
- *executor*: processes and updates instruction files and the cluster notes, writes log and error files, wrapper for called programs, written in C++
- *MAMA-indexing*: a way to relatively address certain regions in antibodies, allows comparisons of different sequences and simplifies indexing of residues in certain areas

- *STAT-file*: XML-based file format for storing and addressing data
- *INS-file*: XML-based file format, stores the execution logic and the progress for binary executor
- *CCF-file*: cluster configuration file defining communication with the cluster system, XML-based

6.1 Description

The main idea behind PROMETHEUS is to enable users to reuse the process logic behind workflows commonly occurring in the context of molecular dynamics simulations. There are certain requirements that have to be met when undertaking this task. First, any automated setup needs to ensure that full control over settings remains with the user, i.e. that all settings available when doing an MD simulation "by hand" remain available. This includes, but is not limited to, input settings such as simulation parameters as the choice of integrators, the cut-off radii, the bond constraint algorithms in use, etc. as well as meta-information such as the number of packages a trajectory is to be divided into. This is further complicated by the fact, that some of these variables are encoded in setting files while others are handed over to the programs by call-parameters and some even require to be read from the console during execution.

Another critical boundary condition is that the wrapping shell does not consume too much resources in order to ensure fast simulations. Still, detailed logging and error reporting capacities are fundamental to follow the execution process of both successful and failed processes. Formats should provide a well-structured (and preferably stable) layout, without imposing too much overhead when parsed. PROMETHEUS uses XML-based formats for this reason.

Last but not least, the possibilities to accept user input should cover both novice users and the needs of experts. This flexibility is offered by PROMETHEUS by providing multiple ways to achieve certain goals, e.g. by two different, possible interfaces.

6.1.1 Applications

PROMETHEUS is extremely effective in areas, where certain tasks (like simulations) differ only in minor details (the sequence of proteins, the structure of ligands, the input folders for complex analysis setups, ...). There, it offers spared time and effort in abundance and also serves as a protocolling instance (every execution performed is stored with a time stamp in the log file) and it is therefore possible to reconstruct in detail, what has been done even years ago.

In the course of this study, PROMETHEUS has been successfully used for such diverse tasks as the simulations of antibody-fragments, other large-scale proteins and a multitude of small molecules. Moreover, comprehensive analyses such as the determination of the deviation of loop configurations and applications of a (still experimental) R package to perform large-scale predictions based on Hamiltonian reweighting. Since the structure is modular it is in principle fairly easy to extend the applicability of PROMETHEUS in the future as required.

6.1.2 Antibody simulation support

In the context of this thesis a special emphasis was placed on the use of PROMETHEUS for simulations of antibodies. Antibodies might require the incorporation of additional information compared to an "usual" protein. Especially the framework and loop region assignments can be of great use, to handle residue addressing in a relative way (using a macro, MAMA-indices can be translated to absolute residue / atom numbering) or to compare different antibodies region-wise.

MAMA-indices Usually, amino acids in antibodies are addressed in an absolute way, i.e. the residues are numbered for the heavy and the light chain. However, it might sometimes be beneficial to choose an alternative way. Antibodies are often considered to consist of sequences of alternating framework (establishing the fold) and highly dynamical loop regions (main agents of binding). The well-known Kabat classification is but one example of a manifold of definitions. This proves useful, when antibodies are compared to one another structurally and allows to search for commonalities beyond the actual sequence. Therefore, PROMETHEUS implements the relative MAMA-indexing, defining these regions by their respective sequence and length (for an example, see the ABF format definition in listing 6.13. The index consists of three parts separated by a ":" as follows:

1. the chain (usually either "H" or "L" for heavy and light)
2. a letter indicating the type of the region (either "F" or "C" for framework or complementarity determining regions) and the respective number
3. the position within the region (note, that "0" stands for the whole range of the region, not for a position)

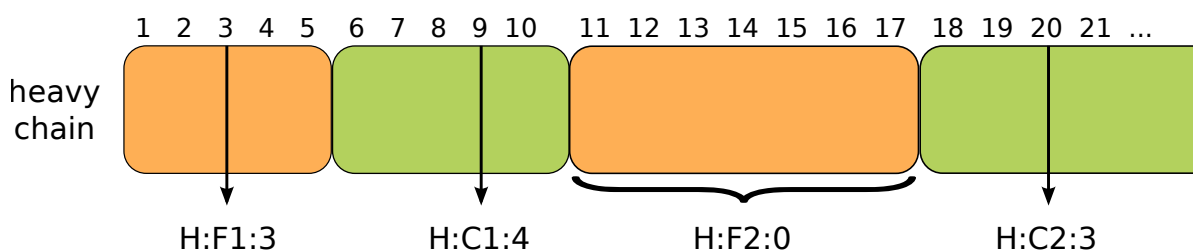


Figure 6.1: Graphical explanation of the MAMA addressing system. The framework regions are shown in orange, the loops (i.e. the complementarity-determining regions or CDRs) in green.

The advantage becomes obvious when considering a comparison between two different antibodies: once annotated, it is very easy to e.g. compare the third position in framework region two between them. Dereferencing to the absolute numbering is facilitated automatically. To use the MAMA-notation within PROMETHEUS, a step needs to have additional information on the regions: before the execution tag, the `inputMAMAfile` tag is placed: `<inputMAMAfile>antibody.abf</inputMAMAfile>`

Inside the parameters, the indexing is done as follows: `[local::abe::{H:C1:4}]` the first part is the namespace, the second loads the antibody module and the third consists of the actual MAMA-index. This whole statement will be replaced prior to execution by the absolute number (using figure 6.1 as input would lead to residue "9"). This can be mixed with other input specifications such as the GROMOS [1] atom selector: `1:res([local::abe::{H:C1:4}]:CA,N,C).`

6.1.3 Structure

PROMETHEUS consists of two independent parts, that communicate with each other: the backend and the frontend. The backend is responsible for the execution of processes and does so by parsing and executing instruction files. These files are generated by the frontend, which serves as the main user interface and does all the necessary structuring. See figure 6.2. In general, the sequence of events is as follows:

1. a metajob is created (a new project)
2. subjects are defined (to represent e.g. different molecules, proteins or boundary conditions)
3. tasks are created (e.g. "simulation", "RMSD", "RMSF", ...)
4. processes are instantiated, meaning that subjobs are used to update variables in a template to form a certain process (all information is stored in the instruction file, except the resubmitting routine)
5. the processes are submitted to a computer cluster
6. the backend executes the instruction file
7. the frontend monitors the progress as it follows the execution protocolled in the instruction file
8. the backend completes the execution and closes the log and error files
9. the status of the process is updated in the frontend (and stored in the database)
10. the user can start to examine the results (e.g. through the frontend)

To sum it up: the frontend prepares an execution and calls the backend, which internally reports how the execution proceeds by updating the respective instruction file. In addition, the frontend has access to the cluster interface to get more information on the process' current state. Moreover, this allows complete control over the processes in terms of aborting and restarting them.

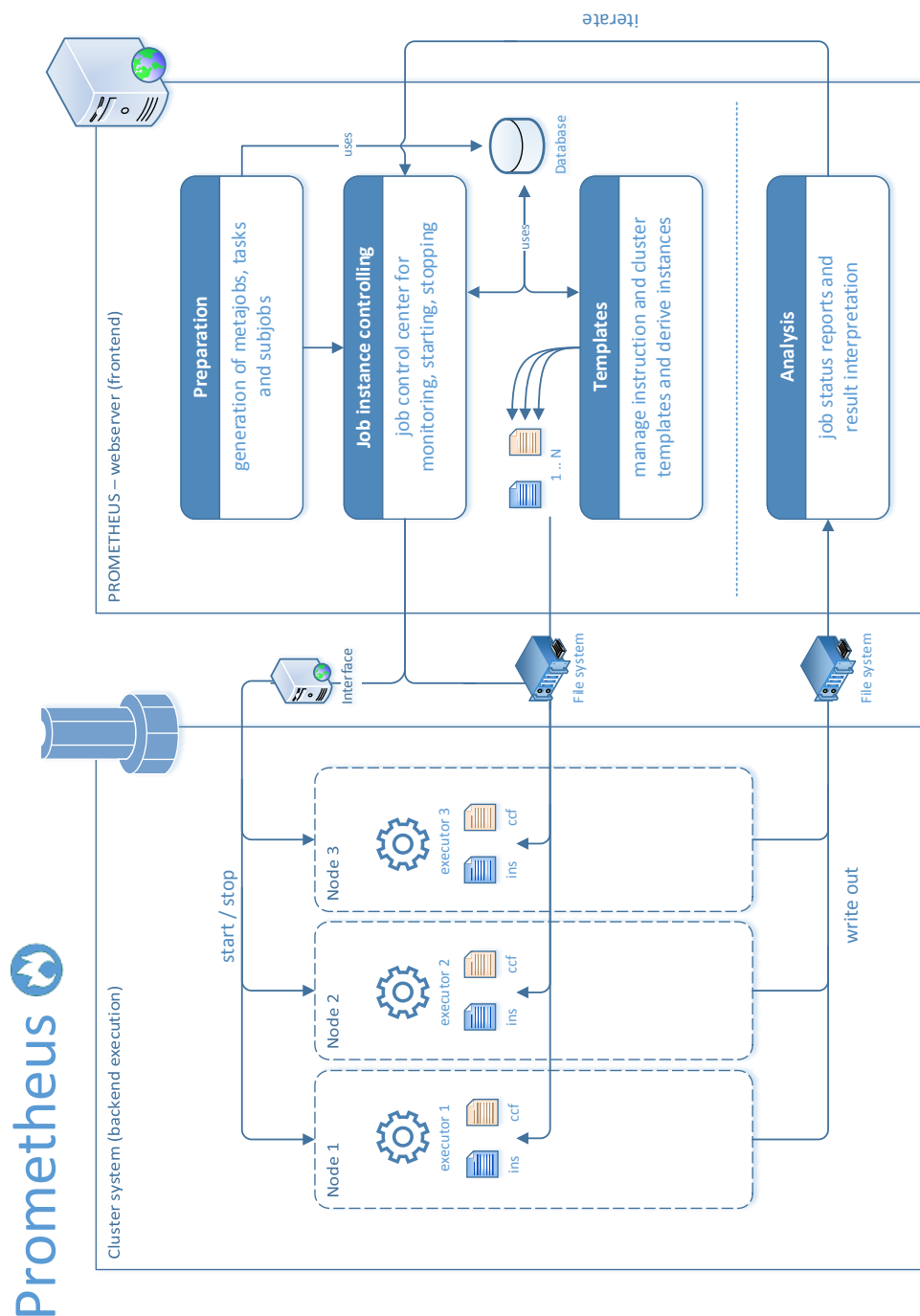


Figure 6.2: PROMETHEUS consists of two distinct parts. The first is the frontend, serving as the organisational unit and the graphical user interface. It is based on a central database and also facilitates the necessary cluster communication. The backend, the other part, represents the executing unit. The central entity is the executor, which processes the respective instruction files. A set of other specialised binaries completes the backend (described in section 6.2.1).

6.1.4 Level of control

PROMETHEUS implements multiple ways to communicate with underlying executables. To call a program, three different ways are available: a `synchronous` call (which will block any further processing till completion, see listing 6.19), an `asynchronous` call (which will be uncoupled from the parent process [not yet implemented]) and finally a `backtick` call (which collects all output returned by the child process, see listing 6.20). Some programs require multiple parameters when being called which can be used to dynamically change their behaviour by usage of global, local and parental variable replacement and the execution of macros (see listings 6.16, 6.22 and 6.23). A vivid example of the latter is the expansion of a trajectory string.

Other settings, i.e. the simulation configuration file in the case of a production simulation, can be provided by templates that are associated with a certain kind of metajob and placed in a defined subfolder. There are multiple internal functions that allow the manipulation of these text file based variables, e.g. the number of solvent molecules used in a simulation (see listing 6.18 for an example). In general, everything that can be adjusted by the user manually, can be set as well.

6.1.5 Used third-party code

The following programs / libraries have been used for the implementation of either the frontend or backend of PROMETHEUS:

C++ [2], the STLplus library [3], JavaScript [4], jQuery [5], jQueryContextMenu [6], jqueryFileTree [7], PHP [8], phpseclib [9], MySQL [10], CodeMirror [11], CodeMirrorUI [12], pureform [13], ezcomponents [14], alertify [15], colorbox [16], tinymce [17], tinymcepath [18], libssh [19] and pcre-8.3 [20].

6.1.6 Workflow and video

The first step in setting up a new simulation type, is to generate a new template folder in the PROMETHEUS root directory (see figure 6.3). In this directory, all additional files are stored which belong to the respective template (an optional feature). Not all templates might require the provision of a certain folder structure and input files (such as argument files, etc. for a simulation). The folders' name can be chosen freely and is stored in field `TemplateClonedirectory` of database table `TEMPLATES` (see listing 6.9). If a directory is specified, it will be cloned into the process directory upon instantiation.

In conjunction with the (optional) template folder, an appropriate template should be constructed (e.g. in the template editor, see figure 6.11 or the text editor, see figure 6.16), specifying the steps necessary to do the simulation (prepare topology, generate coordinates in the GROMOS format from the PDB or ABF file, energy minimization of the structure, etc.). Usually, the header of a template contains certain predefined variables (such as `jobdirectory`, which is the actual process' directory, i.e. where the execution is performed and where the instruction file is expected), whose values are replaced for every process upon instantiation (see format definition of the instruction file in listings 6.16 - 6.24).

From that point on, a new simulation can be applied by creating a new subjob and process in the web-based GUI, uploading the respective PDB file and a click on "the

rocket symbol”. PROMETHEUS will now instantiate the appropriate template files for this subjob, create the appropriate instruction file and execute the tasks through the backend on the cluster(s). The progress is shown by a bar (figure 6.12), giving the absolute numbers of the completed and total steps and a relative value. The jobs’ status (queueing, running, failed, ...) can be updated by a click on ”the gauge symbol”.

A video-based description is available at

- <http://disicl.boku.ac.at/videos/PROMETHEUS.mp4> or
- <http://www.mareal.at/videos/PROMETHEUS.mp4>,

respectively. The author would like to thank Patrick Zahrl and Noor Fakhari for their support in this matter.

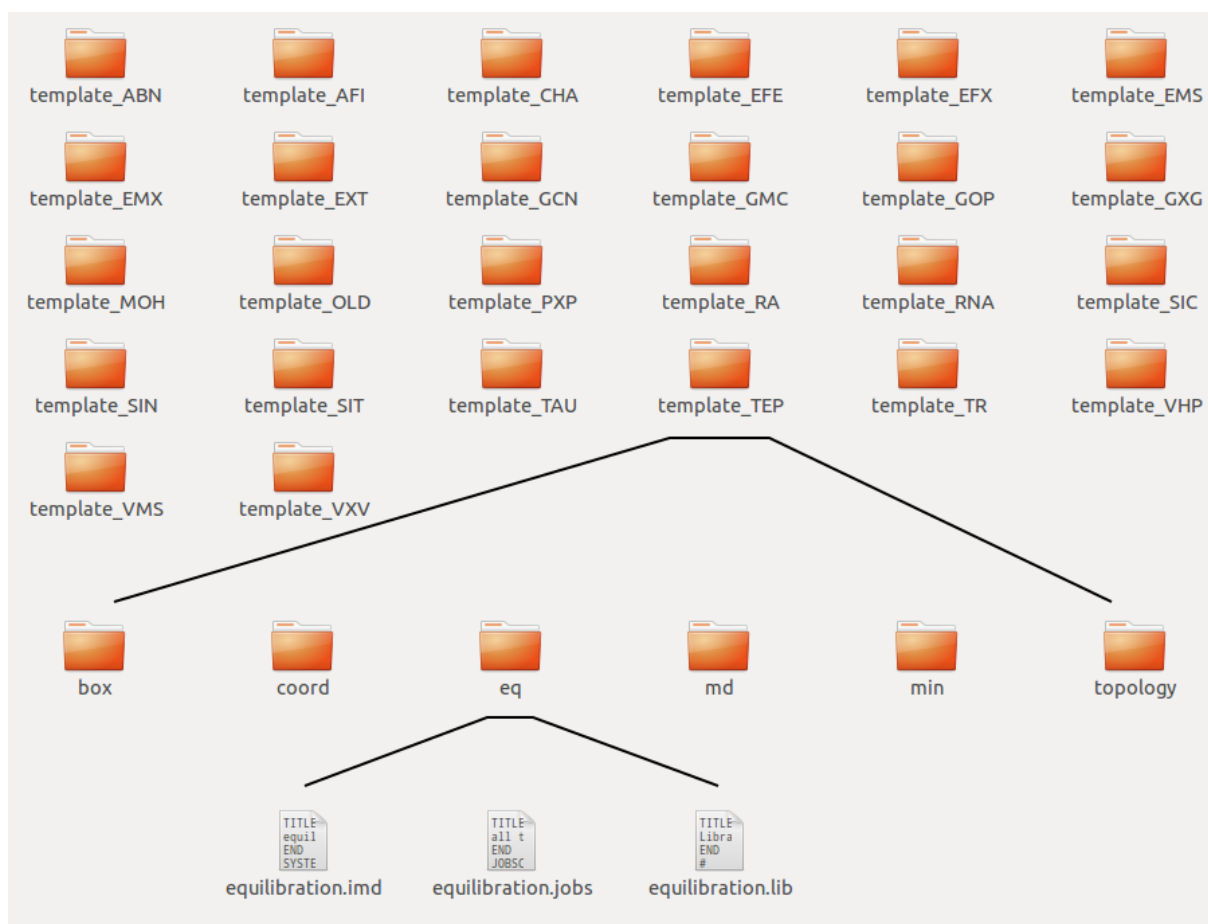


Figure 6.3: Shows the root template directory at the top, one example folder (”template_TEP”) in the middle and the content of the equilibration folder at the bottom. Of course, the content of each folder is arbitrary: all of it will be copied to the jobdirectory upon instantiation. The folder and file permissions necessary will be set appropriately for the user PROMETHEUS is running as. The values of the variables in the input files can be overwritten by the appropriate functions during the course of the execution, see section 6.2.2.

6.2 Backend

The backend is a C++ based framework, that implements a toolbox of programs, with the prominent example of the `executor` binary. This program is the execution entity, that parses and performs the steps encoded in instruction files, e.g. `executor.ins` files. The other programs might be used by the user directly or be part of the execution (e.g. the translation of a PDB file into an ABF file). The only cluster-related information stored by the backend by default, is the process identifier, but the `preparation` steps allow to access every environment variable in principle. In general, the backend executables can be called locally as well, given that a proper environment has been set. For simplicity reasons, all required libraries are linked statically during compilation, enhancing the binaries' "stand-alone" nature.

6.2.1 Standalone programs

This section holds a (incomplete) list of the most important programs provided by the backend together with a short description. To see a compilation of their parameters, see table 6.1.

abf2pdb In order to translate a ABF file into a standard PDB file, this program uses the two respective file streams and the `protein` class. Do note that disulphide bridges are not automatically exported, while chain identifiers are maintained. See format description in listing 6.13 for further details.

check_abf Validates format requirements of an ABF file in order to ensure proper parsing. Besides XML sanity checks, a test-parsing is performed and possible exceptions are logged.

check_ins Validates format requirements of an instruction file in order to ensure proper parsing. Note, that the values of parameters, especially the variable availability and macro sanity is not investigated in the current version.

execute_ccf Execution engine of the cluster configuration files (CCF), which can be used to facilitate the cluster communication. In principle, it resembles the `executor` binary, but has only access to a limited set of steps.

executor Main backend program and execution engine for regular instruction files. Apart from calling external programs, it has access to a variety of internal functions. The instruction file is parsed and executed from top to bottom, step by step. Flow control is facilitated using the respective `done` parameters, which are set to be "true" in case a step has been successfully performed. This binary also writes a log and error file, respectively. Note, that the standard output and standard error streams of child processes are also captured and integrated in these files. As successful return value from subordinate processes, a zero is expected.

merge_plain Program to combine multiple plain text files into one. The number of files accepted as input is arbitrary. A common way to use this program is to combine plain text files prior to a translation to a STAT file for further processing.

plain2stat Converts a plain text file in CSV format into a STAT file. The delimiter for parsing and the table names can be set directly by the respective parameters.

renumber_ins Renumbers an instruction file from top to bottom, which is especially useful in case steps in the middle have been deleted. A proper file in the sense of PROMETHEUS is not required, just a sane XML layout.

stat2plain Converts a STAT file into a plain text file. Table and column names are not maintained. Implements the reverse functionality to program `plain2stat`.

stat binaries In order to allow internal data processing the XML-based STAT format has been derived. There are several binaries related to the analysis and manipulation of these files (`calc_diff_stat`, `cluster_stat`, `math_stat`, `parse_stat` and `process_stat`) that mainly store analysis results. The main advantage compared to plain CSV files is the XML-based addressing by XPATH queries, inclusion of meta-information such as the datatype and labels. For further development, a native R implementation is planned in order to allow a more standardized and more powerful handling of data and direct embedding of R code into the instruction files.

pdb2abf This program converts a standard PDB file into an ABF file and is also used to transfer a certain configuration between similar or identical antibodies. Additional information can be stored in case antibodies are the target molecules. To set up antibody region definitions by hand (or to manipulate chains), an user interface is provided by the program, as shown in figure [6.4](#).

```

The name of the antibody: testAntibody

Chain: L
GLU THR THR LEU THR GLN SER PRO ALA PHE | (1 - 10)
MET SER ALA THR PRO GLY ASP LYS VAL ASN | (11 - 20)
ILE SER CYS ILE THR SER THR ASP ILE ASP | (21 - 30)
ASP ASP MET ASN TRP TYR GLN GLN LYS PRO | (31 - 40)
GLY GLU ALA ALA ILE PHE ILE ILE GLN ASP | (41 - 50)
GLY ASN THR LEU ARG PRO GLY ILE PRO PRO | (51 - 60)
ARG PHE SER GLY SER GLY TYR GLY THR ASP | (61 - 70)
PHE THR LEU THR ILE ASN ASN ILE GLU SER | (71 - 80)
GLU ASP ALA ALA TYR TYR PHE CYS LEU GLN

Chain: H
GLY VAL GLN LEU GLN GLN SER GLY PRO GLU | (1 - 10)
LEU VAL LYS THR GLY ALA SER VAL LYS ILE | (11 - 20)
SER CYS LYS ALA SER GLY TYR SER PHE THR | (21 - 30)
ASP TYR PHE MET HIS TRP VAL LYS GLN SER | (31 - 40)
HIS GLY LYS SER LEU ASP TRP ILE GLY TYR | (41 - 50)
ILE ASN CYS TYR THR GLY ALA THR ASN TYR | (51 - 60)
SER GLN LYS PHE LYS GLY ARG VAL THR ILE | (61 - 70)
THR ALA ASP THR SER

Disulphides (1)
[1]: L:23 - L:88

Ranges (2)
[1]: L:F1:23
[2]: L:C1:12

-----
[D] ... add disulphide      [A] ... add antibody range
[R] ... remove disulphide   [B] ... remove antibody range
[U] ... cut chain           [K] ... remove chain
[M] ... rename chain        [N] ... change antibody name
[C] ... check antibody      [E] ... end (and save)

Press key: 

```

Figure 6.4: Screenshot of the user interface available for `pdb2abf`, which allows to set antibody-specific settings. Apart from disulphide bridges, MAMA-ranges can also be set which allow a relative addressing of the respective antibody regions. Other convenience functions allow to truncate, rename and remove protein chains.

Table 6.1: Table of backend programs, together with their parameter lists and short descriptions.

name	flags	description
abf2pdb	@input	input ABF file
	@output	output PDB file
	@info	ignore other flags, print usage description
calc_diff_stat	@input	input analysis file
	@select	specification as XPATH expression (table)
	@reference	specification as XPATH expression (reference column within table)
	@output	output analysis file [optional]
	@noappend	generated normalized columns will be saved exclusively [optional]
	@difftype	specify calculation type [optional]
	@info	ignore other flags, print usage description
check_abf	@input	input ABF file
	@info	ignore other flags, print usage description
check_ccf	@input	input CCF file
	@output	output CCF file [optional]
	@html	enables result print-out to be in HTML format [optional]
	@xml	enables result print-out to be in XML format [optional]
	@info	ignore other flags, print usage description
check_ins	@input	input INS file
	@output	output INS file [optional]
	@html	enables result print-out to be in HTML format [optional]
	@xml	enables result print-out to be in XML format [optional]
check_stat	@input	input analysis file
	@info	ignore other flags, print usage description
cluster_stat	@input	input analysis file
	@output	output analysis file [optional]
	@numberclusters	specifies, in how many clusters the dataset has to be divided
	@select	specification as XPATH expression
	@noappend	will cause the generated clusters to be exclusively saved [optional]
	@overwrite	enforces overwriting [optional]

name	flags	description
execute_ccf	@input_ccf	input CCF file, which is read in
	@input_ins	input INS file, which is related
	@execute	specifies, which block shall be executed (name)
	@xml	enforces output in XML format [optional]
	@info	ignore other flags, print usage description
executor	<i>none</i>	<i>none</i>
math_stat	@input	input analysis file
	@select	specification as XPATH expression
	@operation	specify operation required
	@output	output analysis file [optional]
	@noappend	generated data will be saved exclusively in the new file [optional]
	@overwrite	force overwriting [optional]
merge_plain	@input	input plain file(s)
	@output	output plain file
	@info	ignore other flags, print usage description
merge_stat	@input	input analysis file(s)
	@output	output analysis file
	@select	specification as XPATH expression
	@overwrite	enforces overwriting [optional]
	@info	ignore other flags, print usage description
	@unite	tries to combine all nodes into one table [optional]
normalize_stat	@input	input analysis file
	@select	specification as XPATH expression
	@output	output analysis file [optional]
	@noappend	generated normalized columns will be saved exclusively [optional]
	@overwrite	enforces overwriting [optional]
parse_stat	@input	input analysis file(s)
	@output	output analysis file
	@select	specification as XPATH expression
	@outputformat	set file format [optional]
	@overwrite	enforces overwriting [optional]
	@maintain	enforces original structure [optional]
info_stat	@input	input analysis file
	@select	specification as XPATH expression
	@show	specifies, what information is actually printed [optional]

name	flags	description
pdb2abf	@input	input PDB file
	@output	output ABF file
	@annotation	input ABF file for range annotation and disulphides [optional]
	@nointerface	prevents interface from being opened for user input [optional]
	@setchain	sets chain ID to specified value for all atoms [optional]
plain2stat	@input	input plain text file
	@output	output analysis file
	@name-table	setting names of all tables in the vector [optional]
	@table-delimiter	where to split data
plot_stat	@input	input analysis file
	@output	output graphical file
	@graphtype	specifies type of graph
	@select	specification as XPATH expression [optional]
	@graphformat	specifies the outputformat [optional]
	@info	ignore other flags, print usage description
process_stat	@input	input analysis file
	@output	output analysis file
	@process	process command
	@outdatatype	type of column data for output [optional]
renumber_ins	@input	input INS file, in which steps are renumbered
	@info	ignore other flags, print usage description
reset_ins	@input	input INS file, in which steps are resetted
	@from	indicates from which step to start the resetting (inclusive) [optional]
	@info	ignore other flags, print usage description
stat2plain	@input	input analysis file
	@output	output plain file
	@info	ignore other flags, print usage description
status_ins	@input	input INS file, which is read in
	@xml	enforces output in XML format [optional]
	@info	ignore other flags, print usage description

6.2.2 Support and internal functions

Table 6.2: List of support functions used in the backend programs. Some are of general use (namespace: supfunc), others implement functionality when the GROMOS simulation package is used (namespace: GROMOS). These functions are available from within the programs, not by calling them (see internal functions for comparison).

name	parameters	description
namespace: supfunc		
matches_selection()	string& pattern string& input return bool	Returns TRUE, if pattern is produced in input.
get_temporary_copy()	const file& originalFile const file& duplicateFile return bool	Generates a temporary duplicate of a file.
replace_all_copy()	string theString const string& pattern const string& replacevalue return string	Replaces all occurrences of the pattern by the specified replacement.
travelDirRecursively()	const string& theDir vector<string>& theStorage	Lists all files attached or in a subdirectory.
parseOptions()	int numberArg char* arrArg[] return map<string, string>	Transfers parameters into usable form.
getEnvironment()	return map<string, string>	Retrieves complete set of environmental variables.
namespace: GROMOS		
get_number_ions()	int & numberNatriums int & numberChlorides GBlock& theBlock	Calculates ions required to neutralize box charge.
calculate_total_charge()	GBlock& topoSolute return int	Calculates box net charge.
generate_trajectory_str.()	int & numberOfTraj. string& templateString return string	Generates path for trajectory.
get_last_numbers()	... vecNumbers ioCNFfile& inputcnfStream ioTOPfile& topologyStream int & solventMolecules return bool	Retrieves last non-solvents atoms.

Table 6.3: Internal functions provided by the backend they offer general tasks, such as the evaluation of a regular expression. For an example, how to call them, see the format definition in listing 6.18.

name	parameters	description
<code>regexexpression()</code>	<pre> "input" "regexextract" "regmatch" return "true" "false" </pre>	Evaluates regular expressions and returns a string accordingly.
<code>execute_ccf_step()</code>	<pre> "clusterconfigurationfile" "stepkey" </pre>	Executes a step defined in a CCF file.
<code>update_INS_variable()</code>	<pre> "input" "variablekey" "instructionfile" </pre>	Replaces a header variables' value in an instruction file.
<code>split_into_PDB_files()</code>	<pre> "inputabf" "template" </pre>	Splits an ABF file into one PDB file (per chain identifier).
<code>move()</code>	<pre> "original1 target1" "original2 target2" "... ..." </pre>	Batch moves an arbitrary number of files.
<code>rename_files()</code>	<pre> "directory" "searchpattern" "replaceby" </pre>	Renames an arbitrary number of files, by replacing a search pattern.
<code>execute_in_directory()</code>	<pre> "chdirpath" "command" "returndirpath" </pre>	Changes process to specified directory string and executes a command.

Table 6.4: List of internal GROMOS functions provided by the program executor. These functions add necessary parts in the execution of a GROMOS based simulation and enhance the convenience significantly.

name	parameters	description
<code>delete....POR_file()</code>	<code>"porfile"</code>	Deletes atoms with zero coordinates in space.
<code>reset....CNF_file()</code>	<code>"cnffile"</code> <code>"iterativeshift"</code>	Sets atoms in origin to be close to an anchor atom to prepare minimization.
<code>md()</code>	<code>"binary"</code> <code>"directory"</code> <code>"topology"</code> <code>"initialcoordinatefile"</code> <code>"previouscoordinatefile"</code> <code>"inputimdfile"</code> <code>"rprfile"</code> <code>"porfile"</code> <code>"tempcoordinatefile"</code> <code>"tempenergyfile"</code> <code>"temptrajectoryfile"</code> <code>"outputfile"</code> <code>"exportvariables"</code> <code>"currentiteration"</code>	Executes GROMOS molecular mechanics engine. The RPR and POR files are optional, the paths of the temporary files can be set to a local destination. The string of export variables specifies which environmental variables should be passed. The currentiteration parameter specifies the progress (usually combined with an iteration).
<code>set_NTIRTC_to_one()</code>	<code>"inputimdfile"</code>	Sets IMD parameter NTIRTC to one in the specified file.
<code>prepare_ion_input_file()</code>	<code>"argumentfile"</code> <code>"topology"</code> <code>"coordinatefile"</code> <code>"maxnatriums"</code> <code>"maxchlorides"</code>	Sets necessary options before the call of GROMOS++ program ion.
<code>generate_RPR_file()</code>	<code>"coordinatefile"</code> <code>"rprfile"</code>	Converts coordinate file into RPR (reference positions).
<code>generate_POR_file()</code>	<code>"coordinatefile"</code> <code>"solutetopology"</code> <code>"porfile"</code>	Converts coordinate file into POR (list of atoms to be restrained positionally).

name	parameters	description
<code>update_IMD_file()</code>	<pre>"imdinputfile" "coordinatefile" "solutetopology"</pre>	Updates critical settings in the specified IMD file, including the number of solvent molecules and the numbers of the last atoms.
<code>set_..._oxygens_mass()</code>	<pre>"mass" "topology"</pre>	Sets the mass of the last two oxygen atoms in a topology to the value of a proton (1.00800 for GROMOS).
<code>prepare_com_top_ion()</code>	<pre>"topology" "natriumtopology" "chloridetopology" "maxnatriums" "maxchlorides" "templateargfile" "outputfile"</pre>	Prepares a argument file for GROMOS program <code>com_top</code> , including the proper ion numbers.
<code>prepare_make_top()</code>	<pre>"inputabf" "templateargfile" "Nterminus" "Cterminus" "argfileoutputdirectory"</pre>	Prepares argument file for the GROMOS program <code>make_top</code> , including the definition of disulphide bridges.

6.2.3 Macros

The backend macros are small functions executed by `executor` prior to the "real" execution of the step. Instead of the macro given in the instruction file, the parameter values is usually replaced by the macro's return value. Every function begins with `(func=`, followed by a number, the macro's identifier and the `keyFunction` with its value, separated by the namespace operator `::`. Therefore, every function is opened by: `(func=#::macroname::keyFunction=value)`. The macro needs to be closed by `(func(#)::end)`. In between, a "spot" can be specified which is replaced by the macros' return value: `(func(#)::spot::specifier)`. Some macros have multiple specifiers. Spots can be used multiple times and can be between a static string, common variables or even other macros. Currently, four macros are implemented, which will be described in detail in this section.

expand Allows to generate a string by either looping over numbers or a subjobgroup. This macro is very useful to e.g. generate trajectory strings or to combine a group of subjobs in one analysis (for example clustering). For a description, how to loop over a subjobgroup one by one, see the subjob-iterations in listing 6.24.

If:

- `keyFunction` equals "numbers": `spot` can be either "currentnumber" (the current iteration index) or "previousnumber" (the previous iteration index)
- `keyFunction` equals "subjobs": `spot` can be either "subjobidentifier" (the string identification as given in the subjobgroup in the header) or "subjobdirectory" (the folder given in the header)

Example:

```
(func=1::expand::numbers=1-[global::maxsimpackages])
md/md_(func(1)::spot::currentnumber).trc.gz (func(1)::end)
```

```
1 "md/md_1.trc.gz md/md_2.trc.gz md/md_3.trc.gz
2 md/md_4.trc.gz md/md_5.trc.gz md/md_6.trc.gz
3 md/md_7.trc.gz md/md_8.trc.gz md/md_9.trc.gz ..."
```

execute Executes a command prior to evaluation and execution of the step itself. The process is paused until the execution is completed. Sometimes, it might be convenient to execute a command (e.g. a system program as in the example) and hand over its output as a parameter.

If:

- `keyFunction` is the command: `spot` is "return" and holds the output gathered from the called command

Example:

```
(func=2::execute::ls)the folders' content:
(func(2)::spot::return) (func(2)::end)
```

```
1 "      one_protein.top  all1.top      class.txt
2          three.mtb   three.top  other_file.cab
3 terminal_parameters.mtb 4carbo.top      three.xls ..."
```

select Takes a number of strings and returns those, that match a criterion. Currently, only filenames are supported. As shown in the example, wild-cards are supported.

If:

- `keyFunction` equals "files": `spot` is either "path" (the file paths) or "name" (the file names)

Example:

```
(func=1::select::files=*.top)
(func(1)::spot::path)/(func(1)::spot::name) (func(1)::end)
```

```
1  "/home/user/one_protein.top      /home/user/all1.top
2  /home/user/three.top           /home/user/4carbo.top ..."
```

environment Retrieves environmental variables and their respective values. The evaluation of the environment is necessary (e.g. when using LUCI [21]) to properly use some programs. Parts or all of the environment can be accessed.

If:

- `keyFunction` equals "selector": `spot` can be either "variable" (the variable name), "value" (the variables' value) or "all" (the whole list of variable-value pairs while ignoring the value of `keyFunction`)

Example:

```
(func=1::environment::selector=JOBID)
(func(1)::spot::value) (func(1)::end)
```

6.2.4 Classes

In this section, the most important classes used in the implementation of the backend are shown in UML class diagrams, including their relationships. In general, the input / output (IO) has been implemented by stream classes, that facilitate the loading from / writing to files as well as the parsing of the content into data structures (extraction) and the transformation into writeable from (injection). The IO classes are shown in figures 6.5 (general IO), 6.6 (XML-based IO) and 6.7 (GROMOS-type IO), respectively. The latter loads the GROMOS specific data into appropriate data structures (figure 6.8). And last, the internal protein / antibody handling is shown in figure 6.9.

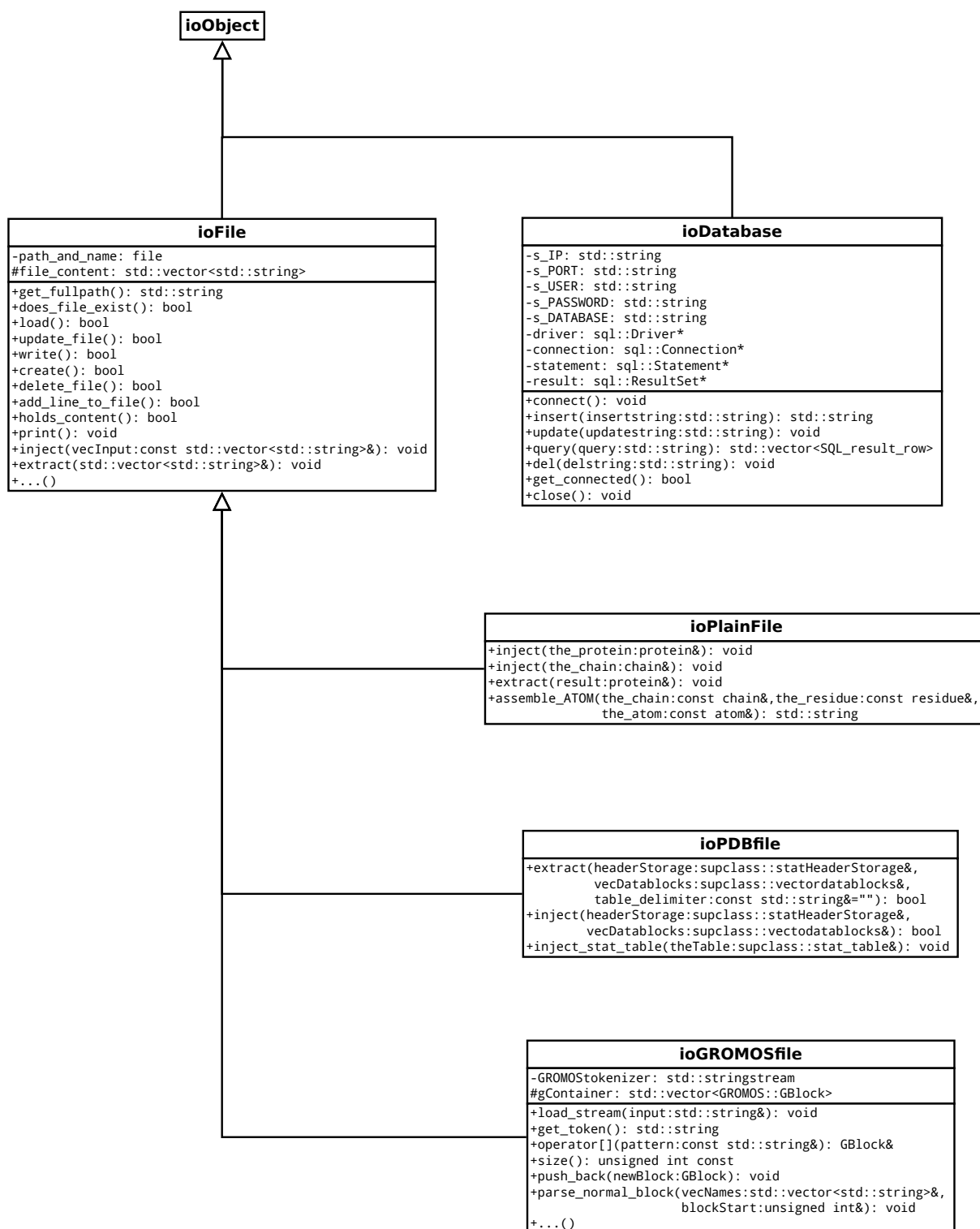


Figure 6.5: UML 2.0 diagram of the top input / output (IO) classes (see also figure 6.6). The database communication from the backend-side has been implemented but is currently not in use.

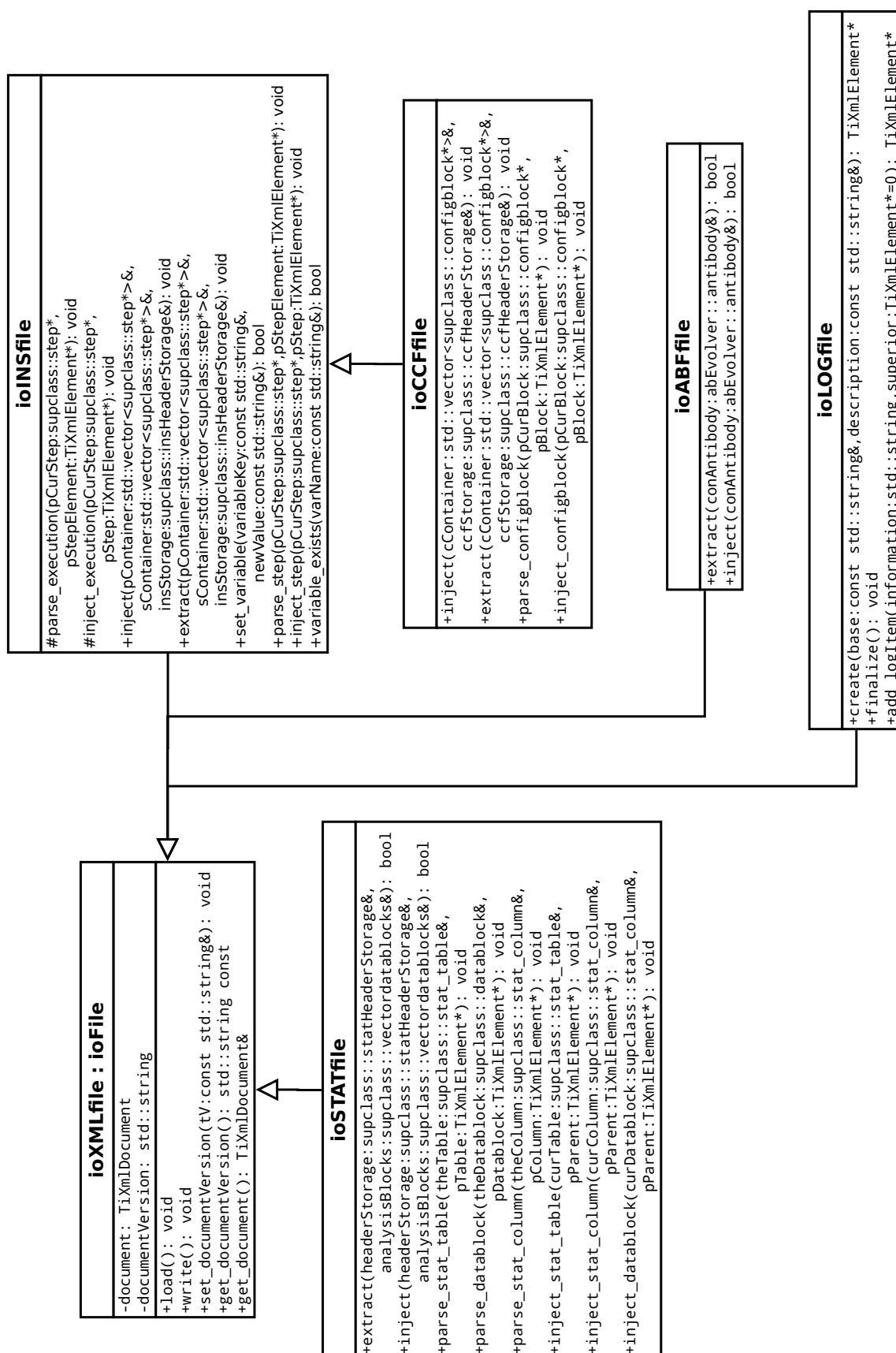


Figure 6.6: UML 2.0 diagram of the top IO classes, using XML formatting (see also figure 6.5). All file formats defined by PROMETHEUS use a XML formatting. Examples and explanations for the respective formats, can be found in section 6.4. Note, that the ioLOGfile class is also used for the error file. All classes implement a main parsing (extract()), a XML-constructing (inject()) and several minor parsing functions.

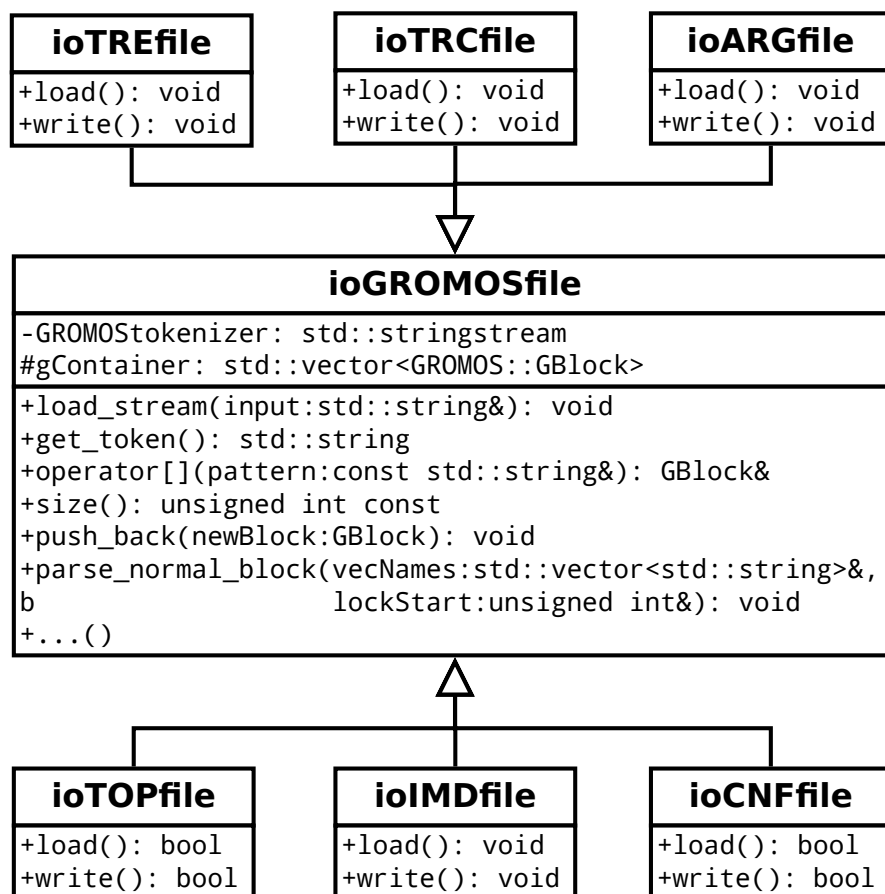


Figure 6.7: UML 2.0 diagram of the IO classes, which load, write and parse GROMOS file formats. There are multiple convenience functions available to access a certain data field easily.

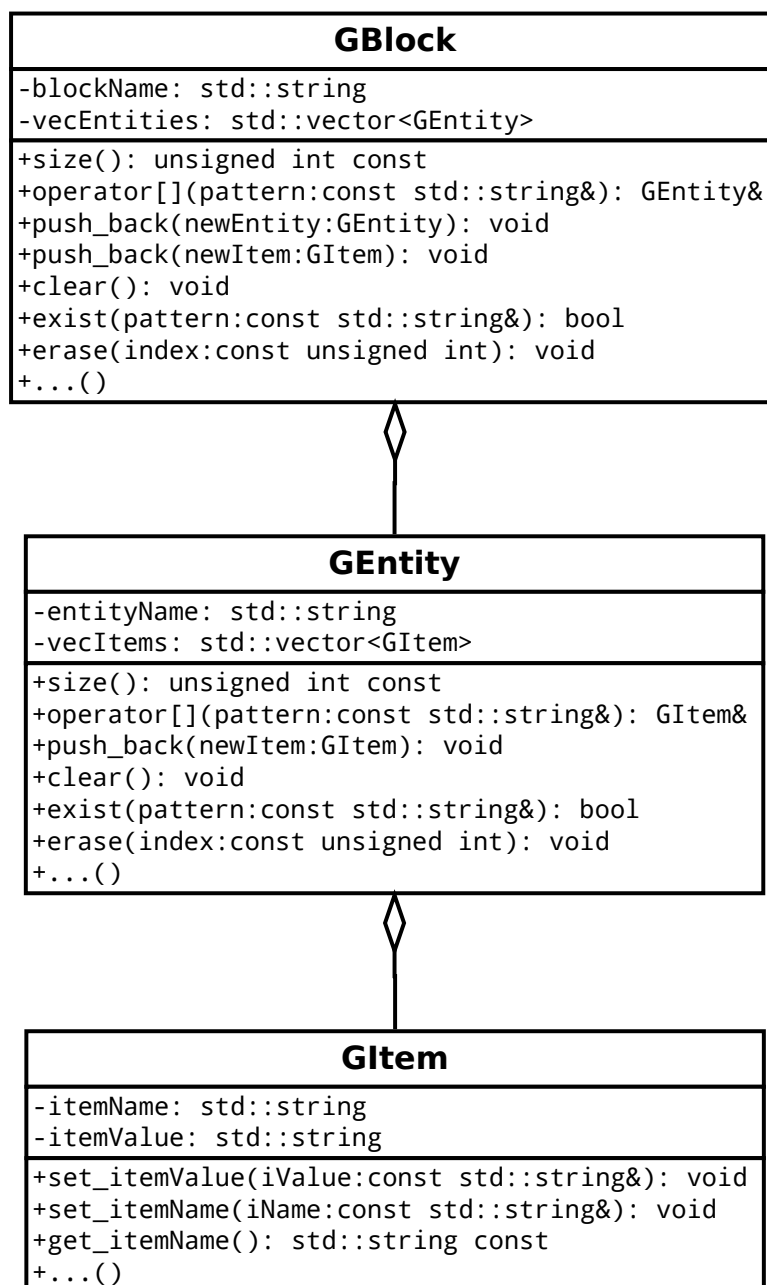


Figure 6.8: UML 2.0 diagram of the GROMOS parsing classes. Every block consists of entities, that in term hold the actual items (the variable names and associated values). The order and number of expected values is hard-coded in special header files.

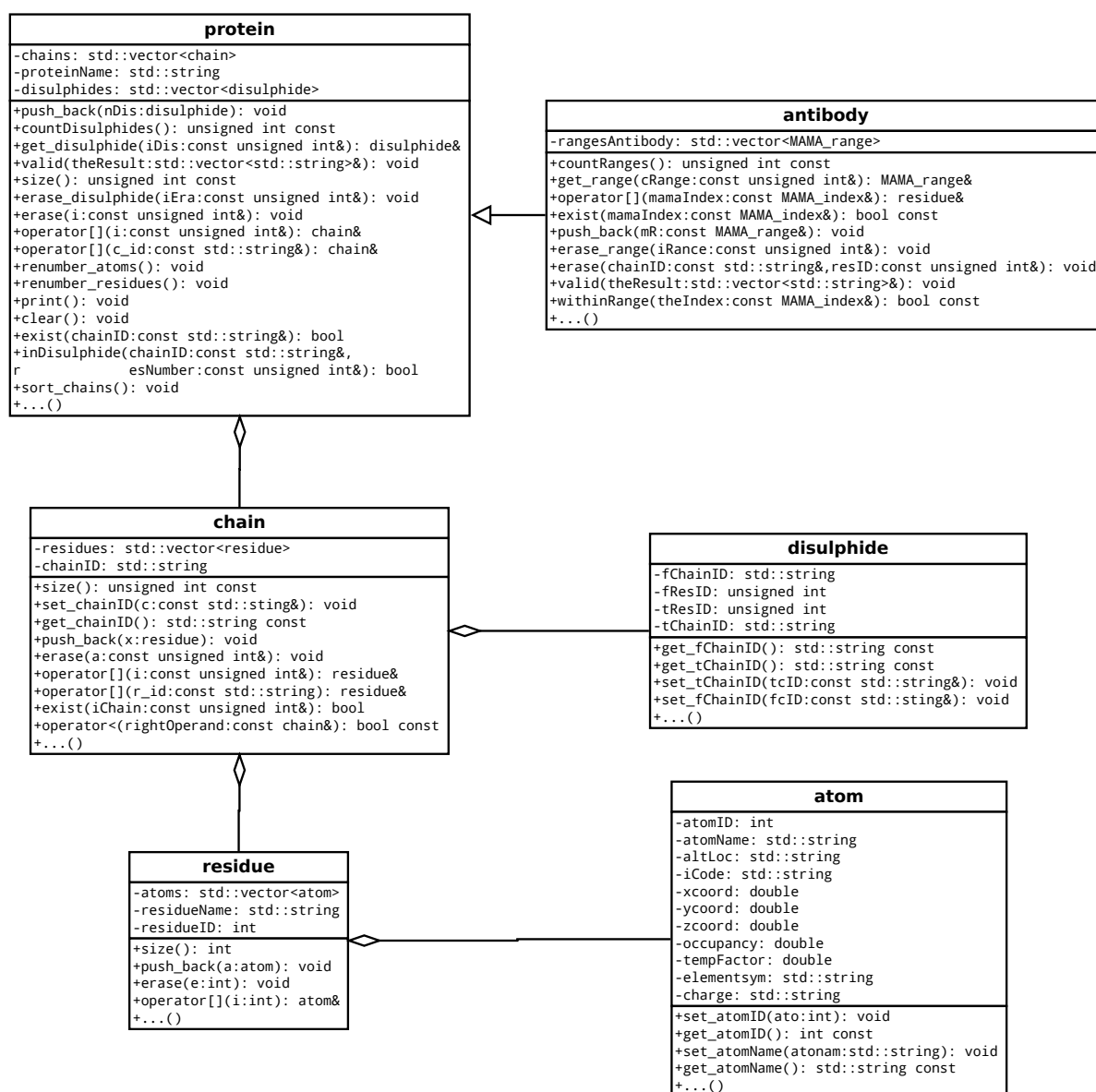


Figure 6.9: UML 2.0 diagram of the internal protein and antibody cluster. Proteins consist of chains, consisting of residues, consisting of atoms. In addition, disulphide bridges can be specified, if the protein happens to be an antibody, its native class (inheriting from the protein) should be used, which offers additional information storage required in this case. For both the protein and the antibody class, PDB and ABF file streams are available.

6.3 Frontend

The frontend has three main purposes: i) structuring large series of molecular dynamics related processes, ii) work as an user interface to allow simple application of complex protocols including batch processing and iii) to present data and processing logic in a comprehensible way.

The structuring is mainly provided by the internal structure itself. A metajob represents a project, has a set of tasks and so on. This is also the way the interface works and the database tables are defined. Although sometimes a matter of taste, it is usually straightforward to model a project in a way such, that it resembles this structure. Defaulted structure with one root directory at the metajob level and derived subfolders also allows to implement a restrictive file permission handling: only the owner can access the files, although from the operating systems' point of view all files belong to the same user (the one, under which PROMETHEUS is operating).

The user interface is web-based, making use of PHP, JavaScript, HTML5 and CSS. This ensures a maximum compatibility without the need to install e.g. a Java-based client locally. Moreover, all meta-data is stored at the same place (the central MySQL database), which simplifies backups and maintenance. As PROMETHEUS is currently in the prototype stage, not all functions in the GUI have been implemented yet. Usually, the frontend offers multiple ways to perform a certain action (e.g. the graphical and textual template parsers), providing different options suited for varying necessities in the user experience. Besides setting up workflows, the GUI can also be used to visually inspect results: either by looking at generated plots (e.g. by MDplot, see chapter 7) or by generating them. Files can also be searched and the implemented filebrowser offers a multitude of functions like down- or uploading, zipping and special functions based on the filetype. For the average user, the frontend is the general working environment.

6.3.1 Database structure

The database is the central information pool, keeping track of users, templates, processes and so on. It is not accessed by the backend though and therefore no SQL access from the executing nodes to the frontend server is required. The listings below describe various tables in detail according to their CREATE queries. Below every listing, a text explanation is given.

Listing 6.1: Database: CLUSTERCONFIGS

```

1 CREATE TABLE IF NOT EXISTS 'CLUSTERCONFIGS' (
2   'ConfigID' bigint(20) NOT NULL AUTO_INCREMENT,
3   'UserID' bigint(20) NOT NULL,
4   'ConfigName' varchar(75) NOT NULL,
5   'ConfigContent' text NOT NULL,
6   PRIMARY KEY ('ConfigID')
7 );

```

Table CLUSTERCONFIGS, as shown in listing 6.1, stores different settings to facilitate the cluster communication by the frontend. It has two fields (ConfigID and UserID) that are used for identification and manipulation, a name field to describe its content and a ConfigContent text field. For an example of the value of ConfigContent, see format definition in listing 6.14, which is as a whole integrated in this text field.

Listing 6.2: Database: METAJOBS

```

1 CREATE TABLE IF NOT EXISTS `METAJOBS` (
2   `MetajobID` int(10) unsigned NOT NULL AUTO_INCREMENT,
3   `MetajobName` text NOT NULL,
4   `MetajobGenerationTS` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
5   `UserID` bigint(20) unsigned NOT NULL,
6   `MetajobRootdirectory` varchar(300) NOT NULL,
7   `MetajobPriority` tinyint(4) NOT NULL,
8   PRIMARY KEY (`MetajobID`)
9 );

```

In this table, the first four fields are used to access, describe, sort and authenticate a metajob. The metajob has a root directory, whose parent directories cannot be accessed by the file browser. The priority functionality assigning different weights to metajobs is not yet implemented. The MetajobID is used by tasks to refer to their metajob.

Listing 6.3: Database: PERMISSIONS

```

1 CREATE TABLE IF NOT EXISTS `PERMISSIONS` (
2   `PermissionID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `GroupID` bigint(20) NOT NULL,
4   `PermissionKeyphrase` varchar(30) NOT NULL,
5   `PermissionDescription` text NOT NULL,
6   `PermissionGranted` tinyint(1) NOT NULL DEFAULT '0',
7   PRIMARY KEY (`PermissionID`)
8 );

```

PROMETHEUS implements a complex permission system, enabling the detailed rights management for individual groups of users. A distinct permission always links a decision on a certain ability to a user group. That means, that every kind of permission is instantiated for every user group. Identification via the frontend functions is facilitated by the PermissionKeyphrase field. By default, a new kind of permission is instantiated with zero (meaning "not-granted") for every user group. Table 6.5 lists currently available permissions.

Table 6.5: Table of the currently implemented permissions, used to restrict access to certain functions in the frontend. New permissions can be added to the database easily and readily applied in further development.

PermissionID	PermissionKeyphrase	PermissionDescription
1	EDIT_TEMPLATES_SELF	Allows users to edit their own templates.
2	ADD_USERS	Allows users to add new users.
3	EDIT_SELF	Allows users to edit their own profile.
4	EDIT_USERS	Allows users to edit other users.
5	START_JOBS_SELF	Allows users to start jobs with their identification.
6	START_JOBS_OTHERS	Allows users to start jobs with foreign identification.
7	EDIT_TEMPLATES_GROUP	Allows users to edit templates of their group.
8	EDIT_TEMPLATES_PUBLIC	Allows users to edit public templates.
9	EDIT_TEMPLATES_OTHER	Allows users to edit private templates of other users.
10	GLOBAL_ADMINISTRATION	Allows users to access the administration panel.
11	LIST_USERS	Allows to list all users registered.

PermissionID	PermissionKeyphrase	PermissionDescription
23	EDIT_CREDITS	Allows group members to set the credits of users.
25	EDIT_USERENABLED	Allows users to set users enabled or disabled.
27	DELETE_USERS	Allows users to delete other users.
29	READ_TEMPLATES_OTHER	Allows to read private templates of other users.
31	READ_TEMPLATES_GROUP	Allows to read templates from their own group.
33	READ_TEMPLATES_PUBLIC	Allows to read public templates.
36	PUBLISH_TEMPLATES_GROUP	Allows users to publish their templates in their own group.
37	PUBLISH_TEMPLATES_PUBLIC	Allows users to publish their templates system wide.
41	ADD_GROUPS	Allows users to add new groups.
42	DELETE_GROUPS	Allows users to delete groups.
43	EDIT_PERMISSION	Allows users to change group permissions.
47	EDIT_GLOBAL_SETTINGS	Allows users to edit the system-wide settings.

PermissionID	PermissionKeyphrase	PermissionDescription
257	END_JOBS_SELF	Allows users to end their own jobs.
259	END_JOBS_OTHERS	Allows users to end other users' jobs.
261	VIEW_JOBS_SELF	Allows users to see own jobs.
263	VIEW_JOBS_OTHERS	Allows users to see other jobs.
296	EDIT_JOBS_SELF	Allows to edit all own jobs.
298	EDIT_JOBS_OTHERS	Allows to work on jobs of other users.
300	EDIT_CLUSTER_CONFIGS	Allows users to edit the cluster configuration.
302	EDIT_CSS_FILES	Allows users to edit the cascading style sheet files.

Listing 6.4: Database: PROCESSES

```

1 CREATE TABLE IF NOT EXISTS `PROCESSES` (
2   `ProcessID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `TaskID` bigint(20) NOT NULL,
4   `ProcessInstructionfile` varchar(750) NOT NULL,
5   `ProcessDirectory` varchar(300) NOT NULL,
6   `UserID` bigint(20) NOT NULL,
7   `ProcessLastUpdateTS` datetime NOT NULL,
8   `ProcessIdentifierOnCluster` varchar(45) DEFAULT NULL,
9   `ProcessStatusOnCluster` varchar(75) NOT NULL,
10  `ProcessConfigFile` varchar(750) NOT NULL,
11  PRIMARY KEY (`ProcessID`)
12 );

```

Processes are the actual units of execution from the frontends' point of view. They always belong to one single task (and these in turn to a metajob) and run in their own, exclusive folder. The folder path is stored in field `ProcessDirectory` and a process expects an *executor.ins* file within. The process table also stores information about every process in terms of its cluster status, which is uploaded by using the appropriate CCF file.

Listing 6.5: Database: SETTINGS

```

1 CREATE TABLE IF NOT EXISTS `SETTINGS` (
2   `SettingID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `SettingKeyphrase` varchar(300) NOT NULL,
4   `SettingDescription` text NOT NULL,
5   `SettingValue` varchar(300) NOT NULL DEFAULT '0',
6   PRIMARY KEY (`SettingID`)
7 );

```

In order to specify multiple frontend-related parameters, the `SETTINGS` table is used. Changes to this table are only allowed for the sites' administrator. Usually, the default values exhibit a proper balance between the distinct settings.

Listing 6.6: Database: SUBJOBGROUPS

```

1 CREATE TABLE IF NOT EXISTS `SUBJOBGROUPS` (
2   `SubjobgroupID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `SubjobgroupName` varchar(45) NOT NULL,
4   `UserID` bigint(20) NOT NULL,
5   PRIMARY KEY (`SubjobgroupID`)
6 );

```

This table serves as a collection memory for sub-conjugated subjobs, which form groups suited for batch processing. Their unique ID serves as an anchor point for the subjobs. Usually, these groups are not distributed over multiple metajobs (which is possible though).

Listing 6.7: Database: SUBJOBS

```

1 CREATE TABLE IF NOT EXISTS `SUBJOBS` (
2   `SubjobID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `SubjobgroupID` bigint(20) NOT NULL,
4   `SubjobDirectory` varchar(750) NOT NULL,
5   `SubjobIdentifier` varchar(350) NOT NULL,
6   `UserID` bigint(20) NOT NULL,
7   `MetajobID` bigint(20) NOT NULL,
8   PRIMARY KEY (`SubjobID`)
9 );

```

Subjobs are a logical entity representing a certain, encapsulated part of a project. Frequently, each subjob represents a different molecule, for which different processes (such as simulation, RMSD analysis, ...) grouped in multiple tasks can be performed. A new e.g. analysis can therefore be performed for a whole group of subjobs by batch instantiation to a new or existing task.

Listing 6.8: Database: TASKS

```

1 CREATE TABLE IF NOT EXISTS `TASKS` (
2   `TaskID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `TaskName` varchar(95) NOT NULL,
4   `MetaJobID` bigint(20) NOT NULL,
5   `TaskDefaultTemplate` bigint(20) NOT NULL,
6   `TaskDefaultConfig` bigint(20) NOT NULL,
7   `UserID` bigint(20) NOT NULL,
8   `TaskDefaultDirectory` varchar(750) DEFAULT NULL,
9   PRIMARY KEY (`TaskID`)
10 );

```

Tasks group the actual executions in the form of processes to the same or similar kind. Often, a simulation task is the first one established in a new project. It is recommended to assign every task its own subdirectory based on the metajobs' root directory and proceed the same way with the adjacent processes. This structuring allows a better control of access to certain parts of a project. For example, analyses tasks can be stored in a "rootdirectory/Analysis" folder.

Listing 6.9: Database: TEMPLATES

```

1 CREATE TABLE IF NOT EXISTS `TEMPLATES` (
2   `TemplateID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `TemplateName` varchar(90) NOT NULL,
4   `TemplateDescription` text NOT NULL,
5   `TemplateContent` text NOT NULL,
6   `TemplateAccess` enum('private','protected','public') NOT NULL,
7   `TemplateStatus` enum('draft','valid','invalid') NOT NULL,
8   `UserID` bigint(20) NOT NULL,
9   `TemplateGenerationTS` datetime NOT NULL,
10  `TemplateLastChangeTS` datetime NOT NULL,
11  `TemplateClonedirectory` varchar(450) NOT NULL,
12  PRIMARY KEY (`TemplateID`)
13 );

```

A blueprint for a certain kind of task is stored in the TEMPLATES table. It is the condensed workflow information involved in a specific task, which is instantiated for one or multiple subjobs to form actual processes. Typically, this is performed by replacing placeholders such as the subjob identifiers and directory paths.

Listing 6.10: Database: USERGROUPS

```

1 CREATE TABLE IF NOT EXISTS `USERGROUPS` (
2   `GroupID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `GroupName` varchar(75) NOT NULL,
4   `GroupIDrelated` bigint(20) NOT NULL,
5   PRIMARY KEY (`GroupID`)
6 );

```

To complete the permission system described before, user groups serve as a classification of the rights a certain user has. A user can only be member of one group at a time. A "credit" system has been proposed in the frontend (ultimately limiting the users' ability to use the cluster resources), but not yet implemented.

Listing 6.11: Database: USERS

```

1 CREATE TABLE IF NOT EXISTS `USERS` (
2   `UserID` bigint(20) NOT NULL AUTO_INCREMENT,
3   `UserName` varchar(30) NOT NULL,
4   `UserPassword` varchar(475) NOT NULL,
5   `GroupID` bigint(20) NOT NULL,
6   `UserEmail` varchar(200) NOT NULL,
7   `UserCredits` int(11) NOT NULL,
8   `UserEnabled` tinyint(4) NOT NULL DEFAULT '0',
9   PRIMARY KEY (`UserID`)
10 );

```

The implementation of users allows to relate metajobs to owners and protects data from intentional or accidental flaws. The password is stored doubly-salted and hashed version.

6.3.2 Cluster communication

To communicate with the cluster a cluster configuration file (CCF) is mandatory. It allows to specify those actions, that should be performed when frontend commands, such as "submit", "stop", etc. are executed. This allows a complete encapsulation of the cluster system in use from the processes. Given that no internal settings are changed in the instruction file, it is possible to transfer one template between different servers hosting PROMETHEUS using different cluster management tools. Moreover, one can specify how to access different clusters from within the same interface. To ensure a broad applicability regular expressions are available. For an example CCF file specified to be used with LUCI4HPC [21], see format listing 6.14.

6.3.3 Important functions and classes

The frontend implements a large number of functions in both PHP and JavaScript. In table 6.6, a few examples are shown. Classes are not described for the backend in detail, but serve as data structures and facilitate database loading / writing in most cases.

Table 6.6: List of exemplary frontend functions, together with their parameters, return values and a short description.

name	parameters	description
<code>do_login()</code>	<code>\$username</code> <code>\$password</code> <code>return true false</code>	checks credentials and updates session
<code>do_logout()</code>		clears session
<code>generateProgressBarDIV()</code>	<code>\$completed</code> <code>\$total</code> <code>return \$divString</code>	generates the progress bar for the process menu
<code>getMetajobs()</code>	<code>\$UserID</code> <code>return \$rowArray</code>	example of a loading function
<code>hasPermission()</code>	<code>\$UserID</code> <code>\$PermissionKeyphrase</code> <code>return true false</code>	checks the status of a user regarding a certain permission
<code>print_process_row()</code>	<code>\$pID</code>	generates the whole menu row for one given process

6.4 Formats

An integral part of PROMETHEUS are the file formats. As described earlier, these should be robust but also impose as little overhead as possible both in terms of parsing efficiency and disk space. The XML-based formats used, offer both of these advantages. The examples shown below are excerpts from actual log, error, cluster configuration files, etc. to show their content and structure.

6.4.1 Log / error file format

Listing 6.12: Log and error file

```

1 <?xml version="1.0" ?>
2 <log>
3   <header>
4     <base></base>
5     <description>Logfile for program executor</description>
6     <start>1413490609</start>
7     <end>1413635532</end>
8   </header>
9   <body>
10    <logItem>
11      <information>Module executor has been initialized successfully.</information>
12      <start>1413490609</start>
13    </logItem>
14    <logItem>
15      <information>Execution call of &quot;new_cuda/program/md&quot; has been completed.</information>
16      <start>1413493219</start>
17    </logItem>
18    <logItem>
19      <information>Execution result written out to file: &quot;MOH1/md/md_195.ond&quot;.</information>
20      <start>1413493219</start>
21    </logItem>
22    ...

```

In order to report events, warnings and errors in a standardized way, the executor program uses two files (*executor.log* and *executor.err*) where all successful step executions and errors are stored. Note, that timestamps are also stored in order to give an impression on the respective execution time. Output written to the standard output and standard error output streams are captured from child processes and added as well. The current behaviour is to overwrite the log and error files upon restarting a process.

6.4.2 Antibody file format

Listing 6.13: Antibody file format

```

1 <?xml version="1.0" ?>
2 <antibody>
3   <header>
4     <version>1.0</version>
5     <name>MOH1.pdb</name>
6     <disulphides>
7       <disulphide number="1">
8         <from chain="L" residue="23"></from>
9         <to chain="L" residue="88"></to>
10      </disulphide>
11      <disulphide number="2">
12        <from chain="H" residue="22"></from>
13        <to chain="H" residue="96"></to>
14      </disulphide>
15    </disulphides>
16  </header>
17  <structure>
18    <chain name="H">
19      <ranges>
20        <range number="1" type="F" typeindex="1" length="30"></range>
21        <range number="2" type="C" typeindex="1" length="5"></range>
22        <range number="3" type="F" typeindex="2" length="14"></range>
23        <range number="4" type="C" typeindex="2" length="16"></range>
24        <range number="5" type="F" typeindex="3" length="33"></range>
25        <range number="6" type="C" typeindex="3" length="13"></range>
26        <range number="7" type="F" typeindex="4" length="10"></range>
27      </ranges>

```

```

28 <residue number="1" name="GLY">
29 <atom number="1" name="N">
30 <xcoor>76.211</xcoor>
31 <ycoor>56.102</ycoor>
32 <zcoor>98.428</zcoor>
33 <element>N</element>
34 <charge>1</charge>
35 <occupancy>1</occupancy>
36 <tempFactor>0</tempFactor>
37 <iCode></iCode>
38 <altLoc></altLoc>
39 </atom>
40 <atom number="2" name="CA">
41 <xcoor>75.978</xcoor>
42 <ycoor>54.801</ycoor>
43 <zcoor>99.072</zcoor>
44 <element>C</element>
45 <charge></charge>
46 <occupancy>1</occupancy>
47 <tempFactor>0</tempFactor>
48 <iCode></iCode>
49 <altLoc></altLoc>
50 </atom>
51 ...

```

To store the additional information required when dealing with antibodies, e.g. the Kabat region definitions in MAMA-notation, a XML-based format can be used. As shown in listing 6.13, disulphide bridges are stored in the header region, while the ranges are associated with the respective protein chains. For the interconversion of PDB and ABF files, see the description of the backend programs `pdb2abf` and `abf2pdb`.

6.4.3 Cluster configuration file format

Listing 6.14: Cluster configuration file

```

1 <?xml version="1.0" ?>
2 <configuration>
3 <header>
4 <version>1.0</version>
5 <caption>SIMULATION WITH CUDA</caption>
6 <description>OXTseries</description>
7 </header>
8 <configblocks>
9 <configblock number="1" key="resubmit" type="local">
10 <caption>Do resubmit</caption>
11 <description>Tells the program, how to resubmit itself</description>
12 <execution type="external">
13 <command type="synchronous">/home/common/grid/lqsub</command>
14 <parameters>
15 <parameter number="1" key="command">-j executor -c 4 -g 1 -p omp -q graphic</parameter>
16 </parameters>
17 </execution>
18 </configblock>
19 <configblock number="2" key="getstatusoncluster_running" type="remote">
20 <caption>Read job id</caption>
21 <description>Contains instructions</description>
22 <execution type="internal">
23 <callFunction type="general">regexexpression()</callFunction>
24 <parameters>
25 <parameter number="1" key="input">(func=1::execute::command=/home/common/grid/lqstat -a)
26 <parameter number="2" key="regextract">(func(1)::spot::return) (func(1)::end)</parameter>
27 <parameter number="3" key="regmatch">[global::currentjobID] (.*)h89400_scrooge</parameter>
28 </parameters>
29 </execution>
30 </configblock>
31 <configblock number="3" key="getstatusoncluster_queueing" type="remote">
32 <caption>Read job id</caption>
33 <description>Contains instructions</description>
34 <execution type="internal">
35 <callFunction type="general">regexexpression()</callFunction>
36 <parameters>
37 <parameter number="1" key="input">(func=1::execute::command=/home/common/grid/lqstat -a)
38 <parameter number="2" key="regextract">Queueing: (.*) Deleting:</parameter>
39 <parameter number="3" key="regmatch">h89400_scrooge (.*) [global::jobdirectory]</parameter>
40 </parameters>
41 </execution>
42 </configblock>
43 <configblock number="4" key="submit" type="local">
44 <caption>Do submit</caption>
45 <description>Tells the program, how to submit itself</description>
46 <execution type="external">
47 <command type="synchronous">/home/common/grid/lqsub</command>
48 <parameters>
49 <parameter number="1" key="">-j executor -c 4 -g 1 -p omp -q graphic</parameter>
50 </parameters>
51 </execution>
52 </configblock>

```

```

55 <configblock number="5" key="stop" type="local">
56 <caption>Stop</caption>
57 <description>Stop the run</description>
58 <execution type="external">
59 <command type="synchronous">/home/common/grid/lqdel</command>
60 <parameters>
61 <parameter number="1" key="">[global::currentjobID]</parameter>
62 </parameters>
63 </execution>
64 </configblock>
65 </configblocks>
66 </configuration>

```

To facilitate the cluster communication performed by the frontend, every process has a corresponding CCF or cluster configuration file. Every file has five `configblock` tags, defining how a process should be submitted (e.g. how many cores should be used), monitored and cancelled. These configuration files are suited for a certain process / cluster system configuration and can be changed in case a different setting is required.

6.4.4 Statistical / data file format

Listing 6.15: Statistical data file

```

1 <?xml version="1.0" ?>
2 <analysis>
3 <header>
4 <version>1.0</version>
5 <title></title>
6 <created>1418119431</created>
7 <modified>1426791244</modified>
8 </header>
9 <datablock number="1" name="" label="" reference="">
10 <table number="1" name="" reference="" type="data">
11 <rows />
12 <columns>
13 <column number="1" name="" unit="" label="" reference="" type="double">
14 <item number="1">10000</item>
15 <item number="2">10001</item>
16 <item number="3">10002</item>
17 <item number="4">10003</item>
18 <item number="5">10004</item>
19 ...

```

To deal with large amounts of analysis data, a XML-based storage format has been derived, which allows the clustering, merging, searching and manipulation of rows and columns of data. The variety of programs associated with files of type `.stat` is not described in detail in this work. Future versions of PROMETHEUS are supposed to replace this functionality by a native R interface.

6.4.5 Instruction file format

Listing 6.16: Instruction file: header

```

1 <?xml version="1.0" ?>
2 <instruction>
3 <header>
4 <version>1.22</version>
5 <caption>Template for simulation of models in series</caption>
6 <variables>
7 <variable namespace="global" type="mandatory" key="prometheusbinarypath">../binaries</variable>
8 <variable namespace="global" type="mandatory" key="jobdirectory">../simulation/model_1</variable>
9 <variable namespace="global" type="mandatory" key="jobidentifier">model_1</variable>
10 <variable namespace="global" type="mandatory" key="gromosbinarypath">../gromos++/programs</variable>
11 <variable namespace="global" type="mandatory" key="gromosXXpath">../gromosXX/program</variable>
12 <variable namespace="global" type="mandatory" key="maxsodiums">0</variable>
13 <variable namespace="global" type="mandatory" key="maxchlorides">0</variable>
14 <variable namespace="global" type="mandatory" key="maximumsimpacts">1000</variable>
15 <variable namespace="global" type="mandatory" key="currentjobID"></variable>
16 </variables>
17 <subjobgroups>
18 <subjobgroup number="1">
19 <subjob number="1" subjobidentifier="runner_1" subjobdirectory=""></subjob>
20 <subjob number="2" subjobidentifier="runner_2" subjobdirectory=""></subjob>
21 <subjob number="3" subjobidentifier="runner_3" subjobdirectory=""></subjob>
22 <subjob number="4" subjobidentifier="runner_4" subjobdirectory=""></subjob>

```

```

23 <subjob number="5" subjobidentifier="runner_5" subjobdirectory=""></subjob>
24 <subjob number="6" subjobidentifier="runner_6" subjobdirectory=""></subjob>
25 <subjob number="7" subjobidentifier="runner_7" subjobdirectory=""></subjob>
26 <subjob number="8" subjobidentifier="runner_8" subjobdirectory=""></subjob>
27 <subjob number="9" subjobidentifier="runner_9" subjobdirectory=""></subjob>
28 </subjobgroup>
29 </subjobgroups>
30 </header>

```

The top of any instruction file forms the header region, where miscellaneous settings can be adjusted. The first mandatory tag is the version number (currently number 1.22) to ensure format compatibility with the used executables. Afterwards, a caption should be defined that allows identification of the process' purpose. The variables tag embeds a dynamic list of variable definitions with the following attributes: namespace (set to *global*, because it is accessible throughout the whole instruction file), a type which is either *mandatory* or *setbyfrontend* and most importantly the key, by which the variable is addressed. The value of the variable is afterwards inserted, whenever the variable is called (note, that changes to a global variable are not permitted after definition). The user can specify variables as required, some are mandatory though. The variable with key *jobdirectory* has to be set in any case and key *jobidentifier* might also be commonly used. If communication with a cluster system is desired, *currentjobID* should also be set. In this example, other variables are also defined (e.g. the path to the PROMETHEUS and GROMOS binaries and the number of maximum sodium and chloride ions for later neutralisation of the simulations' box net-charge). In case multiple subjobs are involved in the execution (e.g. the RMSD of the backbone should be calculated for a whole set of subjobs), then subjobgroups can be defined, as shown in listing 6.16. Every subjob consists of a number, a subjobidentifier and a subjobdirectory, respectively. Note, that the subjobs must be numbered sequentially. Batch processing is always an attractive option to minimize repetitive user input and ensure consistency among a set of processes.

Listing 6.17: Instruction file: preparation

```

1 ...
2 <preparation>
3   <step number="1" done="false">
4     <caption>Job update</caption>
5     <information>Updates the job ID in this file</information>
6     <execution type="internal">
7       <callFunction type="general">update_INS_variable()</callFunction>
8       <parameters>
9         <parameter number="1" key="instructionfile">[global::jobdirectory]/executor.ins</parameter>
10        <parameter number="2" key="variablekey">currentjobID</parameter>
11        <parameter number="3" key="input">(func=1::environment::selector=JOBID)
12          (func(1)::spot::value) (func(1)::end)</parameter>
13      </parameters>
14    </execution>
15  </step>
16 </preparation>
17 ...

```

In order to store the job number retrieved from the cluster management system (in this example handed over by setting an environmental variable), preparation steps can be defined directly after the header section. As implemented in an internal function, called `update_INS_variable()` which takes the instruction file, a variable key (default: *currentjobID*) and the way the identifier is retrieved as parameters (see listing 6.17). Here, already the general layout of a step definition is illustrated: every step is an atomic processing unit having a number and a status as attributes. The number is sequential on the respective depth while *done* stores whether the step has been completed successfully (in which case it is *true*) or not (note, that any other value than *true* will default to *false*). Every step also has a caption and an information tag and the actual processing logic is stored in the execution part. The differences in the execution

types are described below. Note, that parameter three in this example gets the value of a macro which in this case retrieves the environmental variable *JOBID* and inserts it on loading the instruction file (preparation is executed every time prior to the execution of processing unto loading the file). A description of all macros together with examples is provided in section 6.2.3.

Listing 6.18: Instruction file: internal calls

```

1  ...
2  <processing>
3  <step number="1" done="false">
4  <caption>Update arg files</caption>
5  <information>Prepare argument files</information>
6  <execution type="internal">
7  <callFunction type="gromos">prepare_make_top()</callFunction>
8  <parameters>
9  <parameter number="1" key="inputabf">[global::jobidentifier].abf</parameter>
10 <parameter number="2" key="templateargfile">../make_top_template_chains.arg</parameter>
11 <parameter number="3" key="Nterminus">NH3+</parameter>
12 <parameter number="4" key="Cterminus">COO-</parameter>
13 <parameter number="5" key="argfileoutputdirectory">[global::jobdirectory]/topology</parameter>
14 </parameters>
15 </execution>
16 </step>
17 </processing>
18 ...

```

If the execution type is set to be *internal*, this means that the execution does not actually call a program but rather invokes an internally compiled function. That is especially useful in cases where complicated adjustments have to be made like the rearrangement of atoms for a certain group to place them at meaningful initial positions. *internal* executions have different callFunction types (in the example it is set to be *gromos*) and the value of callFunction determines which function should be invoked. As usual, the parameters that are passed can make use of variable replacement and macros / functions. New internal functions are easily integrated in the source code due to its modularity (this requires a recompilation of the backend though).

Listing 6.19: Instruction file: external calls (synchronous)

```

1  ...
2  <step number="2" done="false">
3  <caption>Generate file as abf</caption>
4  <information>Generate the abf file</information>
5  <execution type="external">
6  <command type="synchronous">[global::prometheusbinarypath]/pdb2abf</command>
7  <parameters>
8  <parameter number="1" key="@input">[global::jobdirectory]/[global::jobidentifier].pdb</parameter>
9  <parameter number="2" key="@output">[global::jobdirectory]/[global::jobidentifier].abf</parameter>
10 <parameter number="3" key="@nointerface"></parameter>
11 <parameter number="4" key="@setchain">A</parameter>
12 </parameters>
13 </execution>
14 </step>
15 ...

```

Steps, that implement external calls most often have the command type *synchronous*. The value of the command specifies the binary while the parameters are passed to the underlying program. A *synchronous* call will stall further execution until the call is completed. Every parameter has a number which must be set sequentially and also a key which can be set in case required (it can also be left blank). The example in listing 6.19 shows a call of a stand-alone PROMETHEUS binary, that translates a PDB file into an ABF file. The parameter "@nointerface" suppresses the UI (see program description of *pdb2abf*), which would otherwise lead to a pause, since user input would be expected.

Listing 6.20: Instruction file: external calls (backtick)

```

1  ...
2  <step number="3" done="false">
3    <caption>Minimize (in vacuo)</caption>
4    <information>Set all atoms which are in origin to appropriate coordinates</information>
5    <execution type="external">
6      <command type="backtick">[global::gromosXXbinarypath]/md</command>
7      <parameters>
8        <parameter number="1" key="@topo">./topology/[global::jobidentifier].top</parameter>
9        <parameter number="2" key="@fin">./md/[global::jobidentifier].cnf</parameter>
10       <parameter number="3" key="@conf">./coord/[global::jobidentifier].cnf</parameter>
11       <parameter number="4" key="@input">./coord/arrange.imd</parameter>
12       <parameter number="5" key="@refpos">./coord/[global::jobidentifier].rpr</parameter>
13       <parameter number="6" key="@posresspec">./coord/[global::jobidentifier].por</parameter>
14     </parameters>
15     <outputfile format="plain">[global::jobdirectory]/[global::jobidentifier]_arrange.umd</outputfile>
16   </execution>
17 </step>
18 ...

```

Another possibility of external command types is *backtick*, which stores the output produced by the called program into a file that is specified after the `parameters` tag. Currently, only the *plain* format is supported, which is a mere text file. In many cases, the output of one step is required as input by the next steps. However, a step could fail, in which case PROMETHEUS ensures a complete halt of the execution (unlike a bash script for example).

Listing 6.21: Instruction file: substeps

```

1  ...
2  <step number="4" done="false">
3    <caption>Topologies</caption>
4    <information>Generate topologies for files</information>
5    <execution type="substeps">
6      <step number="1" done="false">
7        <caption>Update arg files</caption>
8        <information>Prepare argument files</information>
9        <execution type="internal">
10         <callFunction type="gromos">prepare_make_top()</callFunction>
11         <parameters>
12           <parameter number="1" key="inputabf">./[global::jobidentifier].abf</parameter>
13           <parameter number="2" key="templateargfile">./topology/make_top_template.arg</parameter>
14           <parameter number="3" key="Nterminus">MECO</parameter>
15           <parameter number="4" key="Cterminus">NAM</parameter>
16           <parameter number="5" key="argfileoutputdirectory">[global::jobdirectory]/topology</parameter>
17         </parameters>
18       </execution>
19     </step>
20     <step number="2" done="false">
21       <caption>Generate topology</caption>
22       <information>Generating chain A topology</information>
23       <execution type="external">
24         <command type="backtick">[global::gromosbinarypath]/make_top</command>
25         <parameters>
26           <parameter number="1" key="@f">[global::jobdirectory]/topology/make_top_A.arg</parameter>
27           <parameter number="2" key="@build">[global::jobdirectory]/topology/54a7.mtb</parameter>
28           <parameter number="3" key="@param">[global::jobdirectory]/topology/54a7.ifp</parameter>
29         </parameters>
30         <outputfile format="plain">./topology/[global::jobidentifier]_combined.top</outputfile>
31       </execution>
32     </step>
33   </execution>
34 </step>
35 ...

```

Apart from the step types explained so far, another possibility is the execution of type *substeps*, in which case the step serves as a structuring feature. Substeps can have loops within them: after every full completion of their subordinate steps, these steps' done attribute is set to *false* and the next iteration begins. For detailed information on iterations see listings 6.23 and 6.24. Note, that if the parent step attribute has been set to *true*, the status of its substeps are ignored.

Listing 6.22: Instruction file: functions

```

1  ...
2  <step number="5" done="true">
3    <caption></caption>
4    <information>Combine all files</information>
5    <execution type="external">
6      <command type="synchronous">[global::prometheusbinarypath]/merge_stat</command>
7      <parameters>
8        <parameter number="1" key="@input">(func=1::expand::subjobs=1::1-1::end)
9          [global::jobdirectory]/(func(1)::spot::subjobidentifier)_CDR_total.plain (func(1)::end)
10       </parameter>
11       <parameter number="2" key="@output">[global::jobdirectory]/CDR_total.stat</parameter>
12       <parameter number="3" key="@select">/analysis/datablock/table[@number=1]</parameter>
13     </parameters>
14   </execution>
15 </step>
16 ...

```

Functions or macros are small built-in functionalities that provide simple implementation of tasks required often. They can be applied to parameter values, as shown in listing 6.17, where function 1 retrieves an environmental variable. It is possible, to apply multiple macros in one value region. However, the numbering should be done with care, since it is part of the spot identification. For a description on the macros, see section 6.2.3.

Listing 6.23: Instruction file: iteration numerical

```

1  ...
2  <step number="6" done="false">
3    <caption>Equilibrate</caption>
4    <information>Execute equilibration</information>
5    <iteration type="numbers">
6      <minimum>1</minimum>
7      <current>1</current>
8      <maximum>5</maximum>
9    </iteration>
10   <execution type="internal">
11     <callFunction type="gromos">md()</callFunction>
12     <parameters>
13       <parameter number="1" key="binary">[global::gromosXXCUDABinarypath]/md</parameter>
14       <parameter number="2" key="directory">[global::jobdirectory]/eq</parameter>
15       <parameter number="3" key="topology">[global::jobdirectory]/input.top</parameter>
16       <parameter number="4" key="initialcoordinatefile">input_combined_H2O_minimized.cnf</parameter>
17       <parameter number="5" key="previouscoordinatefile">input.cnf</parameter>
18       <parameter number="6" key="inputmdfile">eq.imd</parameter>
19       <parameter number="7" key="rprfile">eq.rpr</parameter>
20       <parameter number="8" key="porfile">eq.por</parameter>
21       <parameter number="9" key="tempcoordinatefile">(func=1::environment::selector=TMPDIR)
22         (func(1)::spot::value) (func(1)::end)/eq_[local::iteration::current].cnf</parameter>
23       <parameter number="10" key="tempenergyfile">(func=1::environment::selector=TMPDIR)
24         (func(1)::spot::value) (func(1)::end)/eq_[local::iteration::current].trc</parameter>
25       <parameter number="11" key="temptrajjectoryfile">(func=1::environment::selector=TMPDIR)
26         (func(1)::spot::value) (func(1)::end)/eq_[local::iteration::current].trc</parameter>
27       <parameter number="12" key="outputfile">eq.umd</parameter>
28       <parameter number="13" key="exportvariables">LD_LIBRARY_PATH=/usr/local/cuda/lib64:
29         /usr/local/cuda/lib:/usr/lib/nvidia-current:/home/common/GROMOS/lib/cukernellib</parameter>
30       <parameter number="14" key="currentiteration">[local::iteration::current]</parameter>
31     </parameters>
32   </execution>
33 </step>
34 ...

```

An iteration is a very important flow control structure. In this case a numerical iteration (type set to *numbers*), where a step is executed several times. The index variable runs from a minimum to a maximum integer value: after every completion the current variable in the iteration tag is incremented by 1. In order to access the number of the current iteration, the insertion variable `[local::iteration::current]` may be used. Note, that substep iterations are also accessible in a subordinate step: then the namespace identifier is parent for every depth level upwards, e.g.

`[parent::parent::parent::iteration::current]` in order to access a running variable three levels above the current step.

Listing 6.24: Instruction file: iteration subjobs

```

1  ...
2  <step number="7" done="false">
3    <caption>Do for the subjobs</caption>
4    <information>Execute the following steps for every individual subjob</information>
5    <iteration type="subjobgroups">
6      <minimum>1::1</minimum>
7      <current>1::1</current>
8      <maximum>1::end</maximum>
9    </iteration>
10   <execution type="substeps">
11     <step number="1" done="false">
12       <caption>Do folder</caption>
13       <information>Generate a subfolder if necessary</information>
14       <execution type="external">
15         <command type="synchronous">[global::prometheusbinarypath]/reset_ins</command>
16         <parameters>
17           <parameter number="1" key="@input">../executor.ins</parameter>
18           <parameter number="2" key="@from">1::3</parameter>
19         </parameters>
20       </execution>
21     </step>
22   </execution>
23 </step>
24 </processing>
25 </instruction>

```

Instead of running from one integer value to another, the user can also specify loops over defined subjobs. In order to enhance applicability, there is the *end* keyword which always defaults to the number of subjobs in a subjobgroup and therefore does not need to be set by hand. As before, the variables can be accessed by `local::iteration::` or `parent::iteration::` prefixes and the `subjobidentifier` or `subjobdirectory` postfixes, respectively. A replacement can be combined with others or maybe used an arbitrary number of times in a single parameter value. As usual, insertion variables need to be enclosed by square brackets.

6.5 Graphical User Interface

In this section, excerpted snapshots of the frontends' graphical user interface (GUI) are shown and explained. All examples are based on actually performed simulation and analysis tasks, done in the course of this thesis. For a visual demonstration, see the video in section 6.1.6.

Templates



private templates

Number	Name	Description	Author	Status	Last change	Options
1	TRseries simulation	This instruction file is meant to facilitate simulations of the TR series, which contains anti-bodies. Only one molecule each.	Christian	valid	2014-06-04 13:27	
6	OXTseries	The OXTseries is based on the TOXseries, but the sequence has been inverted in order to test the effect of the dipole manipulation of residues. Length is 100 nanoseconds.	Christian	valid	2013-12-02 00:00	
7	TOX - RMSD,RMSF,DSSP	Template file to calculate the RMSD, RMSF and secondary structure - using DSSP - for the TOX series, which holds short helical peptide fragments.	Christian	valid	2014-10-31 12:54	
8	DSSP TOX - last 80ns with split	This template is for calculation of the development of the secondary structure for the helix project with the TOXseries. Does only cover the last 80% of a 100ns simulation trajectory. It generates three output files: 1) complete helix 2) part with hetero-residues 3) first seven residues	Christian	valid	2014-01-14 22:48	

Figure 6.10: Shows the template overview for a certain user. Each of them has a name along with a short description and a timestamp. Templates are stored in the frontends' MySQL database and can be viewed in either the editor (see figure 6.16 for an example) or in the specialised graphical user interface shown in figure 6.11. They are either private (for the users' use only) or public, in which case it is read-accessible by all users.

Template #1

Meta



Name:

TRseries simulation

Description:

This instruction file is meant to facilitab

Author:

Clone directory:

/pool/h89400_scrooge/template_packagi

Header



Processing



1	Preparation	1
2	Topologies	7
3	Coordinates	5
4	Minimization	2
5	Boxing	0
6	Solvent-minimization	4
7	Ions	2
8	Equilibration	5
9	Production runs	4

Step: 1



Status:

false

Caption:

Preparation

Information:

Generate folder structure and

Child steps

Step: 1



Status:

false

Caption:

Renaming

Information:

Rename all files according t

callFunction

general

Parameters

Number

1

Key

directory

Value

[global::jobdirectory]

Number

2

Key

searchpattern

Value

XXXX

Figure 6.11: The template editor allows to generate and maintain templates without the need of manipulating the source code directly. In the top section, meta-information such as the template description is monitored, followed by the header including variable definitions and cluster communication information. The main body represents the processing part, where the top-level steps are shown on the left according to the number of their respective sub-steps. Every top-level step can be manipulated on the right hand side.

Metajob OXTseries

Subjobs



Number	Group	Identifier	Directory	Actions
2	35	OXT1	/pool/h89400_scrooge/metajobs/OXTseries/OXT1	
3	36	OXT2	/pool/h89400_scrooge/metajobs/OXTseries/OXT2	
4	37	OXT3	/pool/h89400_scrooge/metajobs/OXTseries/OXT3	

simulation

Tasks

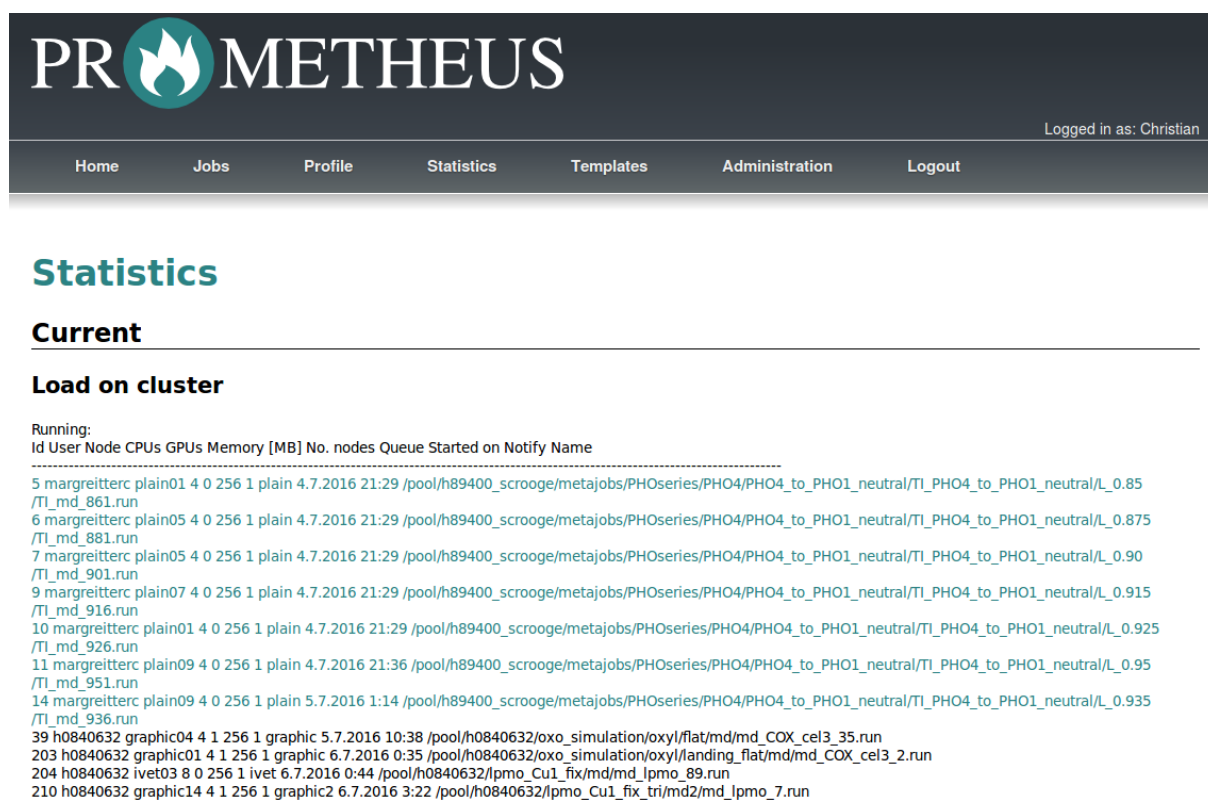


Task simulation



Number	On cluster	Files	Subdirectory	Status	Actions
119	completed	executor.ins cluster.ccf manage files	/pool/h89400_scrooge/metajobs/OXTseries/OXT1	531 / 531 (100%)	
120	completed	executor.ins cluster.ccf manage files	/pool/h89400_scrooge/metajobs/OXTseries/OXT2	531 / 531 (100%)	

Figure 6.12: Detailed view on the subjobs and the first task constituting a metajob. The subjobs are logical entities, typically different compounds. They can be batch-processed using the symbol in the upper right corner (the three arrows), which instantiates a selected template for all of them and forms processes (representing the actual execution). These processes are grouped into tasks, such as *"simulation"*, *"secondary structure"* or *"plotting"* depending on the selected template. The processes can be started by use of the rocket symbol. The status column shows the progress of the process in actual steps performed and percent, respectively. By clicking on *"manage files"*, the filebrowser (see figure 6.17) of the respective process is accessed. The user can either select all tasks at a time (which might impose some delay for large metajobs, due to the parsing of many instruction files) or select a single task at a time by a drop-down menu.



PROMETHEUS

Logged in as: Christian

Home Jobs Profile Statistics Templates Administration Logout

Statistics

Current

Load on cluster

Running:

Id	User	Node	CPUs	GPUs	Memory [MB]	No. nodes	Queue	Started on	Notify	Name
5	margreitterc	plain01	4	0	256	1	plain	4.7.2016 21:29	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.85	/TI_md_861.run
6	margreitterc	plain05	4	0	256	1	plain	4.7.2016 21:29	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.875	/TI_md_881.run
7	margreitterc	plain05	4	0	256	1	plain	4.7.2016 21:29	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.90	/TI_md_901.run
9	margreitterc	plain07	4	0	256	1	plain	4.7.2016 21:29	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.915	/TI_md_916.run
10	margreitterc	plain01	4	0	256	1	plain	4.7.2016 21:29	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.925	/TI_md_926.run
11	margreitterc	plain09	4	0	256	1	plain	4.7.2016 21:36	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.95	/TI_md_951.run
14	margreitterc	plain09	4	0	256	1	plain	5.7.2016 1:14	/pool/h89400_scrooge/metajobs/PHOseries/PHO4/PHO4_to_PHO1_neutral/TI_PHO4_to_PHO1_neutral/L_0.935	/TI_md_936.run
39	h0840632	graphic04	4	1	256	1	graphic	5.7.2016 10:38	/pool/h0840632/oxo_simulation/oxyl/flat/md/md_COX_cel3_35.run	
203	h0840632	graphic01	4	1	256	1	graphic	6.7.2016 0:35	/pool/h0840632/oxo_simulation/oxyl/landing_flat/md/md_COX_cel3_2.run	
204	h0840632	ivet03	8	0	256	1	ivet	6.7.2016 0:44	/pool/h0840632/lpmo_Cu1_fix/md/md_lpmo_89.run	
210	h0840632	graphic14	4	1	256	1	graphic2	6.7.2016 3:22	/pool/h0840632/lpmo_Cu1_fix_tri/md/md_lpmo_7.run	

Figure 6.13: Shows the header of the frontend, together with the menu and the jobs running on the cluster. The ones associated with the user currently logged in are highlighted in turquoise. This output is produced by the respective LUCI program.

Jobmanagement

Metajobs



















Number	Name	Generated	User	Rootdirectory	Priority	Actions
18	GCNseries	2013-11-14 16:15	Christian	/pool/h0840058/p0017/GCNseries	100	  
19	TRseries	2013-11-14 16:26	Christian	/pool/margreitterc/0014_Antibody_design/TRseries	100	  
40	OXTseries	2013-12-02 12:39	Christian	/pool/h89400_scrooge/metajobs/OXTseries	100	  
47	SINseries	2014-01-23 15:30	Christian	/pool/h89400_scrooge/metajobs/SINseries	100	  
48	EXTseries	2014-01-23 18:15	Christian	/pool/h89400_scrooge/metajobs/EXTseries	100	  
51	SICseries	2014-02-12 16:16	Christian	/pool/h89400_scrooge/metajobs/SICseries	100	  

Figure 6.14: List of metajobs belonging to the user currently logged in. The most important information is the root directory, from which the subjob, task and process folders are derived. The metajobs can be accessed by clicking the eye symbol. Each of them has a three-letter series' name in this excerpt, but it can be chosen freely.

Groups



Number	Name	Members	Actions
1	Administrators	1	
3	Users	1	
4	Beta-testers	1	

Permissions



Number	Key	Description	Actions
1	EDIT_TEMPLATES_SELF	Allows group members to edit their own templates.	
2	ADD_USERS	Allows the group members to add new users.	
3	EDIT_SELF	Allows the members to edit the own profile.	
4	EDIT_USERS	Allows the members to edit other users.	

Figure 6.15: Excerpt of the administration panel showing the currently defined groups of users and the upper part of the list of permissions. For every user group, every permission is set either to be granted or prohibited. A list of all implemented permissions is available in table 6.5.

Editor

Mode "database": TRseries simulation (1)

```

1 <?xml version="1.0"?>
2 <instruction>
3   <header>
4     <version>1.22</version>
5     <caption>Template for test runs</caption>
6   </header>
7   <variables>
8     <variable namespace="global" type="mandatory" key="prometheusbinarypath">/home/margreitterc/Desktop/Code/Antibody_modeler/binaries</variable>
9     <variable namespace="global" type="mandatory" key="jobdirectory"></variable>
10    <variable namespace="global" type="mandatory" key="jobidentifier"></variable>
11    <variable namespace="global" type="mandatory" key="gromosbinarypath">/pool/oostenbrink/GROMOS/gromos_270513/gromos++/new_cluster/programs</variable>
12    <variable namespace="global" type="mandatory" key="gromosXXCUDABinarypath">/pool/oostenbrink/GROMOS/gromos_270513/gromosXX/new_cuda/program</variable>
13    <variable namespace="global" type="mandatory" key="gromosXXbinarypath">/pool/oostenbrink/GROMOS/gromos_270513/gromosXX/new_cluster/program</variable>
14    <variable namespace="global" type="mandatory" key="maxnatriums">15</variable>
15    <variable namespace="global" type="mandatory" key="maxchlorides">15</variable>
16    <variable namespace="global" type="mandatory" key="maximumsimulationpackages">250</variable>
17    <variable namespace="global" type="mandatory" key="currentjobID" />
18  </variables>
19 </header>
20 <preparation>
21   <step number="1" done="false">
22     <caption>bie</caption>
23     <information>ne</information>
24     <execution type="internal">
25       <callFunction type="general">update_INS_variable()</callFunction>
26       <parameters>
27         <parameter number="1" key="instructionfile">[global::jobdirectory]/executor.ins</parameter>
28         <parameter number="2" key="variablekey">currentjobID</parameter>
29         <parameter number="3" key="input">{(func1::environment::selector=JOBID)(func1::spot::value)(func1::end)}</parameter>
30       </parameters>
31     </execution>
32   </step>
33 </preparation>
34 </instruction>

```

Figure 6.16: The file editor allows to directly manipulate templates, instruction files etc. in the browser window. It is based on CodeMirror [11] and CodeMirrorUI [12] and can be easily extended if necessary in further developments. For certain file types, special features are available e.g. text highlighting of XML files, syntax checks for instruction and cluster files and more. Depending on the source of the file, either mode "database" or "file" is used. An alternative for directly manipulating the source code of templates is the template GUI described above.

Manage files regarding process 119



Files

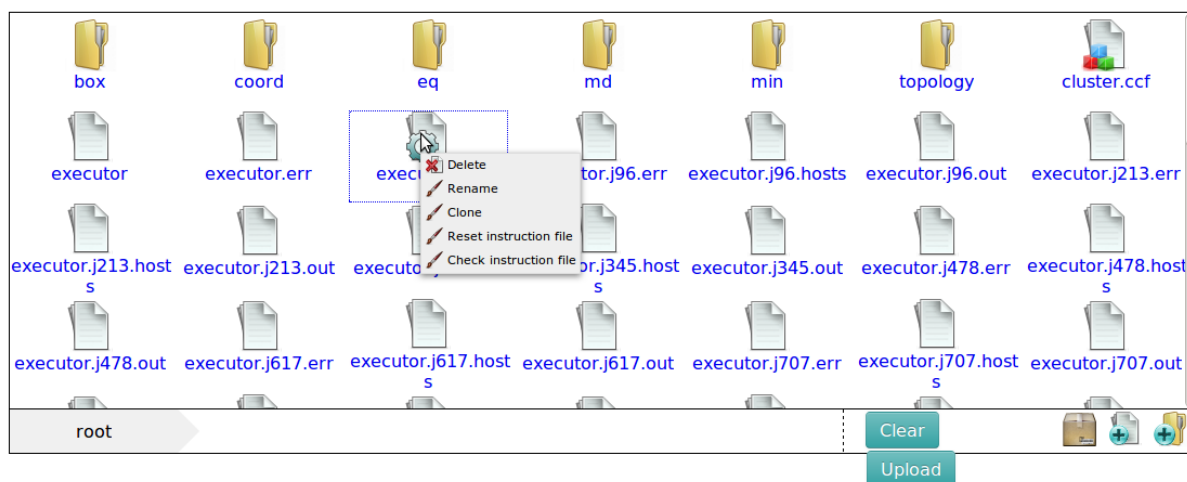


Figure 6.17: File browser showing the contents of a sub-folder regarding a process. Files and folders can be created, renamed, deleted, uploaded, downloaded and packed into a zipped archive. For certain file extensions (png, ins, ccf, ...) special menus are available by right-clicking onto them. This allows e.g. to reset instruction files (set the done attribute in steps to *false*), to check cluster configuration files for their validity or to display image files in the browser directly (see figure 6.18).

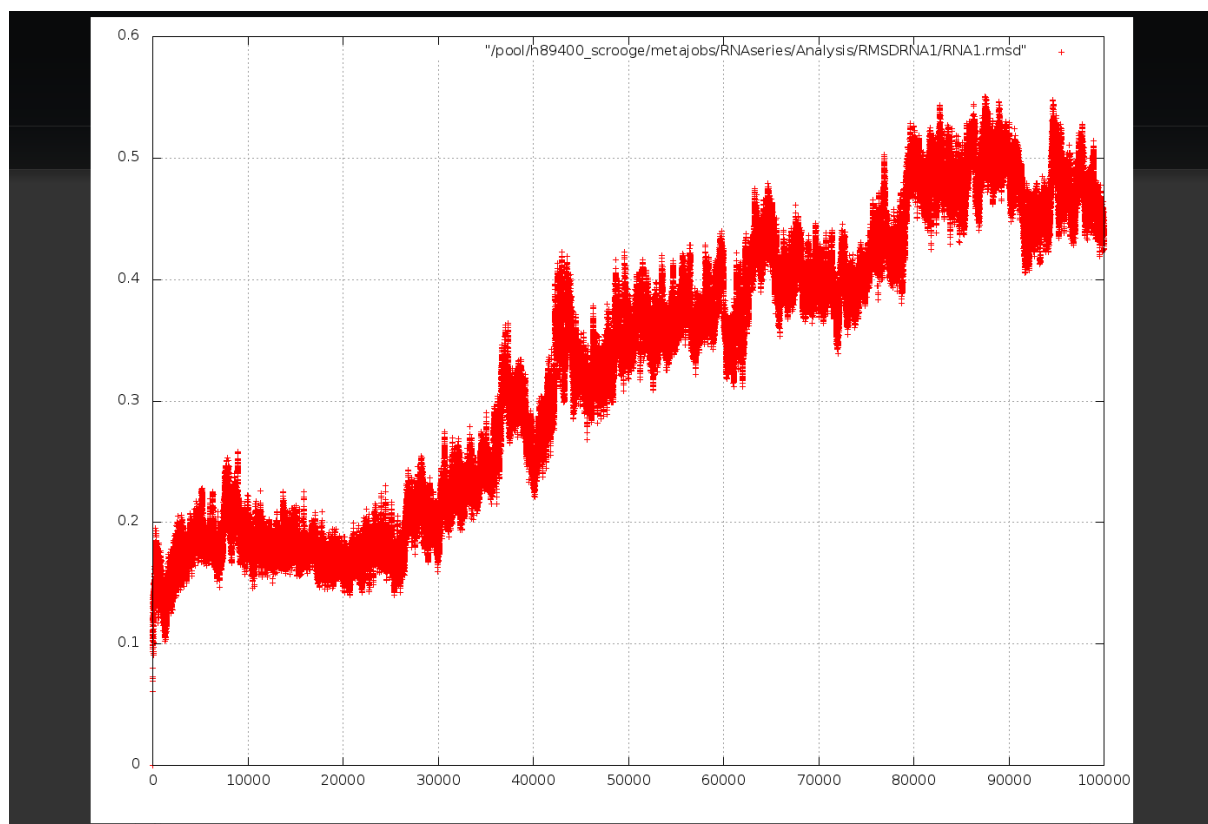


Figure 6.18: Figures such as a RMSD plot can be generated (and displayed) from calculated timeseries directly in the file browser. This is especially powerful if used in conjunction with MDplot (see chapter 7).

6.6 Outlook

In principle, a comprehensive simulation workflow manager as PROMETHEUS is never really complete. There are always new requirements, suggestions, bugs and extensions that could be addressed. However, a few things would be especially worthwhile to integrate and might be part of a future release. For example a very rewarding addition would be the native inclusion of R code: small snippets inserted directly in the instruction files, while bigger fragments could be loaded as packages in the preparation stage. This would enhance the power of PROMETHEUS dramatically in terms of dealing with and producing data and plots. Another obvious extension is the support of other molecular dynamics engines, such as GROMACS [22]. This would broaden the applicability, while it would be rather easy to implement (as a matter of fact, the required files are already prepared). The further development process is not yet decided on, but might include a whole set of extensions matching different users' needs.

References

- [1] Markus Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: [10.1002/jcc.20303](https://doi.org/10.1002/jcc.20303).
- [2] International Organization for Standardization (ISO). *ISO International Standard ISO/IEC 14882:2011 - Programming Language C++*.
- [3] Andy Rushton. *The STLplus C++ Library Collection*.
<http://stlplus.sourceforge.net>.
- [4] Brendan Eich. *JavaScript*.
<http://www.w3schools.com/js>.
- [5] jQuery Team. *jQuery*.
<https://jquery.com>.
- [6] Björn Brala. *jQuery Context Menu*.
<http://swisnl.github.io/jquery-contextMenu>.
- [7] Cory S.N. LaViska. *jQuery File Tree*.
<https://github.com/jqueryfiletree/jqueryfiletree>.
- [8] The PHP Group. *PHP: Hypertext Preprocessor*.
<http://www.php.net>.
- [9] Jim Wigginton. *phpseclib*.
<http://phpseclib.sourceforge.net>.
- [10] Oracle Corporation. *MySQL database*.
<http://www.mysql.com>.
- [11] Marijn Haverbeke. *CodeMirror*.
<https://codemirror.net>.
- [12] Jeremy Green. *CodeMirror UI*.
<https://github.com/jagthedrummer/codemirror-ui>.
- [13] Purecss Team. *pureform*.
<http://purecss.io/forms>.
- [14] eZ Components. *eZ Components*.
<http://ezcomponents.org>.
- [15] Fabien Doiron. *alertify.js - browser dialogs never looked so good*.
<http://fabien-d.github.io/alertify.js>.
- [16] Jack Moore. *Colorbox - a jQuery lightbox*.
<http://www.jacklmoore.com/colorbox>.
- [17] Lee Thomason. *TinyXML*.
<http://www.grinninglizard.com/tinyxml/index.html>.
- [18] Yves Berquin. *TinyXPath*.
<http://tinyxpath.sourceforge.net>.
- [19] libssh Team. *libssh*.
<http://www.libssh.org>.
- [20] Philip Hazel. *Perl Compatible Regular Expressions (PCRE)*.
<http://www.pcre.org>.

- [21] Melanie Grandits, Axel Sündermann, and Chris Oostenbrink. “LUCI4HPC”. In: *Linux Journal* 252 (2015), pp. 68–76.
- [22] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E. Mark, and Herman J. C. Berendsen. “GROMACS: Fast, flexible, and free”. In: *Journal of Computational Chemistry* 26.16 (Dec. 2005), pp. 1701–1718. DOI: [10.1002/jcc.20291](https://doi.org/10.1002/jcc.20291).

MDplot - an R plotting package for molecular dynamics analysis

Adapted from:

Margreitter C., Oostenbrink C., *MDplot - an R plotting package for molecular dynamics analysis*, submitted to: The R Journal

Author contributions:

CM developed and maintains the package, CO supervised the work, suggested plot types and assisted in writing the manuscript.

The R programming language is a powerful tool, available free of charge and with a broad supporting community. Its reputation emerges mainly from its excellent applicability in terms of statistical and data analyses, but can also be used for (complex) plot generation and (partial) automation. It entails a complete programming language with strong focus on knowledge-sharing capabilities. Hence, R is an excellent choice for automated plot generation in the context of molecular dynamics simulations, a field where automation has a strong impact just because of the sheer amount of data generated. Usually, we do not show the information gained by our projects only in numbers, but also as graphical representations. This is considered to be more intuitive, not only for presentations but also because a e.g. thermodynamic integration figure shows immediately, how two curves match. Since the amount of data produced by our simulations is mainly dependent on our computational resources ("the more, the merrier"), and they are constantly growing, the number of figures to be generated and examined grows alike.

The R package MDplot helps to do so, by providing a set of (automated) plot functions for commonly used analyses. For GROMOS, specialized loading functions make it very easy to integrate even complex plotting tasks into automated setups such as bash scripts or PROMETHEUS.

7.1 Abstract

The MDplot package provides plotting functions to allow for automated visualisation of molecular dynamics simulation output. It is especially useful in cases where the plot generation is rather tedious due to complex file formats or when large amounts of plots are generated. The graphs that are supported range from those which are standard such as RMSD/RMSF down to thermodynamic integration analysis and hydrogen bond monitoring over time. In this chapter, we set out package's functions and give examples of the function calls and show the plots that emerge as a result thereof. Loading functions for the GROMOS force field simulation package are currently available, but users can easily integrate additional MD engines. Moreover, we also provide a bash interface that allows for simple integration of MDplot into bash scripts as a final analysis step.

7.2 Introduction

The amount of data produced by molecular dynamics (MD) engines (such as GROMOS [1, 2], GROMACS [3], NAMD [4], AMBER [5] and CHARMM [6]) has been constantly increasing over recent years. This is mainly due to more powerful and cheaper hardware and as a result of this both the lengths and the sheer number of MD-simulations (trajectories) increased enormously. Indeed, large sets of simulations (e.g. in the context of drug design) are attainable nowadays thus suggesting that the processing of the resulting information is undertaken automatically.

In this respect, the automated visualisation of molecular dynamics data would be highly advantageous: both in order to avoid repetitive tasks and to yield the ultimately desired result (see figure 7.1). Moreover, generating some of the graphs can be cumbersome. An example of this would be the time-series of a clustering program. Therefore these are predestined to be handled by a plotting library instead. There have been attempts made in that direction, for example the package `bio3d` [7] which allows the trajectories to be processed in terms of principle component analysis (PCA), RMSD and RMSF calculations. However, to the best of our knowledge there is currently no R package available that offers the wide range of plotting functions that is provided by the MDplot package presented in this paper. R seems the natural choice for this undertaking because of both its power in data handling and its vast plotting abilities.

In the following sections we outline all of the plotting functions that are currently supported. Examples of the function calls based on the data included in the package as well as the resulting plots and argument lists for each function are also examined in the course of the paper. The respective code samples use the loading functions (reported below) to load the input files which are located in folder `inst/extdata`. Currently, the package only supports GROMOS file formats as input to the loading functions since this is the MD software used by our group. However, by extending the loading functions we plan to include other engines as well.

Availability: The package can be obtained in the latest major version from CRAN (<https://cran.r-project.org/web/packages/MDplot>) or in the most recent version from the projects' GitHub page at <https://github.com/MDplot/MDplot>, where feedback also is gathered. MDplot is published under the GPL-3 licence.

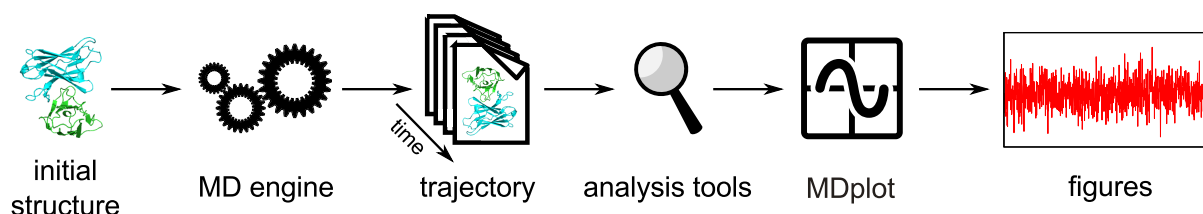


Figure 7.1: Shows the overall workflow typically applied in molecular dynamics simulations beginning with a single PDB structure as the input for a molecular dynamics simulation and ending with the graphical representation of the data obtained. For large amounts of data, generating figures might become a tedious, highly repetitive task.

7.3 Plotting functions

The package currently offers 14 distinct plotting functions (table 7.1), which cover many of the graphs that are commonly required. Although the focus of the package relies on the visualisation of data, in addition to this values are calculated to characterise the underlying data when appropriate. For example `TiCurve()` calculates the thermodynamic integration free-energy values including error estimates and the hysteresis between the integration curves. Examples of the call of the functions via a bash script are provided at the end of the document.

Table 7.1: Lists all of the currently available plotting functions that have been implemented in MDplot. All functions accept a boolean parameter (`barePlot`), that allows to print the plotting area only.

plot function	description
<code>clusters()</code>	summary of clustering over trajectories (RMSD based)
<code>clusters_ts()</code>	time-series of cluster populations (RMSD based)
<code>dssp_summary()</code>	secondary structure annotation plot (DSSP based)
<code>dssp_ts()</code>	time-series of secondary structure elements (DSSP based)
<code>hbond()</code>	hydrogen bonds summary plot
<code>hbond_ts()</code>	time-series of hydrogen bonds
<code>noe()</code>	Nuclear-Overhauser-Effect violation plot
<code>ramachandran()</code>	dihedral angle plot (including DISICL regions)
<code>rmsd()</code>	Root-Mean-Square-Deviation plot
<code>rmsd_average()</code>	Average Root-Mean-Square-Deviation plot
<code>rmsf()</code>	Root-Mean-Square-Fluctuation plot
<code>TiCurve()</code>	thermodynamic integration curves
<code>timeseries()</code>	general timeseries plot
<code>xrmsd()</code>	cross-RMSD plot (heatmap of RMSD values)

7.3.1 clusters()

Molecular dynamics simulation trajectories can be considered to be a set of atom configurations along the time axis. Clustering is applied in order to extract common structural features from these. The configurations are classified and grouped together based on the root mean square deviation (RMSD). These subsets of configurations around the cluster's central member structure and their relative occurrences allow for comparisons between different and within individual simulations. `clusters()` allows to plot a summary of all of the (selected) clusters over a set of trajectories (figure 7.2).

Listing 7.1: Example call of `clusters()`

```
1 clusters( load_clusters( "inst/extdata/clusters_example.txt",
2                       names=c( "wild-type", "mut1", "mut2",
3                               "mut3", "mut4", "mut5" )),
4          clustersNumber=9, main="MDplot::clusters()", ylab="# configurations")
```

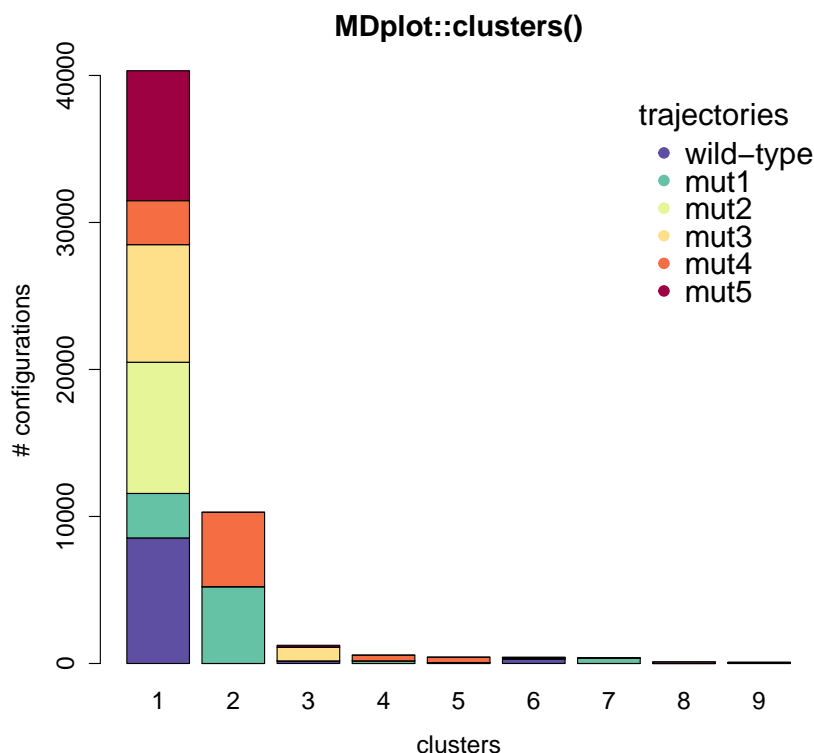


Figure 7.2: The clusters are plotted along the x-axis and the number of configurations for each trajectory for every cluster on the y-axis. The number of clusters is limited in this example to nine, which is useful as many scarcely populated clusters can be omitted.

argument name	default value	description
clusters	<i>none</i>	Matrix with clusters. Trajectories are given in rows, clusters in columns.
clustersNumber	NA	Maximum number of clusters to be shown.
legendTitle	"trajectories"	The title of the legend.
barePlot	FALSE	Boolean, indicating whether the plot is to be made without any additional information or not.
...	<i>none</i>	Additional arguments.

7.3.2 clusters_ts()

In order to monitor the time-dependence of the aforementioned clustering, we can use the function `clusters_ts()`. This plot consists of two parts: the top sub-plot represents the overall distribution whilst the lower sub-plot shows the development over time. In the event that not all clusters are selected, the non-selected clusters appear as white bars in the time series. The time axis may be shown in nanoseconds (see figure 7.3 for an example).

Listing 7.2: Example call of `clusters_ts()`

```

1 clusters_ts(load_clusters_ts("inst/extdata/clusters_ts_example.txt.gz",
2                             lengths=c(4000, 4000, 4000, 4000, 4000, 4000),
3                             names=c("wild-type", "mut1", "mut2",
4                                       "mut3", "mut4", "mut5")),
5         clustersNumber=7, main="MDplot::clusters_ts() example",
6         timeUnit="ns", snapshotsPerTimeInt=100)
```

argument name	default value	description
clustersDataTS	<i>none</i>	List of cluster information as provided by <code>load_clusters_ts()</code> , the associated loading function.
clustersNumber	NA	Integer, specifying the number of clusters that is plotted.
selectTraj	NA	Vector of indices of trajectories that are plotted (as given in the input file).
selectTime	NA	Range of time in snapshots.
timeUnit	NA	Abbreviation of time unit.
snapshotsPerTimeInt	1000	Specifies, how many snapshots make up one time unit.
...	<i>none</i>	Additional arguments.

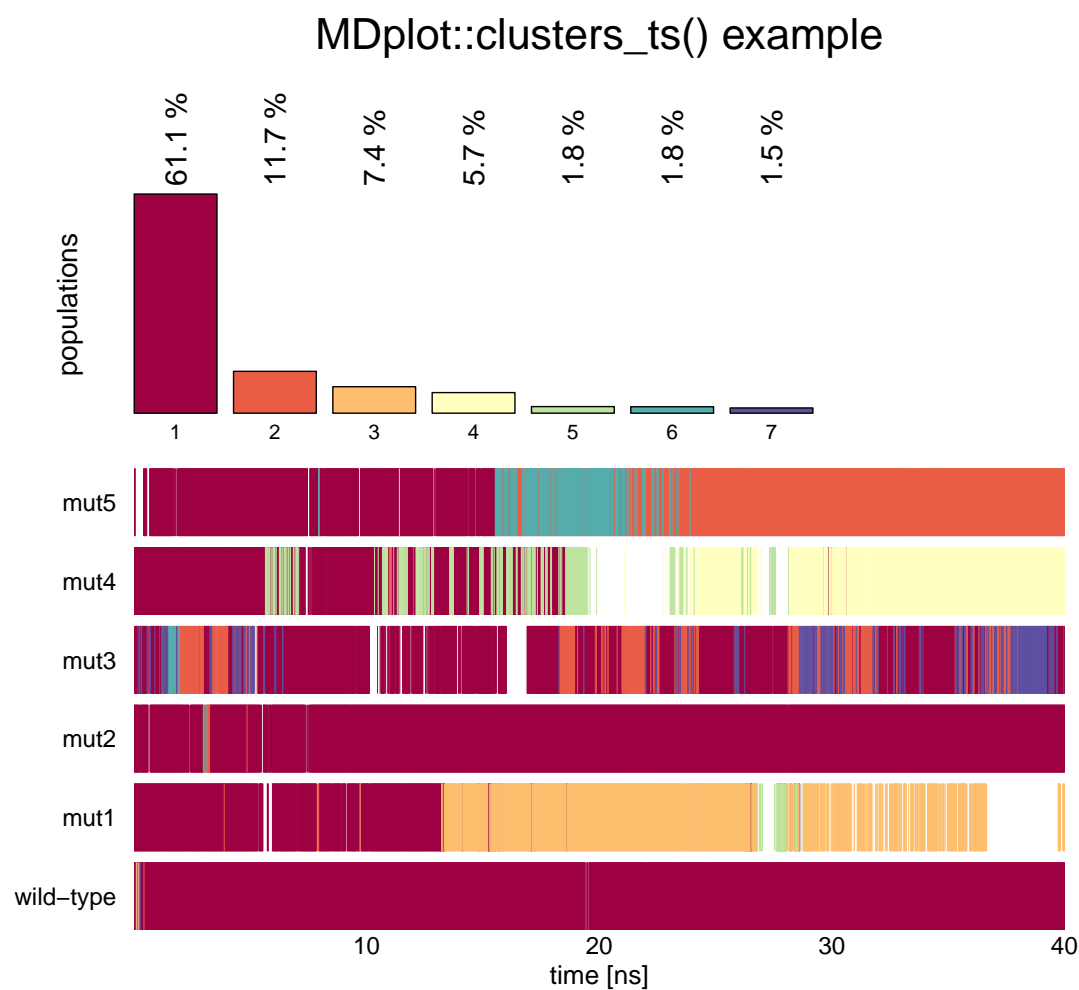


Figure 7.3: The plot shows a selection of the seven most populated clusters for six trajectories. The width of the bars in the top sub-plot is adjusted according to their number.

7.3.3 dssp_summary()

In terms of proteins the secondary structure can be annotated by the widely used program DSSP (Definition of Secondary Structure of Proteins) [8]. This algorithm uses the backbone hydrogen bond pattern in order to assign secondary structure elements such as α -helices, β -strands, and turns to protein sequences. The plotting function `dssp_summary()` has three different visualization methods and plots the overall result over the trajectory and over the residues. The user can specify selections of residues and elements which should be taken into consideration (figure 7.4).

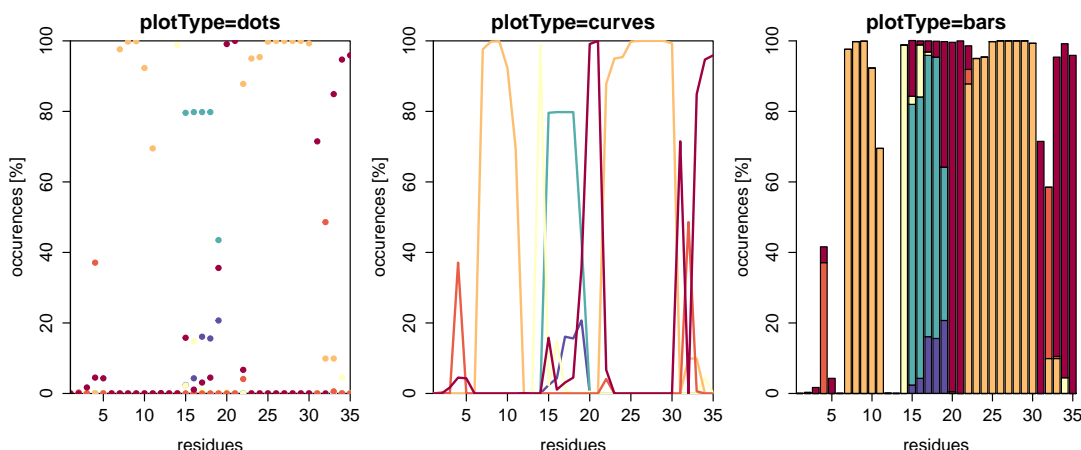


Figure 7.4: Example with `plotType` set to "dots" (default), "curves" or "bars". Note that the fractions do not necessarily sum up to a hundred percent, because some residues might not be in defined secondary structure elements all the time. In this figure, there is no legend plotted due to space limitations (see figure 7.5 for a colour-code explanation).

Listing 7.3: Example call of `dssp_summary()`

```

1 layout(matrix(1:3, nrow=1), widths=c(0.33,0.33,0.33))
2 dssp_summary(load_dssp_summary("inst/extdata/dssp_summary_example.txt.gz"),
3             main="plotType=dots", showResidues=c(1,35))
4 dssp_summary(load_dssp_summary("inst/extdata/dssp_summary_example.txt.gz"),
5             main="plotType=curves", plotType="curves", showResidues=c(1,35))
6 dssp_summary(load_dssp_summary("inst/extdata/dssp_summary_example.txt.gz"),
7             main="plotType=bars", plotType="bars", showResidues=c(1,35))

```

argument name	default value	description
dsspData	<i>none</i>	Table containing information on the secondary structure elements.
printLegend	FALSE	If TRUE, a legend is printed on the right hand side of the plot.
useOwnLegend	FALSE	If FALSE, the names of the secondary structure elements are considered to be in default order.
elementNames	NA	Vector of names for the secondary structure elements.
colours	NA	A vector of colours that can be specified to replace the default ones.
showValues	NA	A vector of boundaries for the values.
showResidues	NA	A vector of boundaries for the residues.
plotType	"dots"	Either "dots", "curves" or "bars".
selectedElements	NA	A vector of names of the elements selected.
barePlot	FALSE	Boolean, indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.4 dssp_ts()

The secondary structure information as described for the function `dssp_summary()` can also be visualised along the time axis using the function `dssp_ts()` (figure 7.5). The time can be annotated in snapshots or time units (e.g. nanoseconds).

Listing 7.4: Example call of `dssp_ts()`

```
1 dssp_ts(load_dssp_ts("inst/extdata"), printLegend=TRUE,
2         main="MDplot::dssp_ts()", timeUnit="ns",
3         snapshotsPerTime=1000)
```

argument name	default value	description
<code>tsData</code>	<i>none</i>	List consisting of lists, which are composed of a name and a values table (x ... snapshots, y ... residues). This can be generated by <code>load_dssp_ts()</code> .
<code>printLegend</code>	TRUE	If TRUE, a legend is printed on the right hand side of the plot.
<code>timeBoundaries</code>	NA	A vector of boundaries for the time in snapshots.
<code>residueBoundaries</code>	NA	A vector of boundaries for the residues.
<code>timeUnit</code>	NA	If set, the snapshots are transformed into the respective time (depending on parameter <code>snapshotsPerTime</code>).
<code>snapshotsPerTimeInt</code>	1000	Number of snapshots per respective timeUnit.
<code>barePlot</code>	FALSE	Boolean, indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

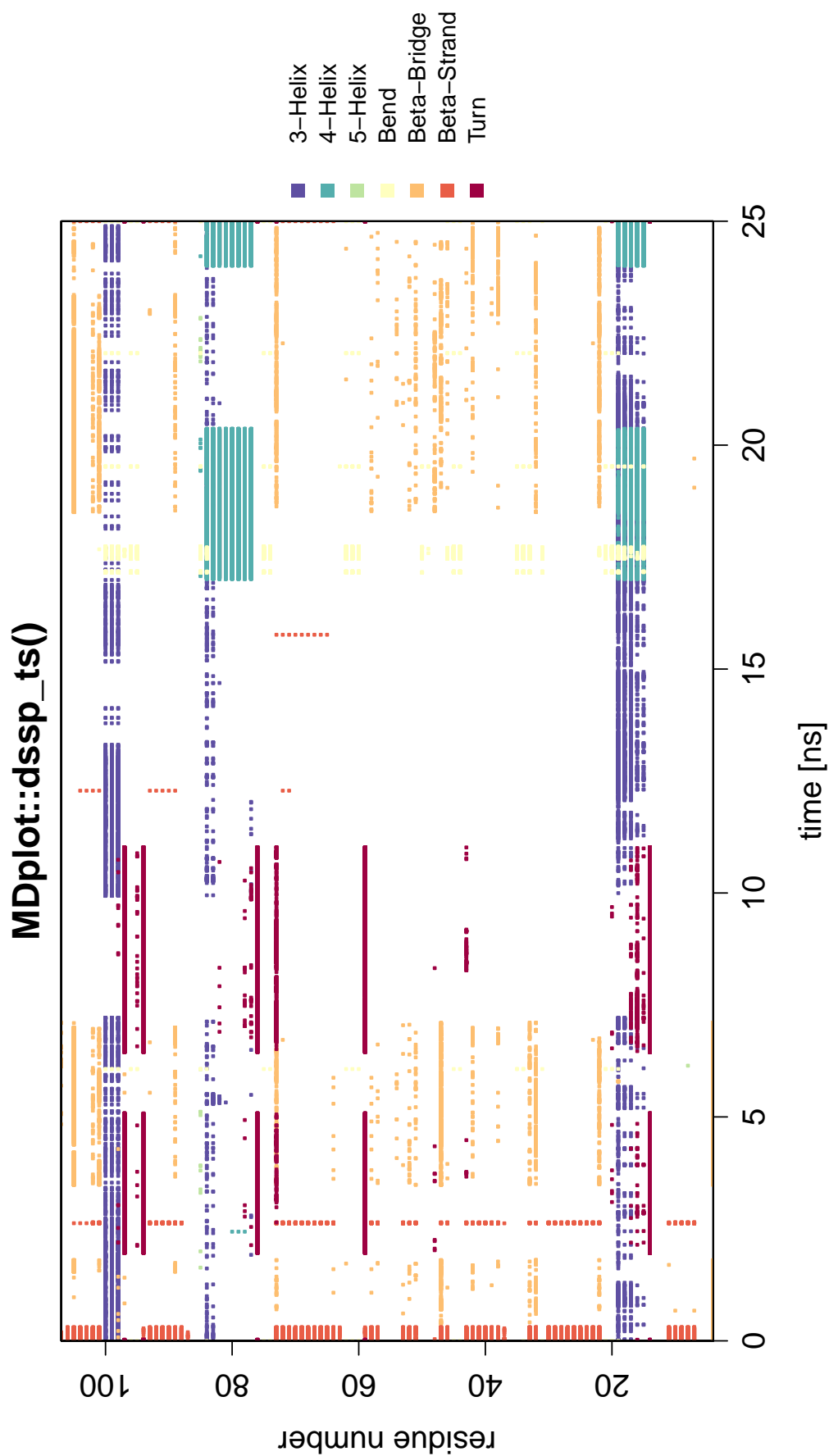


Figure 7.5: Example showing all of the defined secondary structure elements per residue over time. Note, that for this example plot a sparse data set was used to reduce the size of the example data file.

7.3.5 hbond()

In the context of biomolecules, hydrogen bonds are of particular importance. These bonds take place between a donor, a hydrogen and an acceptor atom. The `hbond()` plotting function allows us to plot the occurrence of hydrogen bonds over the whole trajectory indicating their overall occurrences using an array of different colours (see figure 7.6). It is possible to achieve selections of subsets using both the identifiers as well as the acceptor and donor atoms.

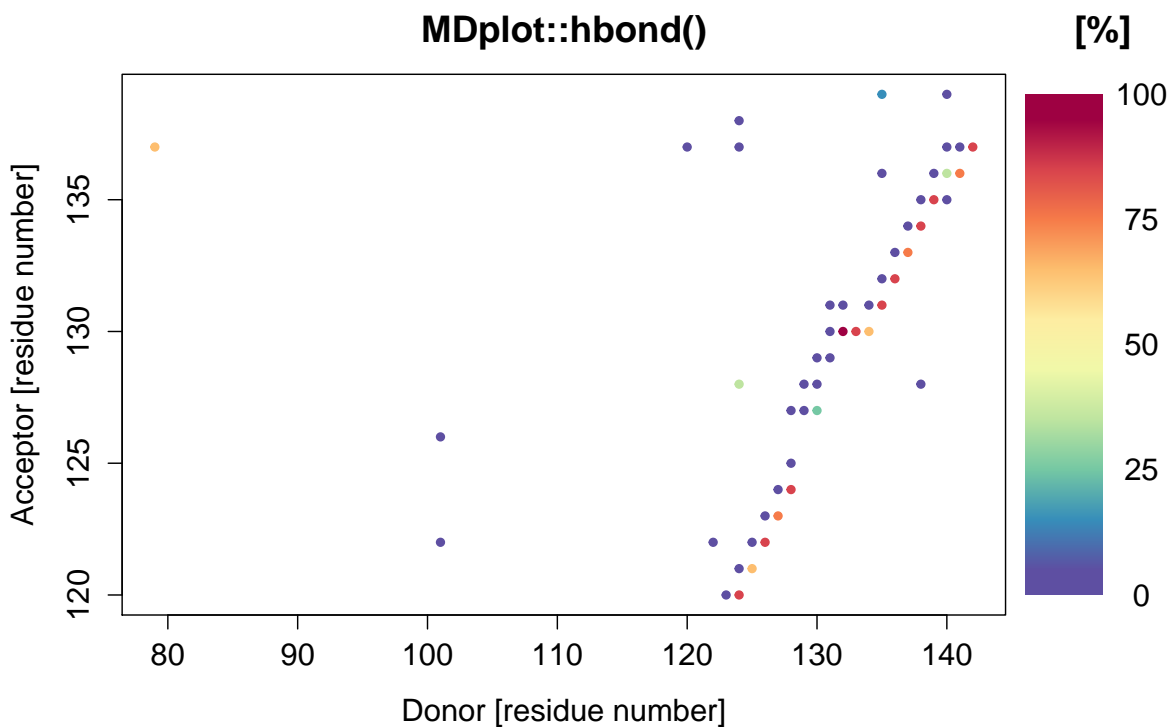


Figure 7.6: The acceptor residues are plotted on the x-axis whilst the donors are shown on the y-axis. The different colours indicate the occurrences throughout the whole trajectory.

Listing 7.5: Example call of `hbond()`

```
1 hbond(load_hbond("inst/extdata/hbond_example.txt.gz"),  
2       main="MDplot::hbond()", acceptorRange=c(120,140))
```

argument name	default value	description
hbonds	<i>none</i>	Table containing the hydrogen bond information in columns "hbondID", "resDonor", "resDonorName", "resAcceptor", "resAcceptorName", "atomDonor", "atomDonorName", "atomH", "atomAcceptor", "atomAcceptorName", "percentage" (automatically generated by function <code>load_hbond()</code>).
plotMethod	"residue-wise"	Allows to set the detail of hbond information displayed. Options are: "residue-wise".
acceptorRange	NA	Vector, specifying the range of acceptor residues.
donorRange	NA	Vector, specifying the range of donor residues.
printLegend	TRUE	Boolean, enabling the legend.
barePlot	FALSE	Boolean, indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.6 hbond_ts()

The timeseries of hydrogen bond occurrences can be visualised using the function `hbond_ts()`, which plots them either according to their identifiers or in a human readable form in three or one letter code (the participating atoms can be shown as well) on the y-axis and the time on the x-axis. If the GROMOS input format is used, this function requires two different files: the summary of the hbond program and the timeseries file. The occurrence of a hydrogen bond is represented by a black bar and the occurrence summary can be added on the right hand side as a sub-plot (figure 7.7).

Listing 7.6: Example call of `hbond_ts()`

```

1 hbond_ts(timeseries=load_hbond_ts("inst/extdata/hbond_ts_example.txt.gz"),
2         summary=load_hbond("inst/extdata/hbond_example.txt.gz"),
3         main="MDplot::hbond_ts()", acceptorRange=c(22,75),
4         hbondIndices=list(c(0,24)), plotOccurrences=TRUE, timeUnit="ns",
5         snapshotsPerTimeInt=100, printNames=TRUE, namesToSingle=TRUE,
6         printAtoms=TRUE)

```

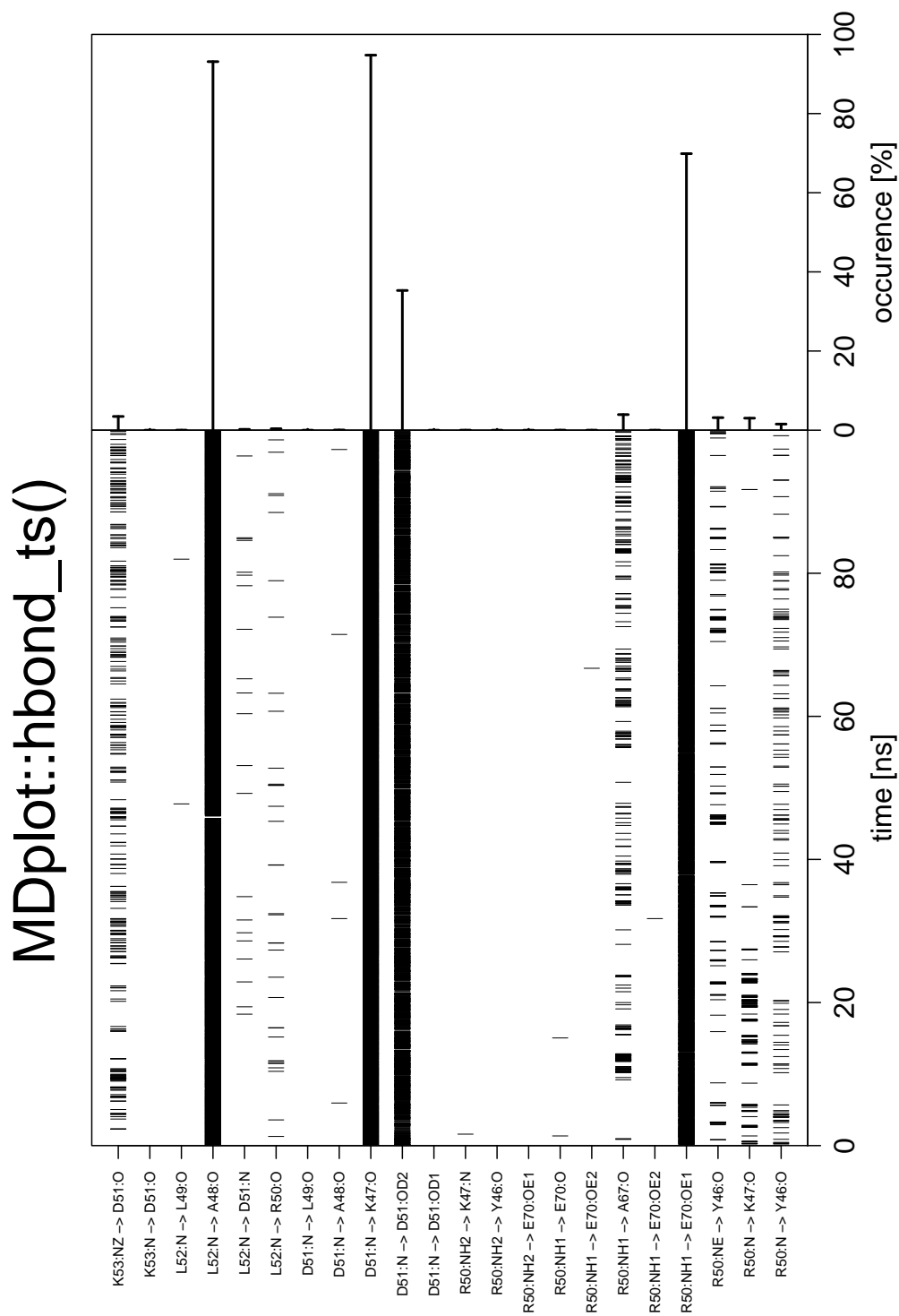


Figure 7.7: Example figure generated by `hbond_ts()` for both an identifier and acceptor residues' selection. The labels for the hydrogen bonds may be printed as identifiers or with names composed of residue names (in single- or three-letter code) and the names of the participating atoms.

argument name	default value	description
<code>timeseries</code>	<i>none</i>	Table containing the timeseries information (e.g. provided by <code>load_hbond_ts()</code>).
<code>summary</code>	<i>none</i>	Table containing the summary information (e.g. provided by <code>load_hbond()</code>).
<code>acceptorRange</code>	NA	Vector of acceptor residues.
<code>donorRange</code>	NA	Vector of donor residues.
<code>plotOccurences</code>	FALSE	Specifies whether the overall summary should be plotted.
<code>scalingFactorPlot</code>	NA	Overwrites the scaling factor of the bars (if necessary).
<code>printNames</code>	FALSE	Enables human readable names rather than the hydrogen bond identifiers.
<code>namesToSingle</code>	FALSE	If <code>printNames</code> is TRUE, this flags enables one letter codes.
<code>printAtoms</code>	FALSE	Enables atom names in hydrogen bond identification on the y-axis.
<code>timeUnit</code>	NA	Specifies the time unit on the x-axis.
<code>snapshotsPerTimeInt</code>	1000	Specifies how many snapshots make up one time unit (see above).
<code>timeRange</code>	NA	Selects a certain time range specified by a vector.
<code>hbondIndices</code>	NA	A list containing vectors to select hbonds by their identifiers.
<code>barePlot</code>	FALSE	Boolean indicating whether the plot is to be made without any additional information.
<code>...</code>	<i>none</i>	Additional arguments.

7.3.7 noe()

The Nuclear-Overhauser-Effect is one of the most important measures of structure validity in the context of molecular dynamics simulations. These interactions are transmitted through space and arise from spin-spin coupling, which can be measured by nuclear magnetic resonance (NMR) spectroscopy. These measurements provide pivotal distance restraints which should be matched on average during molecular dynamics simulation of the same system. The plotting function `noe()` allows to visualize the number of distance restraint violations and their respective spacial deviation. As shown in the example figure 7.8, multiple replicates or different protein systems are supported simultaneously.

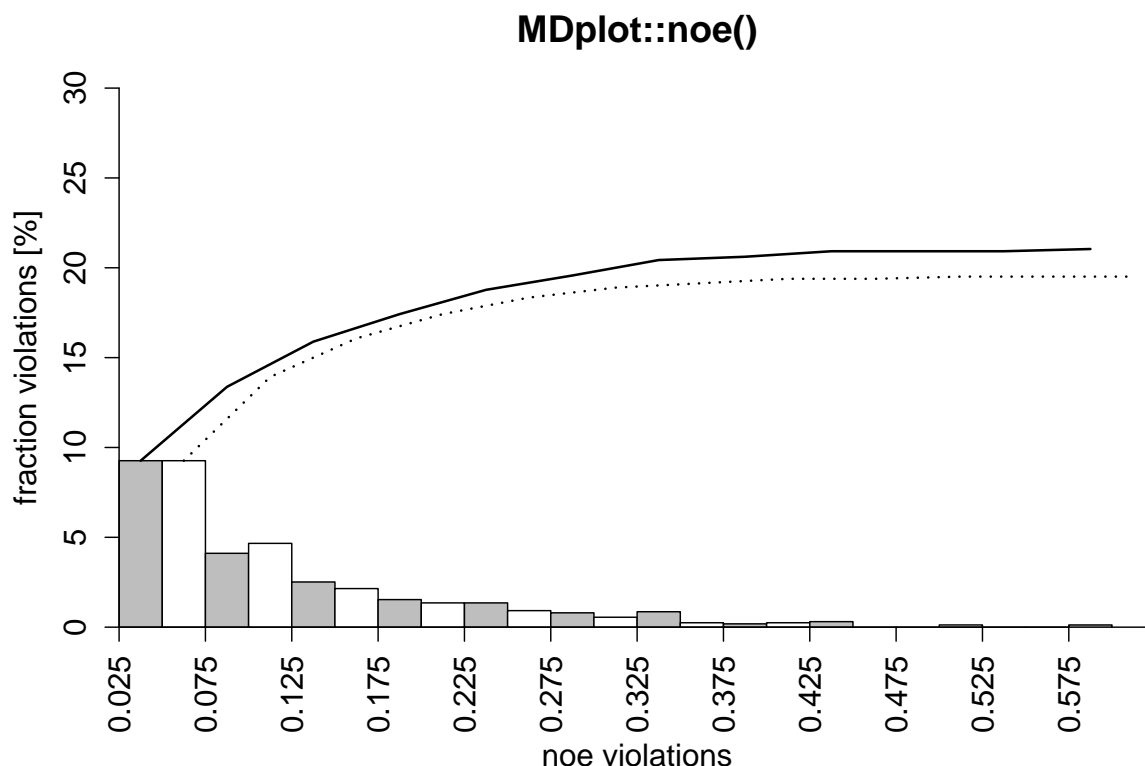


Figure 7.8: Example plot showing two different replicates of a protein simulation (they share the molecule, but have different initial velocities). Note, that the maximum value (x-axis) over all replicates is used for the plot. The sum over all violations from left to right is shown by an additional curve on top. The number of violations may be given as fractions (in %), as shown above, or absolute numbers (flag `printPercentages` either `TRUE` or `FALSE`).

Listing 7.7: Example call of `noe()`

```

1 noe(load_noe(files=c("inst/extdata/noe1_example.txt.gz",
2                     "inst/extdata/noe2_example.txt.gz")),
3     printPercentages=TRUE, plotSumCurves=TRUE, maxYAxis=30, main="MDplot::noe()")

```

argument name	default value	description
<code>noeData</code>	<i>none</i>	Input list holding the NOE information as e.g. provided by <code>load_noe()</code> .
<code>printPercentages</code>	NA	Boolean, determining whether the violations are reported in absolute numbers or relative fractions (in %).
<code>lineTypes</code>	NA	Vector, specifying which kinds of line-types should be used for the summation curves.
<code>names</code>	NA	Vector of names describing the respective input files.
<code>plotSumCurves</code>	TRUE	Boolean, which enables summation curve plotting if TRUE.
<code>maxYAxis</code>	NA	Upper boundary for the plotting frame y-axis.
<code>printLegend</code>	FALSE	Boolean, enabling the legend.
...	<i>none</i>	Additional arguments.

7.3.8 ramachandran()

This graph type [9] is often used to show the sampling of the ϕ/ψ protein backbone dihedral angles in order to assign propensities of secondary structure elements to the protein of interest. The function `ramachandran()` offers a 2D (figure 7.9) and 3D (figure 7.10) variant with the former offering the possibility to print user-defined secondary structure regions as well. The number of bins for the two axes and the colours used for the legend can be specified by the user.

Listing 7.8: Example call of `ramachandran()`

```

1 ramachandran(load_ramachandran("inst/extdata/ramachandran_example.txt.gz"),
2             heatFun="log", plotType="sparse", xBins=90, yBins=90,
3             main="ramachandran() (plotType=sparse)",
4             plotContour=TRUE)

```

Listing 7.9: Example call of `ramachandran()`

```

1 ramachandran(load_ramachandran("inst/extdata/ramachandran_example.txt.gz"),
2             heatFun="norm", plotType="fancy", xBins=90, yBins=90,
3             main="ramachandran() (plotType=fancy)",
4             printLegend=TRUE)

```

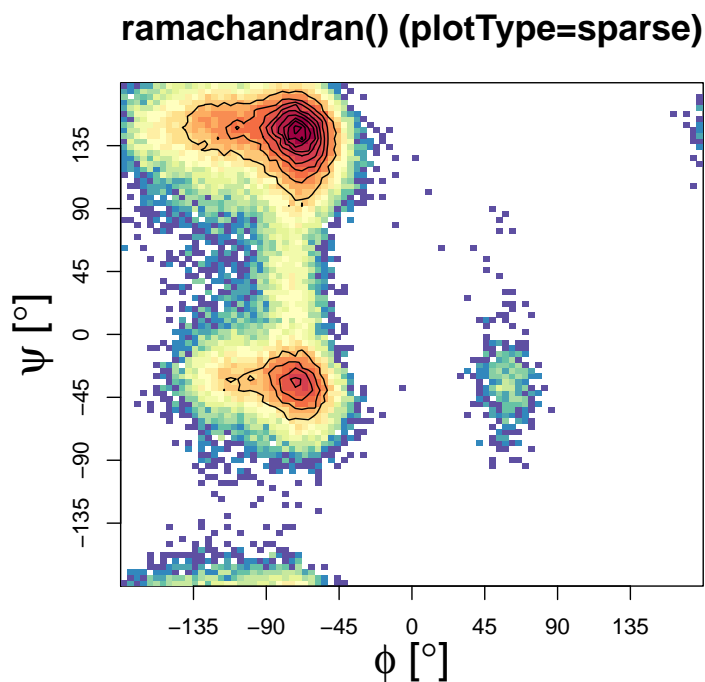


Figure 7.9: Two dimensional plot version "sparse" of the `ramachandran()` function with enabled contour plotting. The number of bins in which the dihedrals are grouped can be specified independently.

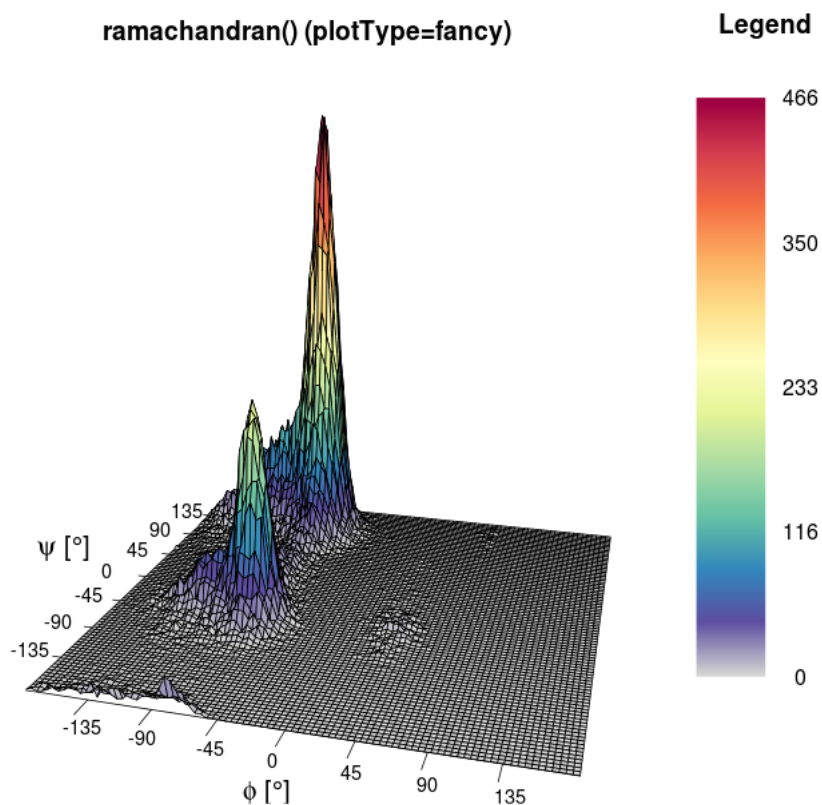


Figure 7.10: Three dimensional example of `ramachandran()`. In addition to the colour, the height (z-axis) also represents the number of dihedrals per bin.

argument name	default value	description
dihedrals	<i>none</i>	Matrix with angles (two columns).
xBins	150	Number of bins used to plot (on the x-axis).
yBins	150	Number of bins used to plot (on the y-axis).
heatFun	"norm"	Function selector for calculation of the colour. The possibilities are either: "norm" for linear calculation or "log" for logarithmic calculation.
structureAreas	<code>c()</code>	Vector of secondary structure areas. By default there are no regions specified.
plotType	"sparse"	Type of plot to be used, either "sparse" or "fancy". The latter is a 3D representation.
printLegend	FALSE	Enables heat legend.
plotContour	FALSE	Boolean specifying whether a contour should be added or not.
barePlot	FALSE	Boolean indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.9 rmsd()

The atom-positional root-mean-square deviation (RMSD) is one of the most commonly used plot types in the field of biophysical simulations. In the context of atom configurations, it is a measure for the positional divergence of one or multiple atoms. The input requires a timeseries in column one and a series of RMSD values in column two. There may be multiple data sets plotted, given in separate input files. Figure 7.11 shows an example for two trajectories.

Listing 7.10: Example call of `rmsd()`

```
1 rmsd(load_rmsd(c("inst/extdata/rmsd1_example.txt.gz",  
2                  "inst/extdata/rmsd2_example.txt.gz")),  
3       printLegend=TRUE, names=c("WT", "mut"), main="MDplot::rmsd()")
```

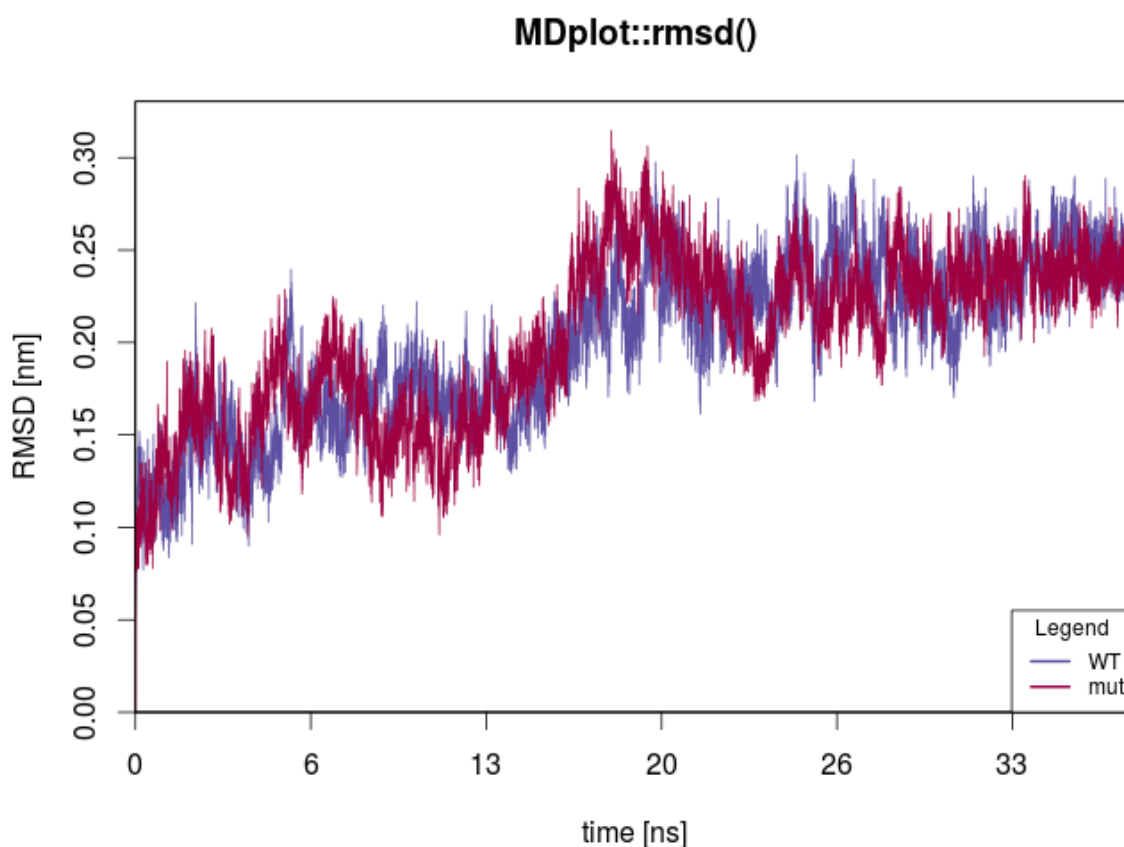


Figure 7.11: This plot shows the RMSD curves for two different trajectories. The time is given in nanoseconds, which requires a properly set `factor` parameter.

argument name	default value	description
rmsdData	<i>none</i>	List of (alternating) indices and RMSD values, as produced by <code>load_rmsd()</code> .
printLegend	TRUE	Boolean which triggers the plotting of the legend.
factor	1000	Number, specifying how many snapshots are within one <code>timeUnit</code> .
timeUnit	"ns"	Specifies the time unit.
rmsdUnit	"nm"	Specifies the RMSD unit.
colours	NA	Vector of colours used for plotting.
names	NA	Vector holding the names of the trajectories.
legendPosition	"bottomright"	Indicates the position of the legend: either "bottomright", "bottomleft", "topleft" or "topright".
barePlot	FALSE	Boolean, indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.10 rmsd_average()

Nowadays, for many molecular systems multiple replicates of simulations are performed in order to enhance the sampling of the phase space. However, since analysis data grows accordingly, a joint representation of the results may be desirable. For the case of backbone-atom and other RMSD plots, the MDplot package provides an average plotting function. Instead of plotting every curve individually, the mean and the minimum and maximum values of all trajectories at a given time point is plotted. Thus, the spread of multiple simulations over time is represented.

Listing 7.11: Example call of `rmsd_average()`

```

1 rmsd_average(list(load_rmsd("inst/extdata/rmsd1_example.txt.gz"),
2                   load_rmsd("inst/extdata/rmsd2_example.txt.gz")),
3               maxYAxis=0.375, main="MDplot:rmsd_average()")

```

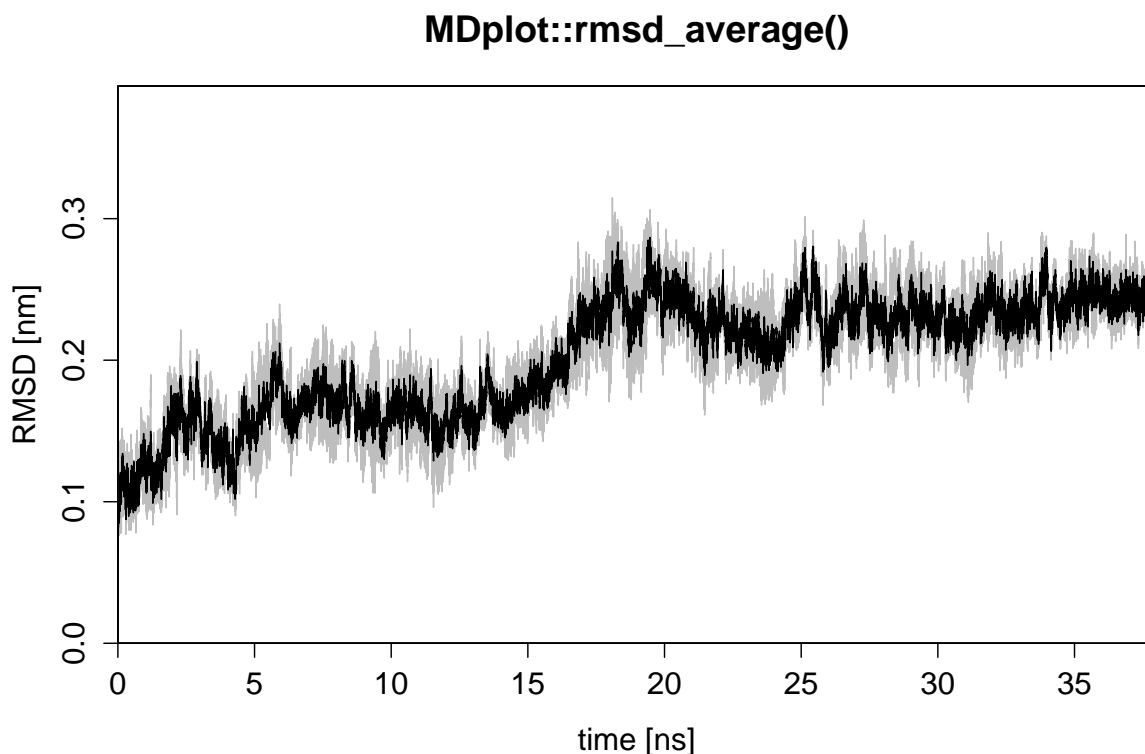


Figure 7.12: In black, the mean RMSD value at a given timepoint is given and in grey the respective minimum and maximum values. In this example, two rather similar curves have been used.

argument name	default value	description
rmsdInput	<i>none</i>	List of snapshot and RMSD value pairs, as e.g. provided by loading function <code>load_rmsd()</code> .
printMeans	FALSE	Boolean, enabling the printing of the calculated mean values per snapshot to the standard output.
snapshotsPerTimeInt	1000	Number of snapshots per time interval.
timeUnit	"ns"	Specifies the time unit.
rmsdUnit	"nm"	Specifies the RMSD unit.
maxYAxis	NA	Maximum y-axis value.
barePlot	FALSE	Boolean, indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.11 rmsf()

The atom-positional root-mean-square fluctuation (RMSF) represents the degree of positional variation of a given atom over time. The input requires one column with all residues or atoms and a second one holding RMSF values. Figure 7.13 shows, as an example the RMSF of the first 75 atoms, calculated for two independent simulations.

Listing 7.12: Example call of `rmsf()`

```
1 rmsf(load_rmsf(c("inst/extdata/rmsf1_example.txt.gz",
2                 "inst/extdata/rmsf2_example.txt.gz")),
3       printLegend=TRUE, names=c("WT", "mut"), range=c(1, 75),
4       main="MDplot::rmsf()")
```

argument name	default value	description
<code>rmsfData</code>	<i>none</i>	List of (alternating) atom numbers and RMSF values, as e.g. produced by <code>load_rmsf()</code> .
<code>printLegend</code>	TRUE	Boolean which triggers plotting of the legend.
<code>rmsfUnit</code>	"nm"	Specifies the RMSF unit.
<code>colours</code>	NA	Vector of colours used for plot.
<code>numberXLabels</code>	7	Specifies how many ticks are used on the x-axis.
<code>names</code>	NA	Vector of the names of the trajectories.
<code>range</code>	NA	Range of residues.
<code>legendPosition</code>	"topright"	Indicates position of legend: either "bottom-right", "bottomleft", "topleft" or "topright".
<code>barePlot</code>	FALSE	Boolean indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

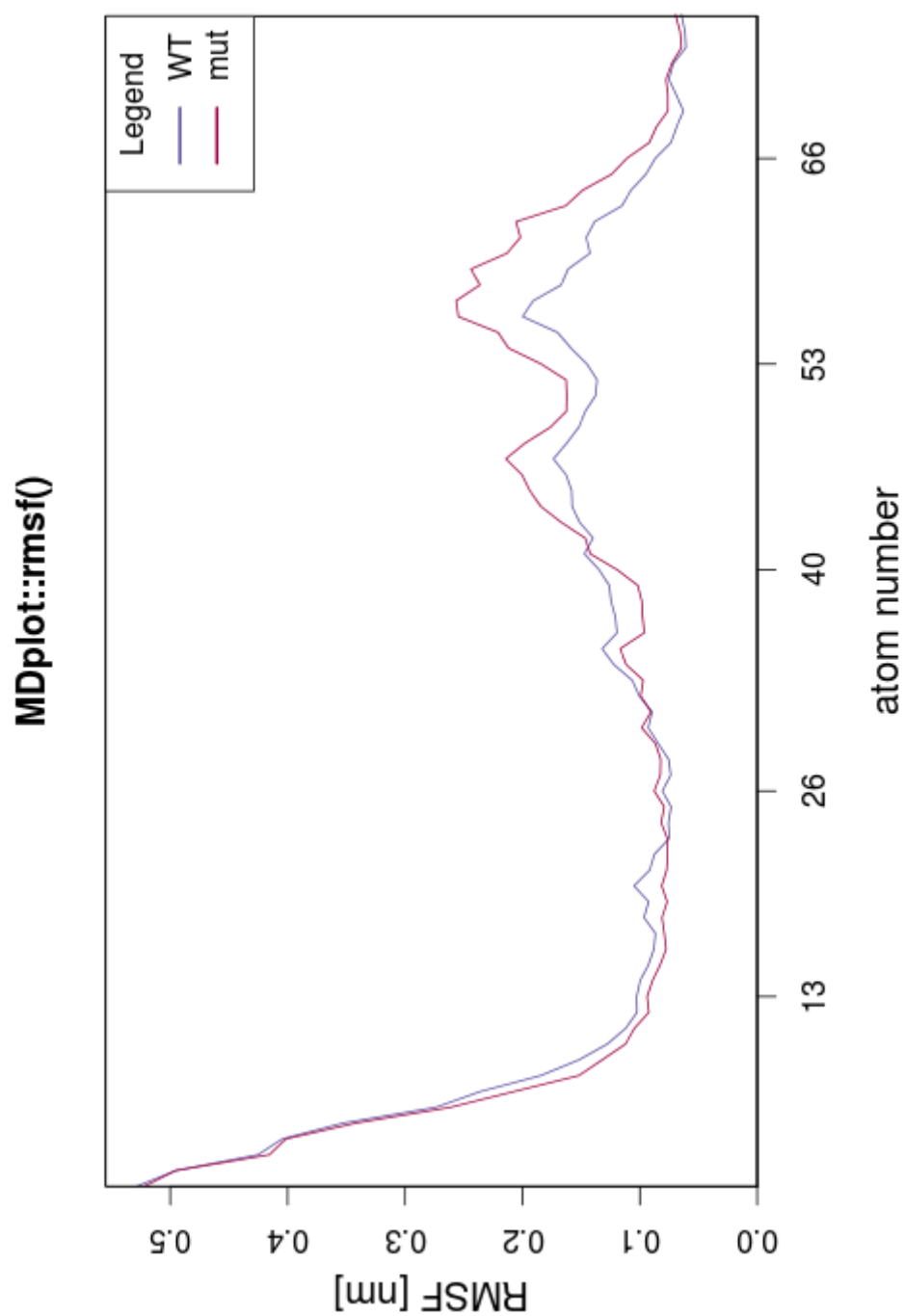


Figure 7.13: Plot showing two different RMSF curves.

7.3.12 TIcurve()

For calculations of the free energy when transforming one chemical compound into another (alchemical changes) or for estimates of the free energy changes upon binding, thermodynamic integration [10] is one of the most trusted and applied approaches. The derivative of the Hamiltonian as a function of a coupling parameter λ is calculated over a series of λ state points (typically around 15). The integral of this curve is equivalent to the change in free energy (figure 7.14). The function `TIcurve()` performs the integration and, if the data for both the forward and backward processes is provided, the hysteresis between them.

Listing 7.13: Example call of `TIcurve()`

```
1 TIcurve(load_TIcurve(c("inst/extdata/TIcurve_fb_forward_example.txt.gz",
2                       "inst/extdata/TIcurve_fb_backward_example.txt.gz")),
3         invertedBackwards=TRUE, main="MDplot::TIcurve()")
```

argument name	default value	description
<code>lambdas</code>	<i>none</i>	List of matrices (automatically generated by <code>load_TIcurve()</code>) holding the thermodynamic integration information.
<code>invertedBackwards</code>	<code>FALSE</code>	If a forward and backward TI are provided and the lambda points are enumerated reversely (i.e. 0.3 of one TI is equivalent to 0.7 of the other), this flag can be set to be <code>TRUE</code> in order to automatically mirror the values appropriately.
<code>energyUnit</code>	<code>"kJ/mol"</code>	Defines the energy unit used for the plot.
<code>printValues</code>	<code>TRUE</code>	If <code>TRUE</code> , the free energy values are printed on the plot.
<code>barePlot</code>	<code>FALSE</code>	Boolean indicating whether the plot is to be made without any additional information.
<code>...</code>	<i>none</i>	Additional arguments.

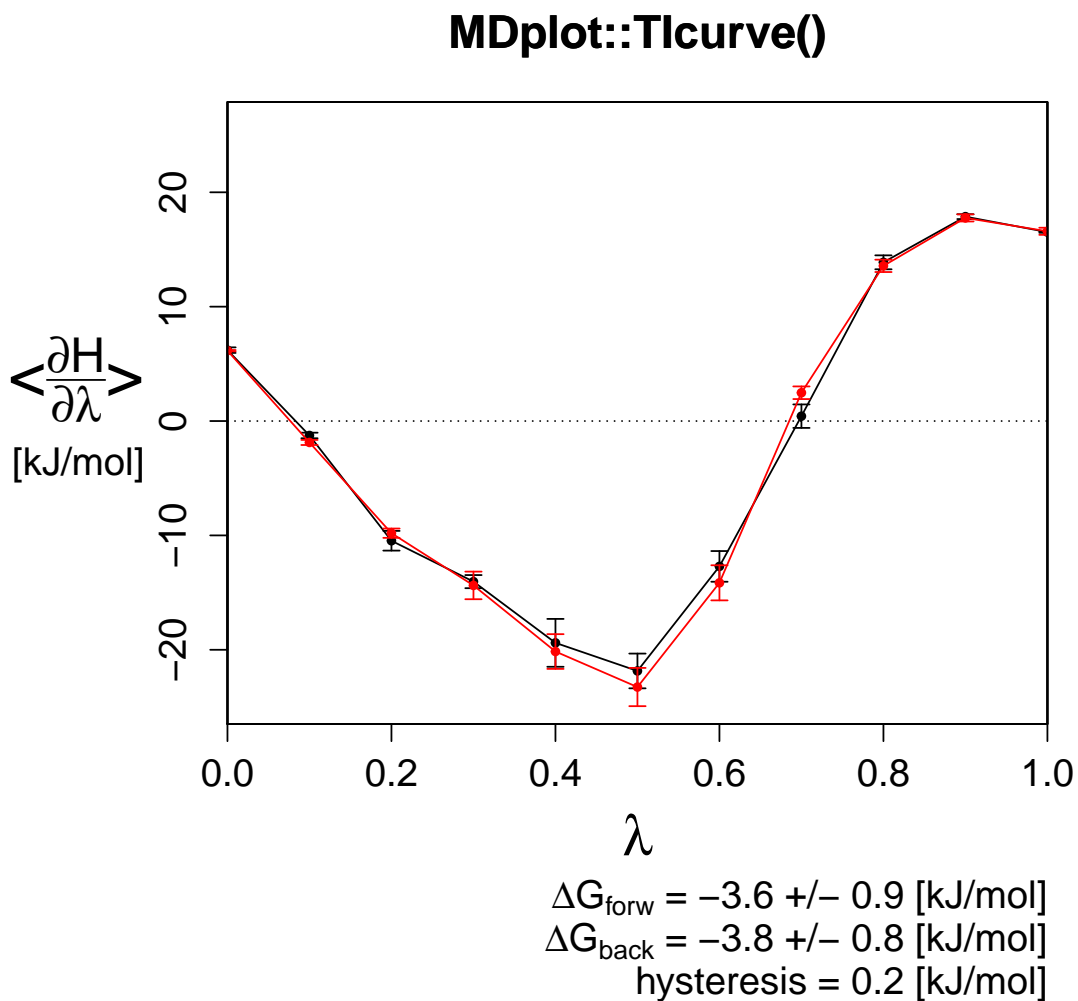


Figure 7.14: A forward and backward thermodynamic integration curve with the resulting hysteresis between them (precision as permitted by the error).

7.3.13 timeseries()

This function provides a general interface for any timeseries given as a time-value pair (figure 7.15).

Listing 7.14: Example call of `timeseries()`

```

1 timeseries(load_timeseries("inst/extdata/timeseries_example.txt.gz"),
2             colours=c("red"), valueName="energy", valueUnit="kJ/mol",
3             main="MDplot::timeseries()", names=c("curve"),
4             legendPosition="topright", snapshotsPerTimeInt=10 )

```

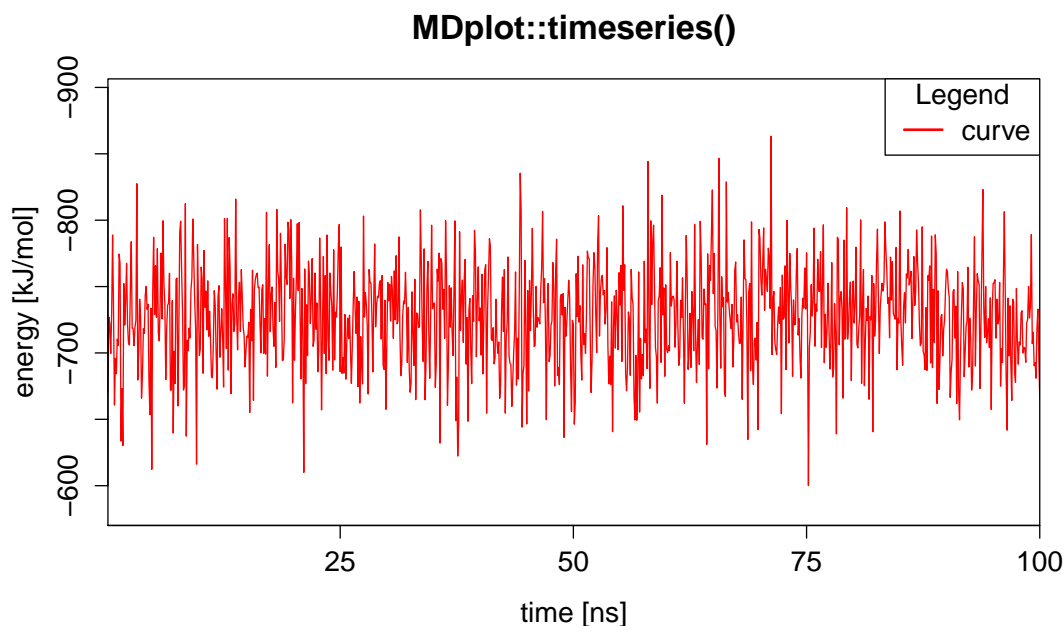


Figure 7.15: Shows a timeseries with parameter `snapshotsPerTimeInt` set in such a way, that the proper time in nanoseconds is plotted. In addition, the legend has been moved to the top-right position.

argument name	default value	description
<code>tsData</code>	<i>none</i>	List of (alternating) indices and response values, as produced by <code>load_timeseries()</code> .
<code>printLegend</code>	TRUE	Parameter enabling the plotting of the legend.
<code>snapshotsPerTimeInt</code>	1000	Number, specifying how many snapshots make up one <code>timeUnit</code> .
<code>timeUnit</code>	"ns"	Specifies the time unit.
<code>valueName</code>	NA	Name of response variable.
<code>valueUnit</code>	NA	Specifies the response variable's unit.
<code>colours</code>	NA	Vector of colours used for plotting.
<code>numberXLabels</code>	5	Specifies how many ticks are used on the x-axis.
<code>names</code>	NA	Vector of the names of the trajectories.
<code>legendPosition</code>	"bottomright"	Indicates position of legend: either "bottomright", "bottomleft", "topleft" or "topright".
<code>barePlot</code>	FALSE	Boolean indicating whether the plot is to be made without any additional information.
...	<i>none</i>	Additional arguments.

7.3.14 xrmsd()

This function generates a plot which shows a heat-map of the atom positional root mean square differences between snapshots (figure 7.16). The structures are listed on the x- and y-axis. The heat-map shows the difference between one structure and another using a coloured bin. The legend is adapted in accordance to the size of the values.

argument name	default value	description
xrmsdValues	<i>none</i>	Input matrix (three rows: x-values, y-values, RMSD-value). This can be generated by function <code>load_xrmsd()</code> .
printLegend	TRUE	If TRUE a legend is printed on the right hand side of the plot.
xaxisRange	NA	A vector of boundaries for the x-snapshots.
yaxisRange	NA	A vector of boundaries for the y-snapshots.
colours	NA	User specified vector of colours to be used for plotting.
...	<i>none</i>	Additional arguments.

Listing 7.15: Example call of `xrmsd()`

```
1 xrmsd(load_xrmsd("inst/extdata/xrmsd_example.txt.gz"),
2       printLegend=TRUE, main="MDplot::xrmsd()")
```

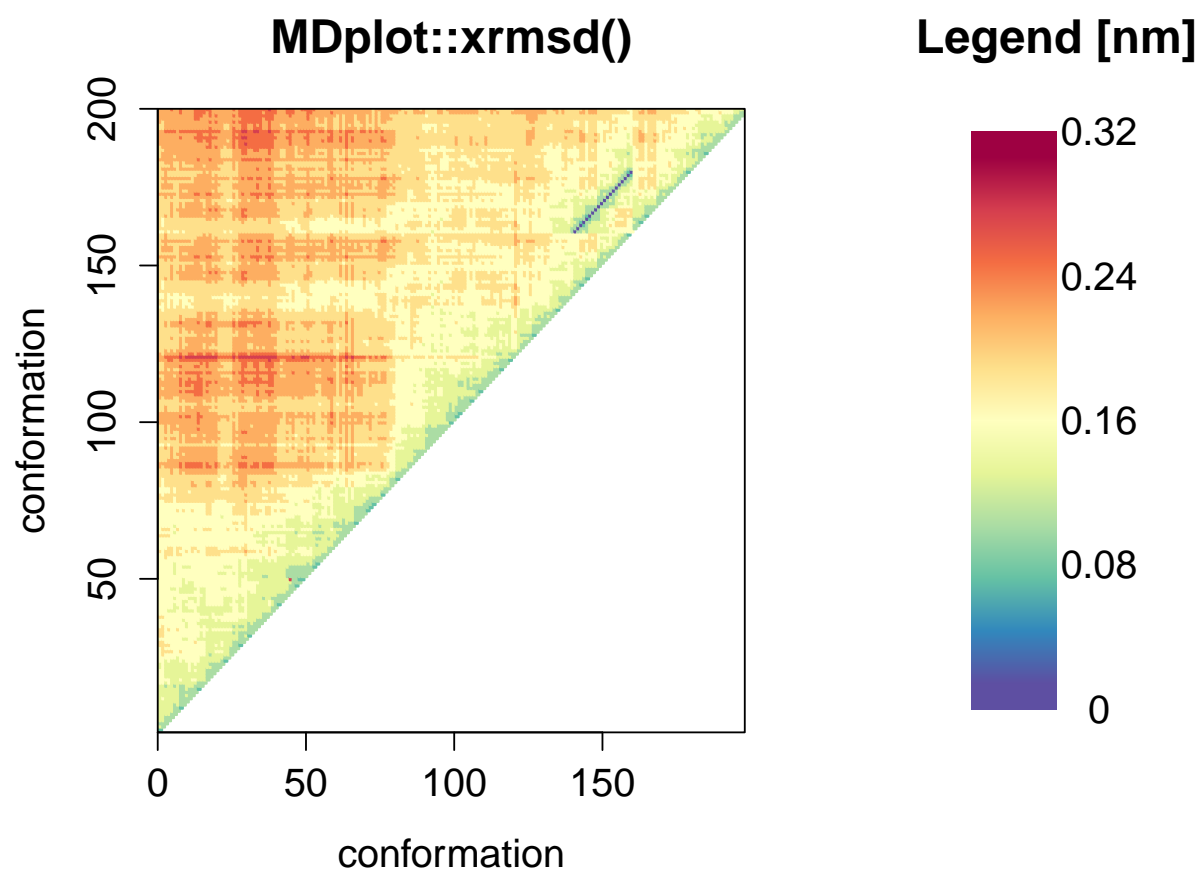


Figure 7.16: An example `xrmsd()` plot showing only the upper half because of the mirroring of the values.

7.4 Additional functions and the **bash** interface

Given that the plotting functions expect input to be stored in a defined data structure, the step of loading and parsing data from the text input files has been implemented in loading functions. Currently, they only support GROMOS file formats but further developments are planned to cover additional ones as well. In order to allow for direct calls from bash scripts, users might use the `Rscript` interface located in the folder `inst/bash` which serves as a wrapper shell. Pictures in the file formats PNG, TIFF or PDF can be used provided that the R installation used supports them. If `help=TRUE` is set, all the other options are ignored and a full list of options for every command is printed. In general, the names of the arguments of the functions are the same for calls by script. The syntax for these calls is `Rscript MDplot_bash.R {function name} [argument1=...] [argument2=...]`, which can be combined with bash variables (see below). The file path can be given in an absolute manner or relative to the `Rscript` folder path.

Listing 7.16: Example for bash script call

```
1  #!/bin/bash
2  # clusters
3  Rscript MDplot_bash.R clusters files=../extdata/clusters_example.txt.gz \
4                               title="Cluster analysis" size=900,900 \
5                               outformat=tiff outfile=clusters.tiff \
6                               clustersNumber=7 \
7                               names=WT,varA,varB,varC2,varD3,varE4
8
9  # xrmsd
10 Rscript MDplot_bash.R xrmsd files=../extdata/xrmsd_example.txt.gz title="XRMSD" \
11                             size=1100,900 outformat=pdf outfile=XRMSD.pdf \
12                             xaxisRange=75,145
13
14 # ramachandran
15 Rscript MDplot_bash.R ramachandran files=../extdata/ramachandran_example.txt.gz \
16                                 title="Ramachandran plot" size=1400,1400 resolution=175 \
17                                 outformat=tiff outfile=ramachandran.tiff angleColumns=1,2 \
18                                 bins=75,75 heatFun=norm printLegend=TRUE plotType=fancy
```

7.5 Conclusions

In this paper we have presented the package `MDplot` and described its application in the context of molecular dynamics simulation analysis. Automated figure generation is likely to aid in the understanding of results at the first glance and may be used in presentations and publications. Planned extension include both the integration of new functionalities such as a DISICL (secondary structure classification [11]) as well as the provision of loading interfaces for additional molecular dynamics engines. Further developments will be published on the projects' GitHub page.

7.6 Acknowledgements

The authors would like to thank Prof. Dr. Leisch for his useful comments and guidance in the package development process and Sophie Krecht for her assistance in typesetting this manuscript.

7.7 Appendix / Supplementary material

This section contains detailed information on the available loading functions. The latest major release of the package can be retrieved by downloading from [CRAN \(The Comprehensive R Archive Network\)](https://cran.r-project.org/web/packages/MDplot/index.html). Development and most up-to-date release candidates are published on the project's github page at <https://github.com/MDplot/MDplot>, where feedback is gathered as well.

7.8 Loading functions

In order to ease data preparation, loading functions have been devised which are currently able to load the output of standard GROMOS analysis tools [12] and store these data such, that they can be interpreted by the MDplot plotting functions. Next, we plan to support GROMACS output [3] as well (which is currently only supported by `load_rmsd()`). For other molecular dynamic engines the user has to specify how they should be read. In case other file formats are requested as well, we appreciate suggestions, requests and contributions on our github page. For detailed description of the data structures see the manual pages of the loading functions and generate them using the respective examples. For storage reasons, the input files have been compressed using `gzip` with R being able to use both compressed and uncompressed text files as input.

7.8.1 `load_clusters()`

This function loads clusters from a text file and stores them in a matrix. Trajectories are stored in rows, cluster occupancies in columns. The trajectories may be named by the user. The respective gromos++ analysis tool is `postcluster`.

```
1 # see "extdata/clusters_example.txt.gz" for format information
2 load_clusters(system.file("extdata/clusters_example.txt.gz",
3                           package="MDplot"))
```

argument name	default value	description
<code>path</code>	<i>none</i>	Specifies the path of the text input file.
<code>names</code>	NA	Optional vector of trajectory names. Needs to be of the same length as the number of clusters to be plotted.
<code>mdEngine</code>	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.2 `load_clusters_ts()`

This function loads the timeseries information for clusters from a text file and stores them in a list of lists. Every trajectory is stored in a list, with the name as first attribute and a vector of cluster identifiers as the second. The trajectories may be named by the user. The respective gromos++ analysis tool is `cluster`.

```

1 # see "extdata/clusters_ts_example.txt.gz" for format information
2 load_clusters_ts(system.file("extdata/clusters_ts_example.txt.gz",
3                             package="MDplot"),
4                  lengths = c(4000, 4000, 4000, 4000, 4000, 4000))

```

argument name	default value	description
path	<i>none</i>	Specifies the path of the text input file.
lengths	<i>none</i>	Mandatory vector holding the number of snapshots for the respective trajectories.
names	NA	Optional vector of trajectory names. Needs to be of the same length as the number of clusters to be plotted.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.3 load_dssp_summary()

Loads DSSP summary output from a text file into a table. Residues are given in rows. The respective gromos++ analysis tool is dssp.

```

1 # see "extdata/dssp_summary_example.txt.gz" for format
2 load_dssp_summary(system.file("extdata/dssp_summary_example.txt.gz",
3                               package="MDplot"))

```

argument name	default value	description
path	<i>none</i>	Path to the input file.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.4 load_dssp_ts()

Loads DSSP output files from a specified directory and combines it into a list suited for `dssp_ts()`. The respective `gromos++` analysis tool is `dssp`.

```
1 # see "extdata" for format information
2 load_dssp_ts(system.file("extdata", package="MDplot"))
```

argument name	default value	description
folder	<i>none</i>	Folder, in which the DSSP output files are located.
filenames	NA	Vector with filenames. Default file names are "3-Helix.out", "4-Helix.out", "5-Helix.out", "Bend.out", "Beta-Bridge.out", "Beta-Strand.out" and "Turn.out". Files not present are ignored.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.5 load_hbond()

This function loads hydrogen bond data from a text file and stores them in a table. The respective `gromos++` analysis tool is `hbond`.

```
1 # see "extdata/hbond_example.txt.gz" for format information
2 load_hbond(system.file("extdata/hbond_example.txt.gz",
3                        package="MDplot"))
```

argument name	default value	description
path	<i>none</i>	Specifies the path of the text input file.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.6 load_hbond_ts()

This function loads hydrogen bond timeseries data from a text file and stores them in a table. The respective `gromos++` analysis tool is `hbond`.

```
1 # see "extdata/hbond_ts_example.txt.gz" for format information
2 load_hbond_ts(system.file("extdata/hbond_ts_example.txt.gz",
3                          package="MDplot"))
```

argument name	default value	description
path	<i>none</i>	Specifies the path of the text input file.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.7 load_noe()

Loads NOE violations and returns a list. The respective gromos++ analysis tools are `prep_noe`, `noe` and `post_noe` (output produced by the last program).

```

1 # loading
2 load_noe(system.file("extdata/noe_example.txt.gz",
3                     package = "MDplot"))

```

argument name	default value	description
files	<i>none</i>	Vector of files holding NOE violations as produced by gromos++ program <code>post_noe</code> .

7.8.8 load_ramachandran()

Loads textfile with two columns of dihedral angles, which are to be stored in a matrix. By default, the first column is ϕ and the second ψ . Angles can be shifted by a constant value (in order to transform them from 0 to 360 degrees to the usually used -180 to 180 degrees). The respective gromos++ analysis tool is `tser`.

```

1 # load table
2 # see "extdata/ramachandran_example.txt.gz" for format information
3 load_ramachandran(system.file("extdata/ramachandran_example.txt.gz",
4                             package="MDplot"))

```

argument name	default value	description
path	<i>none</i>	Path to input file. Two columns of the same length are expected.
angleColumns	<code>c(1, 2)</code>	If more columns are present, the angle columns can be chosen by this vector.
shiftAngles	NA	In order to shift the values by a constant factor (e.g. -180).
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.9 load_rmsd()

Returns a list of vector pairs of datapoint indices and RMSD values. The respective gromos++ analysis tool is rmsd.

```
1 load_rmsd(c(system.file("extdata/rmsd1_example.txt.gz",
2                       package="MDplot"),
3            system.file("extdata/rmsd2_example.txt.gz",
4                       package="MDplot")))
```

argument name	default value	description
files	<i>none</i>	Vector of paths to input text files.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.10 load_rmsf()

Returns a list of vector pairs of datapoint indices and RMSF values. The respective gromos++ analysis tool is rmsf.

```
1 load_rmsf(c(system.file("extdata/rmsf1_example.txt.gz",
2                       package="MDplot"),
3            system.file("extdata/rmsf2_example.txt.gz",
4                       package="MDplot")))
```

argument name	default value	description
files	<i>none</i>	Vector of paths to input text files.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.11 load_TIcurve()

Returns a list of matrices holding three columns (lambda state point, value and error) for every input file. A maximum of two files may be provided. The respective gromos++ analysis tool is ene_ana.

```
1 # loading
2 load_TIcurve(c(system.file("extdata/TIcurve_example.txt.gz",
3                          package="MDplot")))
```

argument name	default value	description
files	<i>none</i>	Vector of files (up to two) to be loaded.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.12 load_timeseries()

Returns a list of vector pairs of datapoint indices and response values. The respective analysis tool depends on the data generated.

```
1 # loading
2 load_timeseries(system.file("extdata/timeseries_example.txt.gz",
3                          package="MDplot"))
```

argument name	default value	description
files	<i>none</i>	Vector of files to be loaded.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

7.8.13 load_xrmsd()

Loads matrix information from the specified file. The respective gromos++ analysis tool is rmsdmat.

```
1 # loading
2 load_xrmsd(system.file("extdata/xrmsd_example.txt.gz",
3                       package = "MDplot"))
```

argument name	default value	description
path	<i>none</i>	Specifies the input file.
factor	10000	In case the RMSD values are given in nm * factor by the analysis program (for efficiency reasons), the factor can be specified. If the unit is already nanometers, 1 is the appropriate value.
mdEngine	"GROMOS"	Argument introduced for distinction between input formats based on the used molecular dynamics engine (to be implemented).

References

- [1] Nathan Schmid, Clara D. Christ, Markus Christen, Andreas P. Eichenberger, and Wilfred F. van Gunsteren. “Architecture, implementation and parallelisation of the GROMOS software for biomolecular simulation”. In: *Computer Physics Communications* 183.4 (Apr. 2012), pp. 890–903. DOI: [10.1016/j.cpc.2011.12.014](https://doi.org/10.1016/j.cpc.2011.12.014).
- [2] Andreas P. Eichenberger et al. “GROMOS++ Software for the Analysis of Biomolecular Simulation Trajectories”. In: *Journal of Chemical Theory and Computation* 7.10 (Oct. 2011), pp. 3379–3390. DOI: [10.1021/ct2003622](https://doi.org/10.1021/ct2003622).
- [3] Sander Pronk et al. “GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit”. In: *Bioinformatics* 29.7 (Apr. 2013), pp. 845–854. DOI: [10.1093/bioinformatics/btt055](https://doi.org/10.1093/bioinformatics/btt055).
- [4] James C. Phillips et al. “Scalable molecular dynamics with NAMD”. In: *J. Comput. Chem.* 26.16 (Dec. 2005), pp. 1781–1802. DOI: [10.1002/jcc.20289](https://doi.org/10.1002/jcc.20289).
- [5] Wendy D. Cornell et al. “A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules”. In: *Journal of the American Chemical Society* 117.19 (May 1995), pp. 5179–5197. DOI: [10.1021/ja00124a002](https://doi.org/10.1021/ja00124a002).
- [6] B.R. Brooks et al. “CHARMM: The Biomolecular Simulation Program”. In: *J. Comput. Chem.* 30.10 (July 2009), pp. 1545–1614. DOI: [10.1002/jcc.21287](https://doi.org/10.1002/jcc.21287).
- [7] Barry J. Grant, Ana P. C. Rodrigues, Karim M. ElSawy, J. Andrew McCammon, and Leo S. D. Caves. “Bio3d: an R package for the comparative analysis of protein structures”. In: *Bioinformatics* 22.21 (2006), pp. 2695–2696. DOI: [10.1093/bioinformatics/btl461](https://doi.org/10.1093/bioinformatics/btl461).
- [8] W. Kabsch and C. Sander. “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features”. In: *Biopolymers* 22.12 (Dec. 1983), pp. 2577–2637. DOI: [10.1002/bip.360221211](https://doi.org/10.1002/bip.360221211).
- [9] G. N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. “Stereochemistry of polypeptide chain configurations”. In: *Journal of Molecular Biology* 7 (July 1963), pp. 95–99.
- [10] J.G. Kirkwood. “Statistical mechanics of fluid mixtures”. In: *J. Chem. Phys.* (1935), pp. 300–313.
- [11] Gabor Nagy and Chris Oostenbrink. “Dihedral-based segment identification and classification of biopolymers II: polynucleotides”. eng. In: *Journal of Chemical Information and Modeling* 54.1 (Jan. 2014), pp. 278–288. DOI: [10.1021/ci400542n](https://doi.org/10.1021/ci400542n).
- [12] Andreas P. Eichenberger et al. “GROMOS++ Software for the Analysis of Biomolecular Simulation Trajectories”. In: *Journal of Chemical Theory and Computation* 7.10 (Oct. 2011), pp. 3379–3390. DOI: [10.1021/ct2003622](https://doi.org/10.1021/ct2003622).

Conclusion

In a nutshell, this work is about the effects of small chemical or parameter changes, more specifically on amino acid side-chain and backbone alterations, in the context of peptides and proteins. The latter are part of the force-fields' basis and, therefore, their proper description is a prerequisite to any accurate protein simulation. The former, regardless of their origin - be it a canonical mutation in a protein-encoding gene or one of the numerous post-translational modifications (PTMs) - are able to exert significant influence on the overall structure and dynamics of the much larger macro-molecules they are components of.

In the context of molecular modeling and simulation, this imposes an enormous challenge on us: because small changes sometimes yield astonishing effects, our models must be as precise as to capture even little shifts in the interactions both between molecules and inside them. This thesis' first part documents our efforts to improve our models for PTMs, by updating them in a consistent way to match our most recent parameter set 54A8. For most of the charged side-chains considered, we could rely on the thoroughly tested parameters of this set. However, the chemical group of phosphates, important both as a common modification to serine, tyrosine and threonine as well as in lipids and the backbone of nucleic acids, was missing. Hence, we parametrized dihydrogen phosphate, methyl-phosphate, di-methyl-phosphate and phenyl-phosphate prior to applying the update of the force field parameters for 23 charged post-translationally modified amino acids, which were deposited on our Vienna-PTM webserver. However, as outlined before, phosphate moieties also occur in other biologically relevant contexts than PTMs. A next step in validating our proposed parameters should involve tests in the nucleic acids' backbone or membrane simulations. The large-scale investigation of PTMs described in chapter 4 revealed the complexity, with which they influence dynamical features of the precursor amino acids. Hence, it would be worthwhile to test the predictions made by actual simulations, or to extend the length of the used model system, e.g. by testing the effect of hyper-phosphorylations on the β/P_{II} -ratio in proline-rich peptide segments.

Post-translational modifications are important to modulate a protein, but their surrounding, the 20 canonical amino acids are critical as well. Due to their interactions, they build the free energy surface and virtually the "scaffold" in which small changes may alter the balance between different energetic minima. Since our scope of interest is usually on the protein scale, the accurate description of the amino acid level easily appears circumstantial. However, as has been elaborated in the introduction, in principle we should be able to describe everything from the single building block to the huge protein-complex with similar and sufficient accuracy - as long as our models cover all relevant interactions. In this work we could show that this is not the case for the current GROMOS backbone dihedral angle parameters used for amino acids. These torsions have an immediate effect on the

preferred conformations along the backbone, being crucial for the (secondary) structure of peptides and proteins. Somewhat surprisingly, it is still possible, though, to run viable protein simulations using the current parameters. This is, because the non-bonded interactions become a major contribution inside a large, well-structured and stable backbone. Nevertheless, in scenarios where that does not hold, such as in a very flexible part of a protein, an intrinsically unstructured region or in short peptide sequences, our models might become inappropriate. In this study, we tried to improve the description of individual amino acids by screening a huge collection of parameter candidates for their reproduction of an experimental NMR coupling constant and secondary structure preferences derived from Raman spectroscopy. We identified four major groups of amino acids, separated by their C_β -branching pattern: glycine, with its almost unrestricted sampling of the Ramachandran space, alanine having no attachment to its C_β , the "common" amino acids that have a single protrusion from their respective C_β s and finally the three amino acids that have two substituents at C_β limiting their rotations around the ϕ and ψ backbone dihedral angles, namely valine, threonine and isoleucine. Based on this data, we conclude, that a common parametrization of all amino acids would have the potential to hamper the force-fields' accuracy. For each of these subgroups, we propose new parameters which significantly surpass the performance of the original ones. However, in order to apply these parameters to peptide and protein simulations, further rounds of refinement will be required on these levels, again making use of experimental data (e.g. NMR). Moreover, there are multiple solutions in our test set of 100 000 combinations, which perform similarly well. These high-potential subsets could be further evaluated on the capped amino acid level, making use rather of primary data (e.g. Raman spectroscopy, which we could also obtain from simulations) than secondary data (such as the secondary structure propensities). In any case, there is much space for further improvement.

One example where the modification or mutation of even a single amino acid has crucial effects, is the super-humanization of antibodies and the subsequent reengineering. In this process, one tries to "humanize" an antibody derived from model organisms in order to mitigate immunogenic reactions against it by the human body. However, that leads to partial or complete loss of binding, because the scaffold regions matter to the configurations of the loops, the actual binding sites. One solution is the deliberate backmutation of as few amino acids as possible in order to obtain a binding (chimeric) antibody again. Together with our wet-lab collaborators, we established a computational method to predict the effect of one or a few single-point mutations to the structure and dynamics (and hence ultimately) the binding of antibodies from *in-silico* simulations. A second achievement was the identification of a tyrosine-cluster in close spatial proximity to the third complementarity-determining region (CDR) of the heavy chain as an important feature shared by all binding variants. The critical backmutation that reestablished binding to a significant extent, was arginine 98 (R98) which is surrounded by tyrosines comprising the above-mentioned cluster. Through cation- π interactions, this construct has a strong influence on the conformations the nearby CDR will adopt. The conformational selection paradigm might give an answer, why this leads to an alteration in binding affinity: the configurations, that are necessary for binding are stabilized by the motional restriction imposed by the cluster. Currently, additional predictions for other mutants are experimentally validated and show promising preliminary results, proving our method capable of providing accurate qualitative results. Further developments might focus on the refinement of the score calculation (e.g. by taking electrostatic surfaces into account), in order to capture more subtle differences between mutants and allowing a quantitative prediction of the effect of a mutation on the binding affinity.

Another, more technical, problem that concerns the search for critical backmutations in antibodies and the screening of parameters alike, which is the sheer size of the projects. Many mutants and parameter candidates had to be prepared, simulated and analyzed to obtain the necessary information. Also, it is almost impossible to tell in advance how many simulations or analysis steps are required. For example, one might discover a new promising site in the course of the antibody project. However, to prepare, execute and supervise the simulations (multiple per candidate, to cover more of the phase space) as well as to perform the complex analysis afterwards, including clustering and various RMSD calculations, is a cumbersome undertaking, requiring a lot of human input. The most striking feature of these large projects, is the strong similarity of the respective simulations to the previously performed ones. Apart from minor details, such as the precise number of atoms in the systems, these simulations are the same, i.e. they share the same workflow logic. We made use of this fact, by implementing PROMETHEUS, a molecular dynamics workflow manager capable of instantiating process templates such, that these minor differences are fully considered while automatically proceeding through the workflow steps defined. Moreover, all subjobs (e.g. each representing a distinct mutated antibody) can be processed as a whole and are easily monitored by provision of a web-based graphical user interface to display their status. In the course of this study, PROMETHEUS enabled the automated handling of thousands of processes, including hundreds of simulations.

The data produced grows according to the number of simulations and different types of analyses. When searching for appropriate dihedral angle parameters, we looked at hundreds of low-resolution Ramachandran plots, screened potential energy curves of backbone-torsions and calculated average-RMSD graphs for the backbone atom of proteins. Moreover, figures are in general the preferred way to transfer and present data. Therefore, for immediate use in the above mentioned projects and also as an assistance to others, we developed the R package `MDplot` which allows the automated plot generation of many of the most required figure types used in the context of molecular dynamics simulations. Currently, 14 plot and numerous loading functions (the latter supporting only GROMOS formats, momentarily) and their simple integration into `bash` scripts or PROMETHEUS instruction files allow to produce high-quality graphs easily. A natural extension to both PROMETHEUS and `MDplot` is to support other MD engines than GROMOS. This would not only enlarge the basis of potential users, but also allow e.g. the combination of the advantages of GROMOS and GROMACS within one workflow.

As mentioned in the introduction, force-fields will always be under constant development, either by optimizing existing parameters as we described in this work, or by the introduction of new degrees of freedom such as the polarisability of atoms. In this sense, we hope to contribute to the description of (post-translationally) modified amino acids, both by the update of the charged PTMs and phosphate containing groups as well as by the identification of promising backbone dihedral angle parameters for amino acids. Part of the ever better accuracy of our simulation results also stems from the long time-scales and numbers of simulations affordable nowadays due to cheap and better computational hardware. As elaborated above, this also tightens the need for automated, user-friendly workflows for simulations and analyses, which we addressed by the development of PROMETHEUS and `MDplot`. With these tools, suited for large-scale projects such as the antibody project and the force-field improvements described, this thesis aims to enhance the quality of protein studies in general and investigations aiming on changes introduced by small chemical alterations in particular.

Addendum

9.1 Dihedral parameters in proteins

9.1.1 Introduction

In chapter 3, a Hamiltonian reweighting approach has been applied in order to determine backbone dihedral angle parameters for blocked amino acids. The ones used by the GROMOS community till now, have been shown to be performing sufficiently at the complexity level of proteins, but do fail for these very small peptide systems. The reason for that is most likely the critical contribution of neighboring atoms by both bonded and non-bonded interactions to the potential energy (shown in figure 3.6). Although according to the small compounds, the rotations around the backbone dihedral angles ϕ and ψ are not properly described by the 54A7 parameter set, the strong interactions occurring in the well-structured context of proteins lead to a meaningful behavior.

One could argue, that different backbone parameters are required for the small compounds, than for peptide chains. However, this situation would be highly unsatisfying for it implies that the protein structures are only stable thanks to a cancellation of errors or at least a compensation of local interactions by longer ranged ones. Therefore, new backbone parameters such as the suggested ones in this thesis need to be ultimately validated in large systems, which will in the ideal case lead to a parameter set, that fulfills all requirements sufficiently simultaneously.

In this section we report preliminary protein simulations of lysozyme and the GCN4 trigger peptide (PDB-codes 4B0D [1] and 2OVN [2] model 1, respectively) using our suggested parameters as a first step towards the validation at the protein level. We compare them to simulations with the 54A8 parameter set [3] (the backbone dihedrals have not changed since 54A7 [4]). One assumption is the principle transferability of our suggested parameters, which were established by use of the 54A7 parameter set.¹

¹For 54A8, the partial charges of the charged side-chains (D, E, H, K, R) have been changed. This could in theory lead to a shift in the best match for the four amino acid groups parametrized.

9.1.2 Methods

9.1.2.1 Simulated systems

For lysozyme, four replicate simulations using a regular 54A8 topology have been performed² applying the simulation setup described below. The very same settings were also used to simulate four copies using a 54A8 topology with updated backbone dihedral angle parameters (chapter 3).

The GCN4 trigger peptide (sequence: NYHLENEVARLKKLVE) has similarly been subjected to 2x3 simulations using a standard 54A8 and an updated topology. The individual replicate simulations were started with different initial seeds to assign different velocities to their atoms in the beginning. This protocol ensures statistical relevance of any observations and allows simple parallelization compared to a single very long simulation.

9.1.2.2 Simulation setup

All simulations were performed using the GROMOS simulation package [5, 6] with the 54A8 parameter set [3]. The rectangular periodic simulation box contained for lysozyme 10493 and for the GCN4 trigger peptide 7045 water molecules with a minimum solute to wall distance of 1.0 nm and 1.75 nm, respectively. They were carried out at 300 K and without additional counter-ions. The systems were equilibrated to this temperature in six discrete steps with a simulation length of 20 ps each and a weak thermostat-coupling with two baths for the solute and the solvent (relaxation time τ of 0.1 ps) and weak barostat-coupling [7] (relaxation time τ_P of 0.5 ps) and an isothermal compressibility of $4.575 \cdot 10^{-4} (\text{kJ} \cdot \text{mol}^{-1} \cdot \text{nm}^{-3})^{-1}$. The simulations of lysozyme amounted to 50 ns and those of the GCN4 trigger peptides to 100 ns each (with a storage rate of the configurations of 250 and 500 steps, respectively). For the lysozyme simulations, roto-translational constraints have been applied. SHAKE [8] was used to maintain the bond lengths at their energy minima. The integration time step was 2 fs. Interactions within 0.8 nm were calculated at every time step from a pairlist that was updated every five steps. Intermediate range interactions up to a distance of 1.4 nm were calculated at pairlist updates and kept constant between updates. Long range interactions were approximated with a reaction field contribution [9] to the energies and forces, accounting for a homogeneous medium with a relative dielectric constant [10] of 61 beyond the cut-off of 1.4 nm.

9.1.2.3 Analysis

Backbone atoms root-mean-square-deviation (RMSD) The backbone atom RMSD with respect to the crystal structure of a protein is often used to describe the overall stability of a protein. However, since X-ray crystallography provides a time and ensemble average, it is not a valid quality criterion by itself but rather a general indicator as a high or ever increasing RMSD (depending on the size of the system) hints at an instability that might be artificial for stable, globular proteins. The RMSD is calculated according to equation 9.1.

$$RMSD(x) = \sqrt{\frac{\sum_i^N (x_i - x_i^{\text{ref}})^2}{N}} \quad (9.1)$$

²By Matthias Diem, whom we would like to thank for that effort.

over all N atoms i for a structure described by coordinates x . Note, that a very low RMSD does not necessarily mean a good model. Overstabilisation of the structure of biomolecules is as problematic as artificial instability, since it avoids a proper representation on the dynamics and hampers sampling.

Secondary structure occurrence Another measure of protein and peptide structure is the persistence of secondary structure elements. The define secondary structure of proteins algorithm (DSSP) [11] allows a classification of the backbone conformations into various structural classes. The persistence of certain elements can be averaged over a simulation trajectory to obtain an estimate of stability or conformational preference.

Nuclear Overhauser Effects (NOEs) An NOE relates two atoms i and j through space, with the intensity being proportional to the distance to the power -6 (see equation 9.2). It is therefore possible, to validate a given force field parameter set by calculating the $\langle r^{-6} \rangle^{-\frac{1}{6}}$ averaged distances between the atoms and comparing them to the actual experimental distance boundaries. A longer distance in the simulations represents a violation of the experimental NOE signal. In this study, we used the NOE violations as a measure for the matching of the structural ensemble as obtained from a protein simulation to NMR experiments.

$$Q(r_{ij}) \propto r_{ij}^{-6} \quad (9.2)$$

In general, the lower the number of NOE violations and the shorter their distance the better the agreement. However, since the crystal structure also violates the NMR boundaries significantly and not all boundaries are of equal quality, violations are not *a priori* a sign of general invalidity of a force field. The NOE data set has been extracted and hand-curated from PDB entry 1E8L [12].

9.1.3 Results and discussion

All figures in this section were generated with the MDplot package, described in chapter 7. For lysozyme, the RMSD of the backbone atoms is slightly higher for our set (figures 9.1 and 9.2), with a small spread from the mean values. Overall, the structure seems to remain stable and also the helical content remains at $37.6 \pm 2.8\%$ as compared to $36.6 \pm 1.1\%$ for 54A8 (averaged over the sequence simulation length and replicates). The NOE violations observed for our suggested set of parameters are slightly worse both by the highest individual value but also in the total amount for lysozyme (figures 9.3 and 9.4) compared to the 54A8 simulations. However, the differences seem minor and our suggested set has not yet been optimized in this respect and performs reasonable.

The RMSD for the GCN peptide behaves similarly for all simulations, regardless of the exact force field used (figures 9.5 and 9.6). All simulations show some degree of unfolding, but this seems to be faster for the simulations with the updated dihedral angle parameters than for the original 54A8 parameter set. The secondary structure analyses in figures 9.7 and 9.8 seem to support this finding.

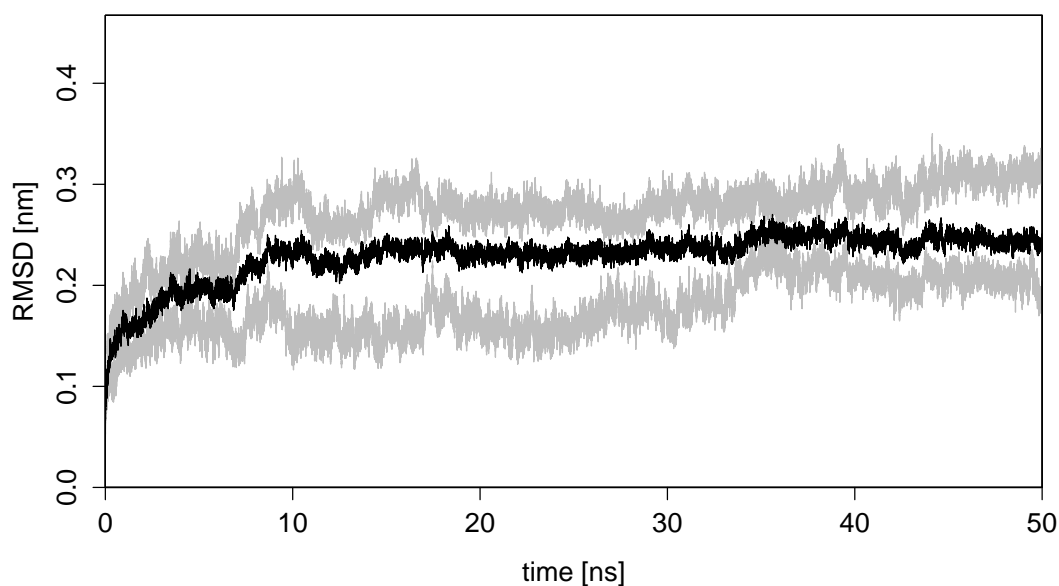


Figure 9.1: Average backbone atom root-mean-square-deviation plot for the four replicates of the lysozyme protein simulations using the 54A8 parameter set. The grey curves show the maximum and minimum values for the four runs at every snapshot (i.e. the spread) and the black curve the mean values.

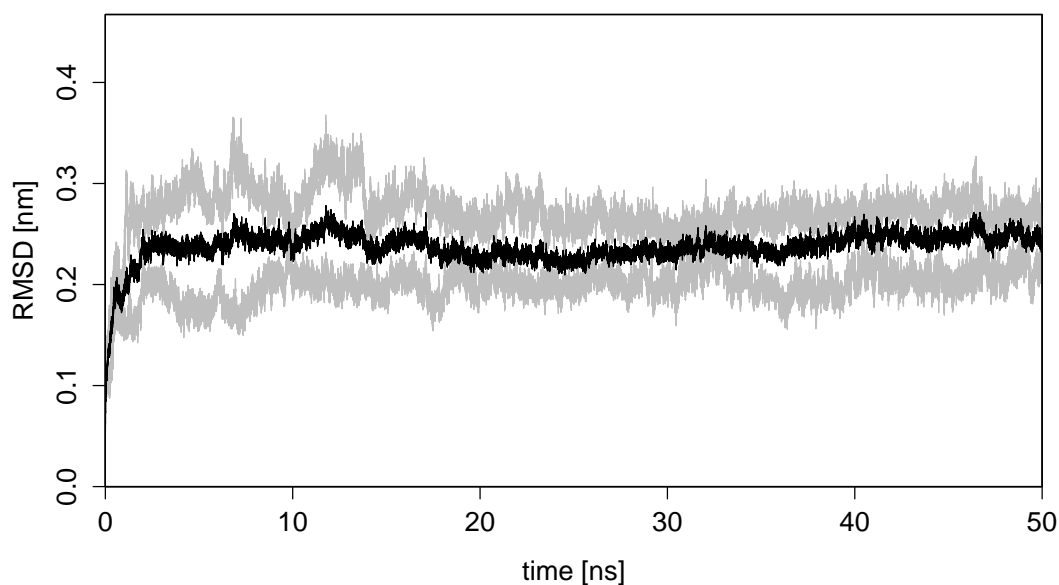


Figure 9.2: Average backbone atom root-mean-square-deviation plot for the four replicates of the lysozyme protein simulations using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. Note, that unlike the curves in figure 9.1, there is no increase in the RMSD at the end of the trajectories.

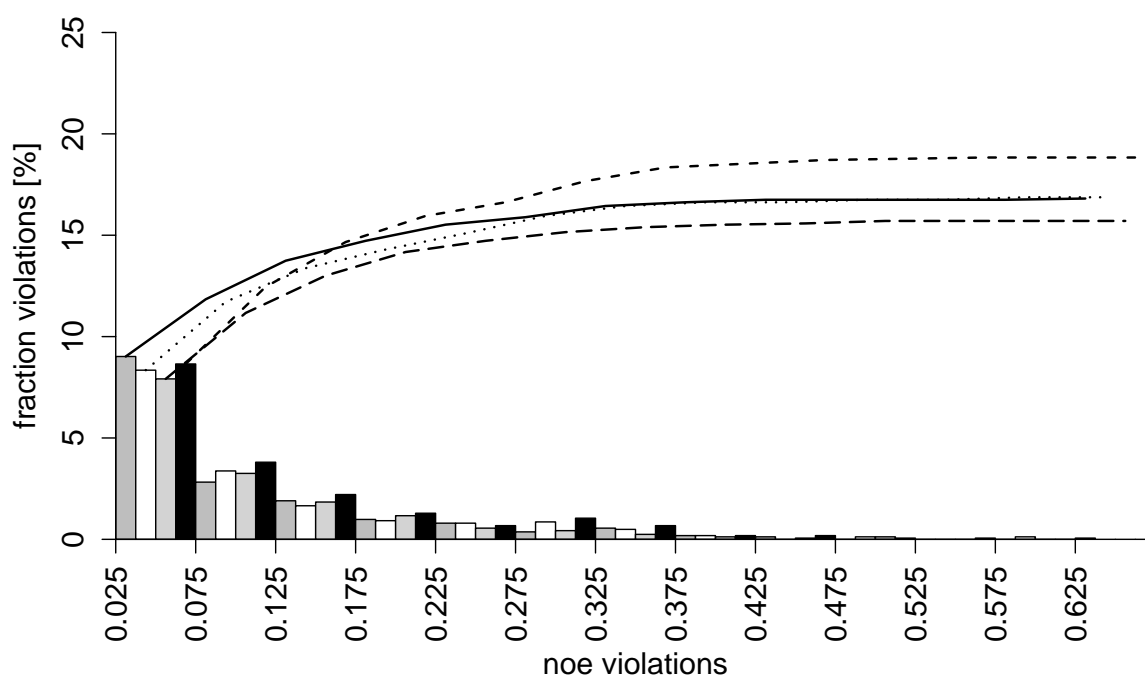


Figure 9.3: Distribution of Nuclear Overhauser Effect (NOE) violations for lysozyme using the 54A8 parameters. The curves on top of the bars, are the sum of the violations.

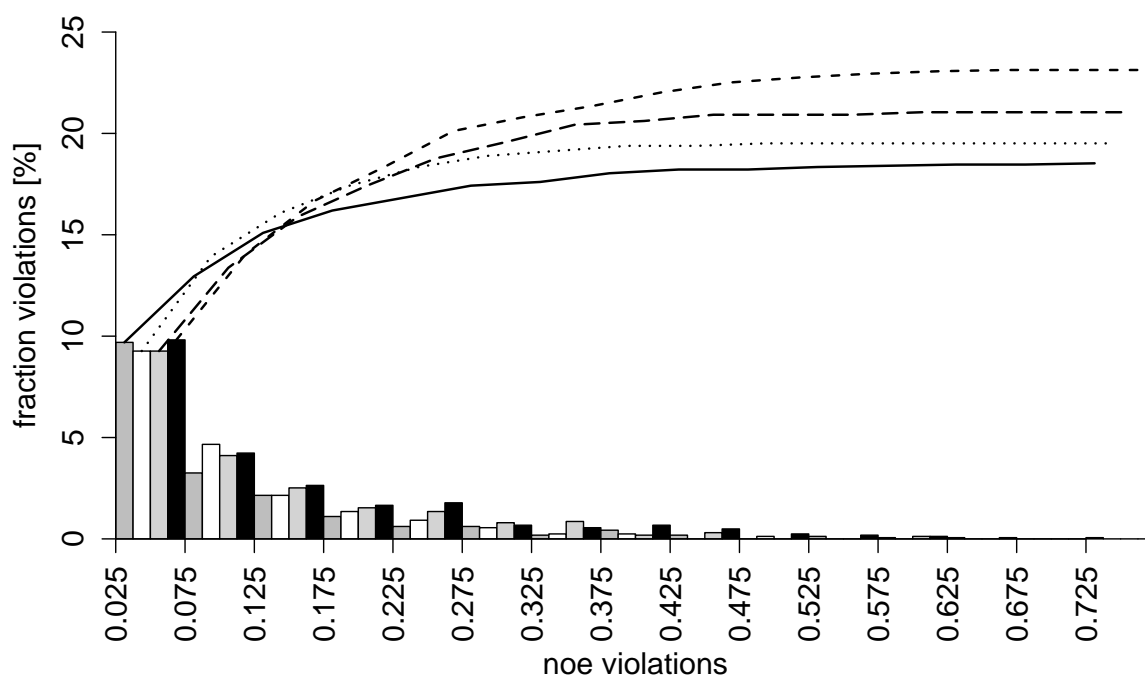


Figure 9.4: Nuclear Overhauser Effect (NOE) violations for lysozyme using our suggested parameters.

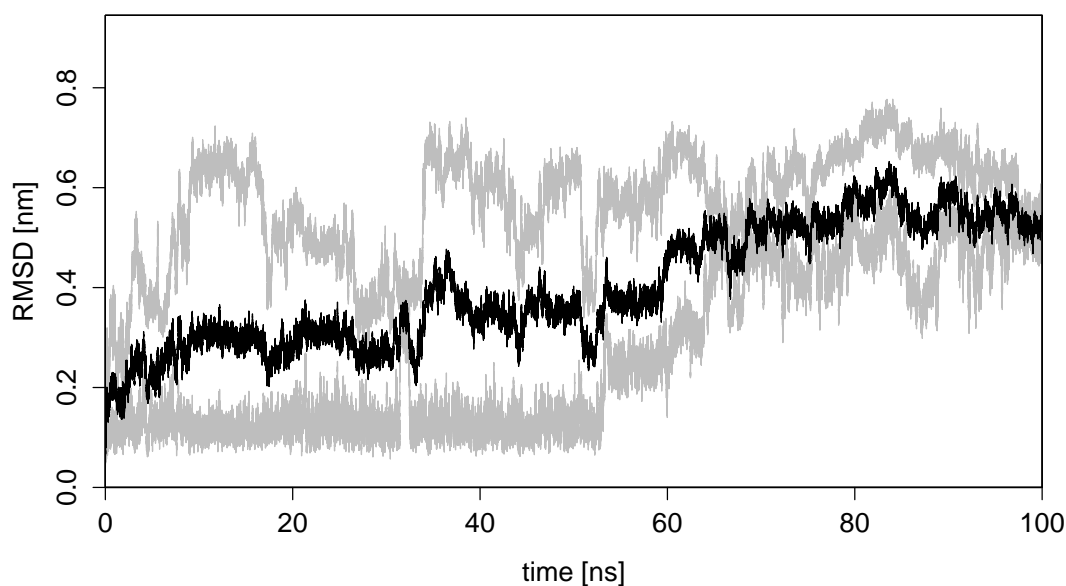


Figure 9.5: Average backbone atom root-mean-square-deviation plot for three replicates of the GCN4 trigger protein using the 54A8 parameter set.

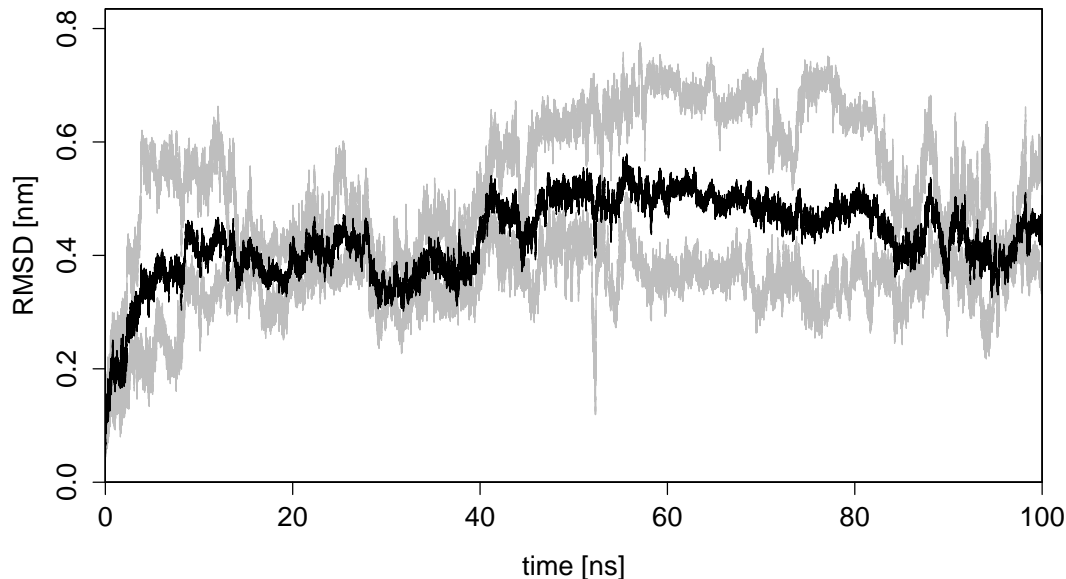


Figure 9.6: Average backbone atom root-mean-square-deviation plot for three replicates of the GCN4 trigger protein using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. The spread is comparable to the one in figure 9.5, both sets of simulations show a strong increase in the RMSD values (most likely from the flexible residues at the termini).

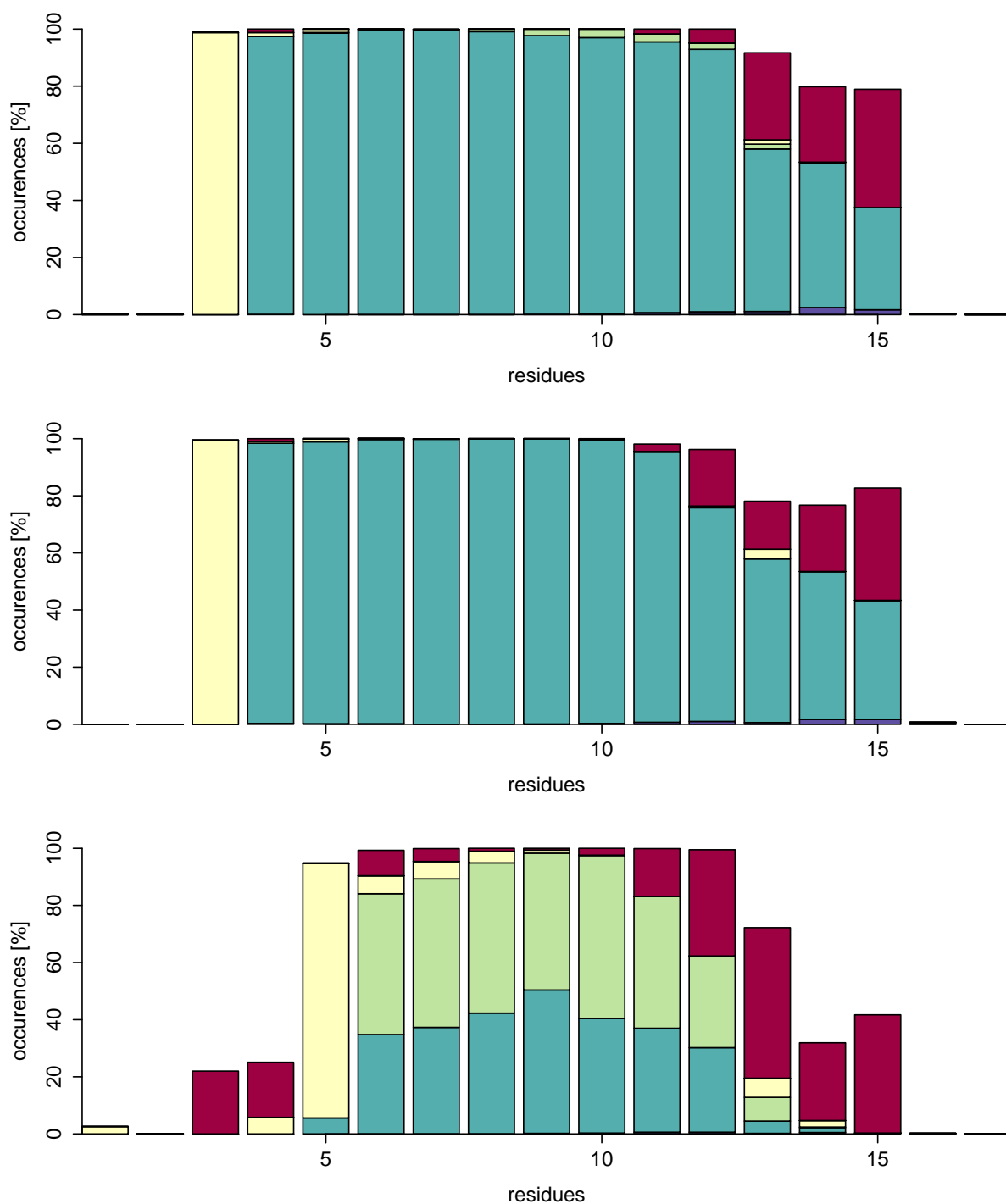


Figure 9.7: DSSP summary plot over the whole trajectory for the GCN4 trigger peptide simulations using the 54A8 parameter set. While the first two simulations are quite similar, the secondary structure propensities of the third are rather different, especially in the helix-type of the right-handed helical core region.

Legend: ● ... 3-Helix, ● ... 4-Helix, ● ... 5-Helix, ● ... Turn, ● ... β -Strand, ● ... β -Bridge, ● ... Bend.

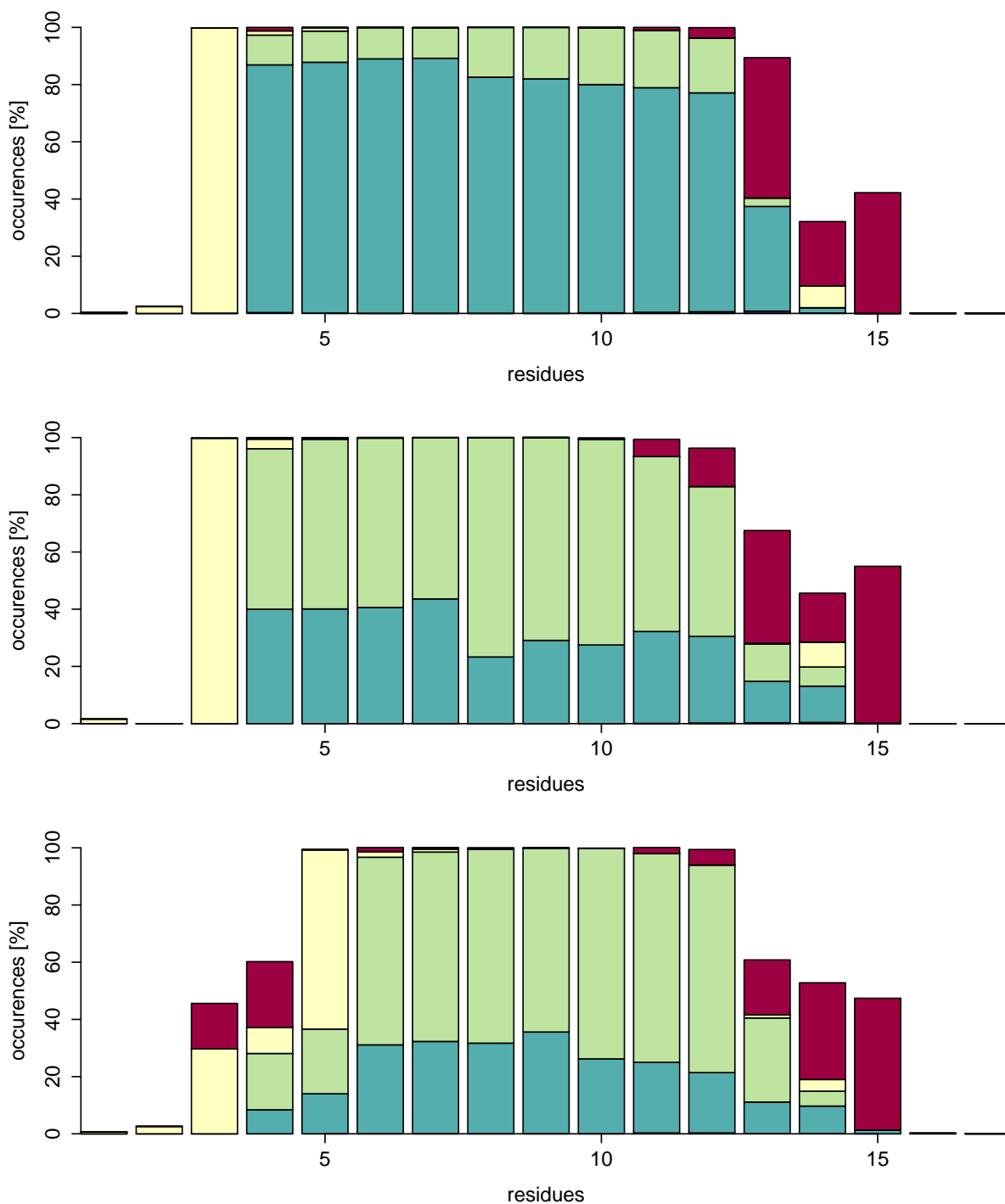


Figure 9.8: DSSP summary plot over the whole trajectory for the GCN4 trigger peptide simulations using the 54A8 parameter set updated with the suggested dihedral angle parameters taken from chapter 3. Compared to figure 9.7, there is a shift in the helical populations.

Legend: ● ... 3-Helix, ● ... 4-Helix, ● ... 5-Helix, ● ... Turn, ● ... β -Strand, ● ... β -Bridge, ● ... Bend.

While a large part of the peptide remains α -helical (4-helix) in two of the three simulations with 54A8, a larger fraction of π -helix (5-helix) is observed for the simulations with the updated parameter set, indicating a degree of unwinding. The π -helical fraction was not so pronounced in the larger lysozyme simulations.

To summarize, our parameters reproduce the experimental data at the capped amino acid level well, including a proper representation of glycine and the elimination of numerous outliers as described in previous chapters. They also show quite a promising performance in the two protein systems described. The ultimate goal, however, is a set of parameters that works at least as well in the context of structured proteins, as in describing unstructured regions and very small peptidic fragments. Therefore, further rounds of improvement and broader tests on the different levels of complexity are required prior to any application.

References

- [1] Florent Cipriani, Martin Röwer, Christophe Landret, Ulrich Zander, Franck Felisaz, and Jose Antonio Marquez. “CrystalDirect: a new method for automated crystal harvesting based on laser-induced photoablation of thin films”. In: *Acta Crystallogr. Sect. D* 68.10 (Oct. 2012), pp. 1393–1399. DOI: [10.1107/S0907444912031459](https://doi.org/10.1107/S0907444912031459).
- [2] M. O. Steinmetz, I. Jelesarov, W. M. Matousek, S. Honnappa, W. Jahnke, J. H. Missimer, S. Frank, A. T. Alexandrescu, and R. A. Kammerer. “Molecular basis of coiled-coil formation”. In: *Proceedings of the National Academy of Sciences* 104.17 (Apr. 2007), pp. 7062–7067. DOI: [10.1073/pnas.0700321104](https://doi.org/10.1073/pnas.0700321104).
- [3] Maria M. Reif, Philippe H. Hünenberger, and Chris Oostenbrink. “New Interaction Parameters for Charged Amino Acid Side Chains in the GROMOS Force Field”. In: *Journal of Chemical Theory and Computation* 8.10 (Oct. 2012), pp. 3705–3723. DOI: [10.1021/ct300156h](https://doi.org/10.1021/ct300156h).
- [4] Nathan Schmid, Andreas P. Eichenberger, Alexandra Choutko, Sereina Riniker, Moritz Winger, Alan E. Mark, and Wilfred F. van Gunsteren. “Definition and testing of the GROMOS force-field versions 54A7 and 54B7”. In: *European Biophysics Journal* 40.7 (July 1, 2011), pp. 843–856. DOI: [10.1007/s00249-011-0700-9](https://doi.org/10.1007/s00249-011-0700-9).
- [5] Markus Christen et al. “The GROMOS software for biomolecular simulation: GROMOS05”. In: *Journal of Computational Chemistry* 26.16 (2005), pp. 1719–1751. DOI: [10.1002/jcc.20303](https://doi.org/10.1002/jcc.20303).
- [6] Nathan Schmid, Clara D. Christ, Markus Christen, Andreas P. Eichenberger, and Wilfred F. van Gunsteren. “Architecture, implementation and parallelisation of the GROMOS software for biomolecular simulation”. In: *Computer Physics Communications* 183.4 (Apr. 2012), pp. 890–903. DOI: [10.1016/j.cpc.2011.12.014](https://doi.org/10.1016/j.cpc.2011.12.014).
- [7] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. “Molecular dynamics with coupling to an external bath”. In: *The Journal of Chemical Physics* 81.8 (Oct. 15, 1984), pp. 3684–3690. DOI: [10.1063/1.448118](https://doi.org/10.1063/1.448118).
- [8] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J. C Berendsen. “Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes”. In: 23.3 (Mar. 1977), pp. 327–341. DOI: [10.1016/0021-9991\(77\)90098-5](https://doi.org/10.1016/0021-9991(77)90098-5).
- [9] Ilario G. Tironi, René Sperb, Paul E. Smith, and Wilfred F. van Gunsteren. “A generalized reaction field method for molecular dynamics simulations”. In: *The Journal of Chemical Physics* 102.13 (Apr. 1, 1995), pp. 5451–5459. DOI: [10.1063/1.469273](https://doi.org/10.1063/1.469273).
- [10] Tim N. Heinz, Wilfred F. van Gunsteren, and Philippe H. Hünenberger. “Comparison of four methods to compute the dielectric permittivity of liquids from molecular dynamics simulations”. In: *The Journal of Chemical Physics* 115.3 (July 15, 2001), pp. 1125–1136. DOI: [10.1063/1.1379764](https://doi.org/10.1063/1.1379764).
- [11] W. Kabsch and C. Sander. “Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features”. In: *Biopolymers* 22.12 (Dec. 1983), pp. 2577–2637. DOI: [10.1002/bip.360221211](https://doi.org/10.1002/bip.360221211).
- [12] Harald Schwalbe, Shaun B. Grimshaw, Andrew Spencer, Matthias Buck, Jonathan Boyd, Christopher M. Dobson, Christina Redfield, and Lorna J. Smith. “A refined solution structure of hen lysozyme determined using residual dipolar coupling data”. In: *Protein Science* 10.4 (Apr. 2001), pp. 677–688. DOI: [10.1110/ps.43301](https://doi.org/10.1110/ps.43301).

9.2 Vienna-PTM 2.0

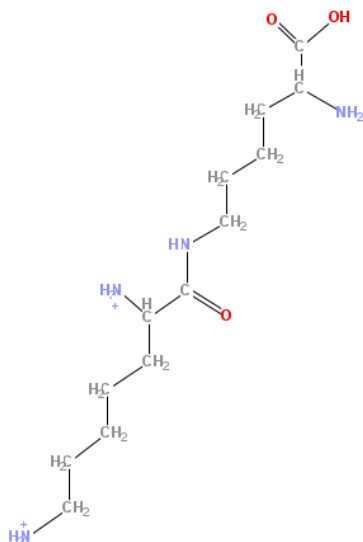
The simulation of post-translational modifications lags behind both the significance and occurrence of these chemical side-chain alterations. However, in recent years some developments took place in this area: besides the publication of Vienna-PTM [1] in 2013 (including parameters for the GROMOS force field [2]), a similar addition for AMBER has been established by Khoury and coworkers [3]. This section describes the changes and bug fixes done on the Vienna-PTM server since 2013 and shows the updates and additions that have been done to the parameter sets.

9.2.1 Improvements

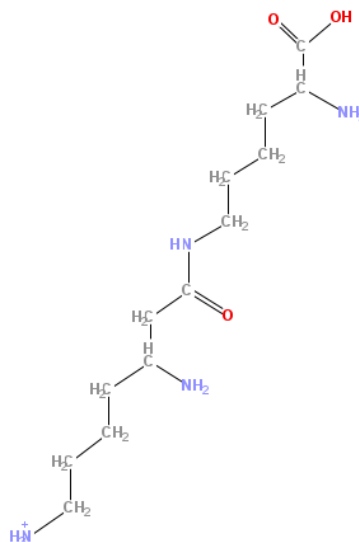
A number of residues has been changed during the last years (see the file "changelog.txt", which will be updated for all coming changes). A major change affected the widely used phosphorylations, using the 54A7 parameter set: accidentally, the torsions used for the phosphate-groups, were not set properly (GROMOS types 19 and 22 were used instead of the correct 20 and 27 torsions).

In addition to parameter fixes, some new building blocks have been proposed. Mostly, they have been requested by users over the last three years. The coverage of post-translational modifications is vast, but still incomplete and suggestions for additions are usually considered (if possible). The following new side-chains have not been reported before:¹

LYS $\xrightarrow{\text{LYSA}}$ KLA, α -lysyllysine



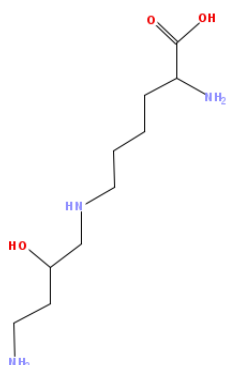
LYS $\xrightarrow{\text{LYSB}}$ KLB, β -lysyllysine



¹On the left, the precursor amino acid is given, the label on the arrow denotes the type of change and on the right the resulting residue is shown, both using its three-letter abbreviation and its chemical name.

LYS $\xrightarrow{\text{HYPR}}$ KHR, hypusine (R)

LYS $\xrightarrow{\text{HYPS}}$ KHS, hypusine (S)

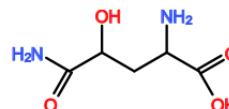


GLN $\xrightarrow{\text{HYDR}}$ Q4H,

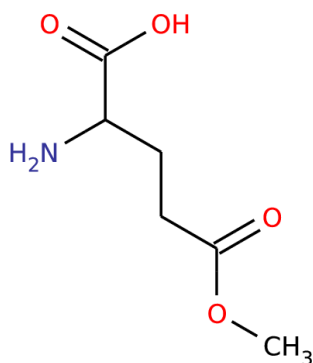
4-hydroxyglutamine (R)

GLN $\xrightarrow{\text{HYPS}}$ QH4,

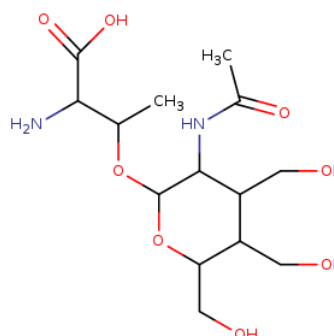
4-hydroxyglutamine (S)



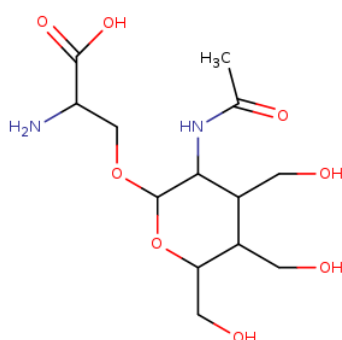
GLU $\xrightarrow{\text{METY}}$ EME,
glutamate methyl ester



THR $\xrightarrow{\text{OGLY}}$ TOG,
threonine-O-acetylglucosamine



SER $\xrightarrow{\text{OGLY}}$ SOG,
serine-O-acetylglucosamine



9.2.2 Interface changes

The server and its graphical user interface (GUI) have been subjected to major revisions:

- the packages were partially incomplete (sometimes the residue libraries were missing)
- the HTTP redirection failed sometimes when downloading files
- the Jmol applet [4] has a HTML5 basis now to avoid crashes
- a new GROMACS version caused problems with the terminal modification menu
- complete redesign of the homepage, with a WordPress [5] template as basis
- fixes when reading PDB files: many of the PDB files that users submitted for jobs had one or more serious violations of the PDB file format definition [6] leading to crashes; the server is now able to fix most of them automatically
- usage of asynchronous JavaScript (AJAX) to enhance user experience (e.g. by using the alertify plugin [7])

The interface itself has been changed to provide a more appealing site; see the following screenshots of the current and previous implementation.



Welcome

to **Vienna-PTM**, a resource for exploring protein post-translational modifications (PTMs) using molecular dynamics (MD) simulations! Here, you can modify a protein PDB file of your choice with one or more supported PTMs and obtain force field parameters (GROMOS [45A3](#), [54A7](#) [ref] and [54A8](#) [ref]) and input files needed to perform MD simulations of the modified proteins using the GROMACS package [ref]. Currently, we support a total of [260](#) different enzymatic and non-enzymatic modifications.

Figure 9.1: The new front page. There is, in addition to the top bar, a second navigation menu available on the left hand side (not shown). The upload form is at the bottom (not shown) and allows to upload PDB files from the users' hard drive as well as to give a PDB identifier in which case the denoted file is retrieved automatically.

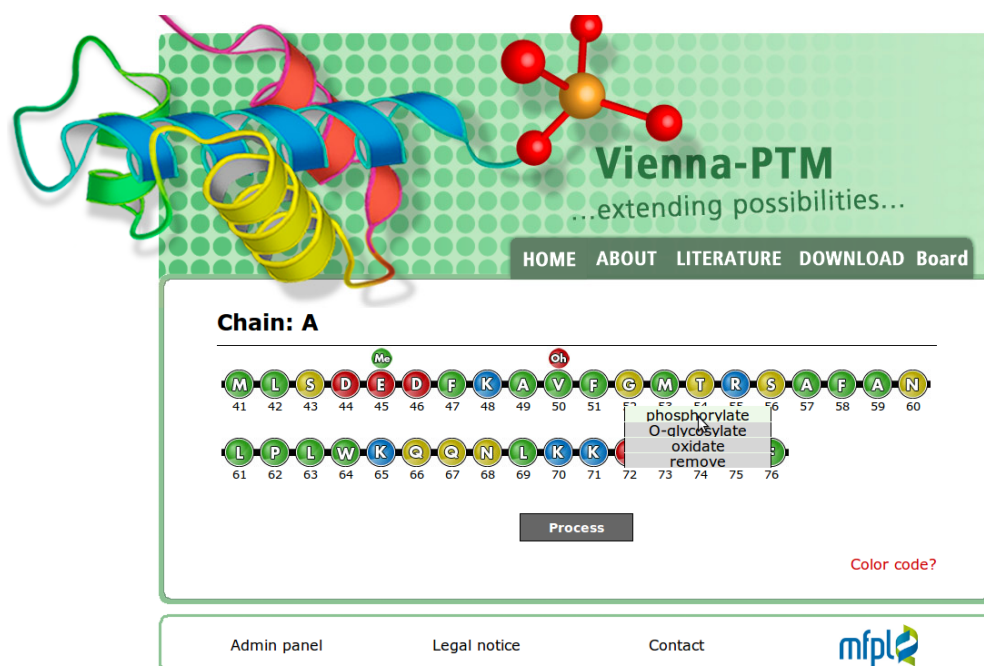


Figure 9.2: A screenshot of the old homepage (version 1.0) as described in the original publication [1]. A major criticism was the representation of the modifications as the actual process, the names of the resulting post-translationally modified amino acids were not shown. The new interface provides more information on the result.

Modification Menu

Chain: A

41 MET	42 LEU	43 SER	44 ASP	45 GLU	46 ASP	47 PHE	48 LYS	49 ALA	50 VAL	51 PHE
52 GLY	53 MET	54 THR	55 ARG	56 SER	57 ALA	58 PHE	59 ALA	60 ASN	61 LEU	62 PRO
63 LEU	64 TRP	65 LYS	66 GLN	67 GLN	68 ASN	69 LEU	70 LYS	71 LYS	72 GLU	73 LYS
74 GLY	75 LEU	76 PHE								

Legend:

... non-polar ... polar ... positive ... negative

Figure 9.3: New implementation of the modification menu. The jpeg-based "pearls-on-a-string" have been replaced by CSS-enhanced buttons. This allows to easily integrate new modifications into the webserver. By clicking on them, the overlay-menu shown in figure [9.4](#) opens.

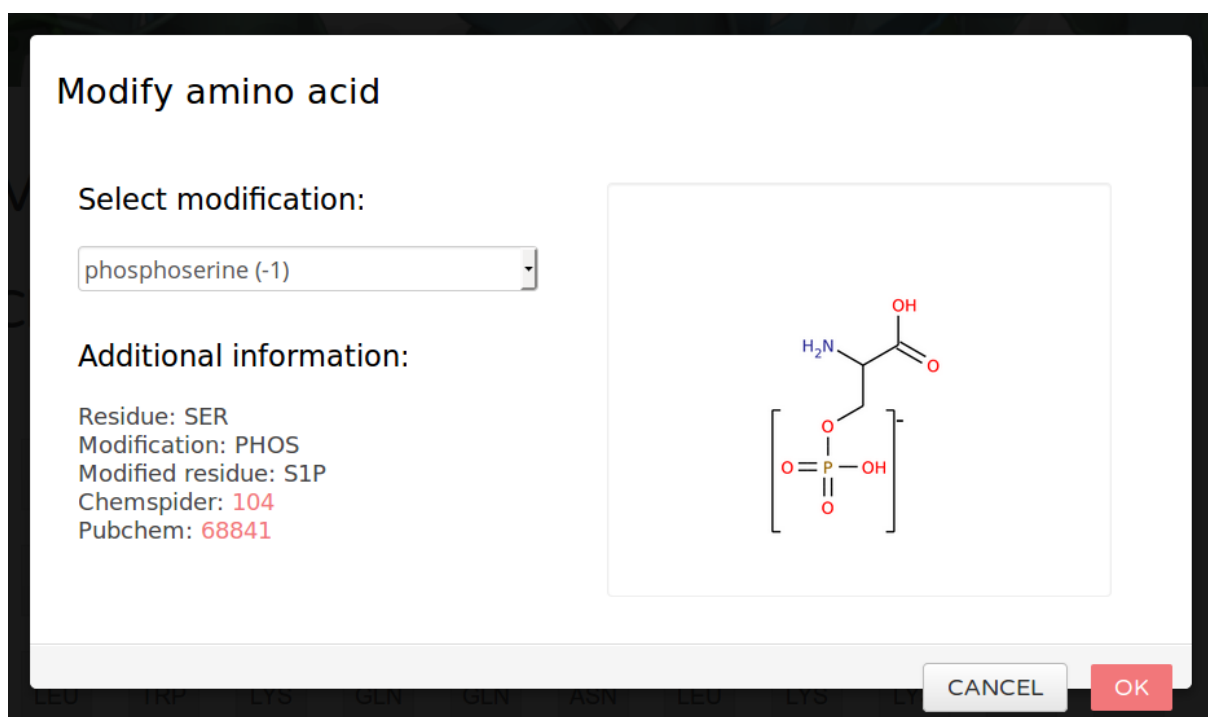


Figure 9.4: The modification menu for a selected amino acid (serine 56 in figure 9.3). By using the drop-down menu, post-translational modifications can be selected, which are graphically shown at the right hand side. The menu is different for every type of amino acid. The original residue, the Vienna-PTM internal modification code (see also the modifications listed on the homepage), the new residues' three-letter-code and (if available) the respective ChemSpider and Pubchem identifiers and their links are shown in the left-bottom part.

Modification Menu

Chain: A

41 MET	42 LEU	43 SER	44 ASP	45 GLU	46 ASP	47 PHE	48 LYS	49 ALA	50 VAL	51 PHE
52 GLY	53 MET	54 THR	55 ARG	56 S1P	57 ALA	58 PHE	59 ALA	60 ASN	61 LEU	62 PRO
63 LEU	64 TRP	65 LYS	66 GLN	67 GLN	68 ASN	69 LEU	70 LYS	71 LYS	72 GLU	73 LYS
74 GLY	75 LEU	76 PHE								

Legend:

 ... non-polar  ... polar  ... positive  ... negative

Figure 9.5: The selected changes are represented in the change of the abbreviation of the respective button (residue name) and a color code, explained at the bottom. The colors are selected according to the resulting compound, not the precursor (canonical amino acids are shown in gray). Clicking on "Process" starts the modification (and minimization, if selected) and forwards to the results page.

9.2.3 Impact

By July 2016, Vienna-PTM has been used by close to 1000 users, submitting roughly 7000 jobs since its launch three years earlier. The parameters have been downloaded about 3700 times (including all simulation engine parameter set pairs).



Figure 9.6: The IP-addresses from which jobs have been submitted, are widely spread over the world. However, "geolocation" is not perfectly accurate and sometimes leads to slightly shifted marks.

References

- [1] Christian Margreitter, Drazen Petrov, and Bojan Zagrovic. “Vienna-PTM web server: a toolkit for MD simulations of protein post-translational modifications”. In: *Nucleic Acids Research* 41.Web Server issue (July 2013), W422–426. DOI: [10.1093/nar/gkt416](https://doi.org/10.1093/nar/gkt416).
- [2] Drazen Petrov, Christian Margreitter, Melanie Grandits, Chris Oostenbrink, and Bojan Zagrovic. “A Systematic Framework for Molecular Dynamics Simulations of Protein Post-Translational Modifications”. In: *PLoS Computational Biology* 9.7 (July 2013), e1003154. DOI: [10.1371/journal.pcbi.1003154](https://doi.org/10.1371/journal.pcbi.1003154).
- [3] George A. Khoury, Jeff P. Thompson, James Smadbeck, Chris A. Kieslich, and Christodoulos A. Floudas. “Forcefield.PTM: Ab Initio Charge and AMBER Force-field Parameters for Frequently Occurring Post-Translational Modifications”. In: *Journal of Chemical Theory and Computation* 9.12 (2013), pp. 5653–5674. DOI: [10.1021/ct400556v](https://doi.org/10.1021/ct400556v).
- [4] Jmol development team. *Jmol: an open-source Java viewer for chemical structures in 3D*.
<http://www.jmol.org>.
- [5] WordPress Foundation. *WordPress*.
<http://www.wordpress.org>.
- [6] wwPDB. *The PDB file format version 3.2*.
<http://www.wwpdb.org>.
- [7] Fabien Doiron. *alertify.js - browser dialogs never looked so good*.
<http://fabien-d.github.io/alertify.js>.



Christian Margreitter

Curriculum Vitae

Schlachthausgasse 35/9 – 1030 Vienna, Austria

☎ + 43 680 3331119 ✉ christian.margreitter@gmail.com

Education

- 2012-present **PhD studies in Computational Biophysics**, *University of Natural Resources and Life Sciences*, Vienna.
- 2006-2012 **Studies in Molecular Biology**, *University of Vienna*, Vienna, *Master's degree*.
received with distinction (August 2012)
- 2007-2015 **Studies in Informatics**, *University of Vienna*, Vienna, rests.
- June 2005 **High School Diploma**, *Bundesgymnasium Bludenz*, Bludenz.

Publications

- [1] C. Margreitter* and C. Oostenbrink, "On the Optimization of the Protein Backbone Dihedral Angles by Means of Hamiltonian Reweighting," *J Chem Inf Model*, Aug. 2016.
- [2] C. Margreitter*, P. Mayrhofer*, R. Kunert, and C. Oostenbrink, "Antibody humanization by molecular dynamics simulations - in-silico guided selection of critical backmutations," *Journal of Molecular Recognition*, vol. 29, pp. 266–275, June 2016.
- [3] C. Margreitter*, D. Petrov*, and B. Zagrovic, "Vienna-PTM web server: a toolkit for MD simulations of protein post-translational modifications," *Nucleic Acids Research*, vol. 41, pp. W422–W426, 2013.
- [4] D. Petrov*, C. Margreitter*, M. Grandits, C. Oostenbrink, and B. Zagrovic, "A Systematic Framework for Molecular Dynamics Simulations of Protein Post-Translational Modifications," *PLoS Comput Biol*, vol. 9, p. e1003154, July 2013.

To be published

- [1] C. Margreitter* and C. Oostenbrink, "MDplot - an R plotting package for molecular dynamics analysis," *submitted*, 2016.
- [2] C. Margreitter*, M. M. Reif, and C. Oostenbrink, "Update on phosphate and charged post-translationally modified amino acid parameters in the GROMOS force field," *submitted*, 2016.
- [3] C. Margreitter* and C. Oostenbrink, "Post-translational modifications: effects on the backbone," *in preparation*, 2016.

Software development

- [1] "Vienna-PTM webserver: facilitates the post-translational modification of amino acid side-chains." <http://vienna-ptm.univie.ac.at>.
- [2] "MDplot - an R plotting package for molecular dynamics analysis." <http://cran.r-project.org/package=MDplot>.
- [3] "PROMETHEUS - automatization of molecular dynamics simulations."

Experience

- Nov 2012 **PhD studies**, *University of Natural Resources and Life Sciences, MMS, Vienna*, Group of Chris Oostenbrink.
 - present Developed an automated molecular dynamics simulation engine using templates and batch processing, worked on force field development regarding both charge distributions on ions and the optimization of the protein backbone angles ϕ and ψ , anti-body analysis and development of a binding prediction metric, continued development on a R plotting package for molecular dynamics simulations, investigations of the effect of post-translational modifications on amino acids.
- Apr 2011 **Diploma thesis**, *University of Vienna, MFPL, Vienna*, Group of Bojan Zagrovic.
- Jun 2012 Vienna-PTM: Establishment of a server extending simulation capabilities of proteins by post-translational modifications
- Feb 2011 **Internship**, *ETH Zurich, Zurich*, Group of Professor Matthias Peter.
- Mar 2011 Phospho-protein interaction study regarding *S. cerevisiae*'s Avo1p and Atg8p in the context of the *Cvt-pathway*
- Oct 2010 **Internship**, *University of Vienna, Vienna*, Group of Professor Gustav Ammerer.
 - Biochemical hands-on training on sample preparation and mass spectrometry-based analysis of phospho-proteins in *S. cerevisiae*
- Aug 2005 **Alternative Civilian Service**, *Kolpinghaus, Bregenz*.
- Aug 2006

Skills

Computer related

- Basic JAVA, bash scripts
- Advanced XML, DTD, UML, R, JAVASCRIPT, jQuery
- Expert C/C++, PHP, HTML, MYSQL, \LaTeX
- Tools etc. gimp, inkscape, GROMOS, wordpress, Windows, Office, Linux, pymol, MOE

Scientific

- Expert molecular dynamics simulations, free energy calculations (TI, OSP, Hamiltonian reweight), clustering, anti-body simulations and analysis

Languages

- German **Native**
- English **Fluent**

Qualities

Working independently, critical and analytical thinking, strength under pressure, reliable, assertive, communicative, eager to acquire new skills