universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Molecular Dynamics Simulation of Functionalized NU-1000 MOF

verfasst von | submitted by

Jonathan Netsch BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 862
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Chemie
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Chem. Dr. Christian Schröder

Acknowledgements

My deepest gratitude goes out to everyone who has supported me on this path. First and foremost, I would like to thank Christian Schröder and Marion Sappl, who supervised me throughout this thesis. My appreciation also goes to the whole team of the Computational Biological Chemistry department, especially Christian (Steel) and Maja for emotional support, Marko for IT support, and Mazin and Márta for help with professional questions. I would also like to sincerely thank DI Flora Schöfbeck from the Department of Functional Materials and Catalysis at the University of Vienna, who provided me with experimental data and was always patient in answering my questions. Furthermore, thanks to everyone who contributed to the StackOverflow.com online forum, providing hot fixes and comments for everyone stuck with errors or crippled graphs, as well as the GeeksforGeeks.org community.

I am grateful to the University of Vienna, the Faculty of Chemistry, and by extension to Austria, for providing logistics and financing my education. I deeply appreciate living in a country where education is accessible, and I am aware that it should not be taken for granted.

Nonetheless, nothing would have been possible without the love and support from my family. I am proud and grateful for my mother and father, who have always believed in me and motivated me to realize my potential. I thank my father's father - I miss our conversations on the balcony - and I am thankful for the time I had with my late grandmothers. I would also like to thank my cousin Pezi and wish him and Finja the best from the bottom of my heart.

Last but definitely not least, I want to extend my deepest gratitude to my friends. Thank you, Alex, who often joined me in rants and rambling. Thank you, Jazz - we share a similar burden, and I wish you success as well. I cannot forget Christoph, who drove from St. Pölten to Vienna to support me in a time of need, and many more times before and after. Finally love also goes out to my D&D group, for balancing out work and fun.

ᚨ

ᚹᛟᛏᚨᚾ·ᚷᛟᛞ·ᛟᚠ·ᛊᛗᚲᚱᛗᛏᛊ·ᚨᚾᛞ·ᚲᚾᛟᚹᛚᛖᛞᚷᛖ·ᛒᛖ·ᛈᚱᚨᛁᛊᛖᛞ

Abstract

Metal Organic Frameworks (MOFs) are a versatile group of porous materials based on inorganic metal clusters connected by organic linkers. NU-1000 is a zirconium-based MOF that was found to have interesting catalytic properties. A research group at the University of Vienna recently experimented with ligand modifications within the pores of NU-1000 and found that incorporating a photoswitch allowed them to trigger the alignment of the functionalized NU-1000 rods in an electromagnetic field using UV light. This phenomenon sparked the interest in a computational investigation, which led to this project.

Soft matter molecular dynamics simulations are a valuable asset for explaining mechanisms at the molecular level and identifying the contributions of specific species. For this thesis project, a force field for NU-1000 and the photoswitch spiropyran was adapted from literature and refined to set up molecular dynamics simulations in OpenMM. From the simulation data, infrared spectra and dielectric spectra were calculated and evaluated. To explore possible reasons for the alignment effect, the diffusion of ions and solvent within a modeled pore structure was analyzed. For each data set, the closed spiropyran form and the activated open merocyanine form were compared. Several coordination positions were tested to assess the effects of different spiropyran position combinations on the results. No significant difference was found between UV-activated and default photoswitch, but the slowing influence of the confined space within pores was successfully reproduced. The developed SNUFF force field provides a valuable foundation for future investigations.

Kurzfassung

Metallorganische Gerüstverbindungen (eng. abgekürzt MOF) sind vielseitige, poröse Materialien, die auf einer Kombination aus anorganischen Metallverbindungen und organischen Verbindungsstücken basieren. NU-1000 ist ein zirkoniumbasiertes MOF, das interessante katalytische Eigenschaften aufweist. Eine Forschungsgruppe der Universität Wien hat vor kurzem durch Modifikationen der Liganden innerhalb der Poren entdeckt, dass der Einbau photoreaktiver chemischer Schalter es ermöglicht, eine Ausrichtung der NU-1000-Stäbchen in einem elektrischen Feld mittels UV-Bestrahlung auszulösen. Dieses Phänomen hat das wissenschaftliche Interesse an computerbasierten Untersuchungen geweckt, was zu dieser Arbeit führte.

Molekulardynamische Simulationen weicher Materie stellen eine wertvolle Ressource zur Aufklärung von Mechanismen auf molekularer Ebene und zur Identifizierung spezifischer Beiträge einzelner Spezies dar. Für diese Arbeit wurde ein Kraftfeld für NU-1000 und seinen Photoschalter Spiropyran aus Literaturquellen adaptiert und optimiert, um molekulardynamische Simulationen in OpenMM zu erzeugen. Aus den Simulationsdaten wurden Infrarot- sowie dielektrische Spektren berechnet und ausgewertet. Um mögliche Ursachen des Ausrichtungseffekts zu identifizieren, wurde die Diffusion von Ionen und Lösungsmittel innerhalb der modellierten Porenstruktur analysiert. Bei jedem Datensatz wurde das geschlossene Spiropyran mit dem offenen, aktivierten Merocyanin-Form verglichen. Zudem wurden mehrere Koordinations-Positionen getestet, um den Einfluss der Spiropyran-Positionierung in der Pore zu erörtern. Unterschiede zwischen der UV-aktivierten und der Ausgangsform des Photoschalters waren nicht eindeutig feststellbar. Dennoch konnte der verlangsamende Effekt der Porenumgebung in der Simulation erfolgreich nachgewiesen werden. Das generierte SNUFF-Kraftfeld stellt eine wertvolle Grundlage für zukünftige Untersuchungen dar.

Contents

Acknowledgements i

Abstract iii

Kurzfassung v

1. Introduction 1
1.1. Modern Aspects of Material Design . . . 1
1.2. Motivation . . . 2
1.3. Metalorganic Frameworks MOF . . . 3
1.4. NU-1000 . . . 3
1.5. Spiropyran . . . 6

2. Theory 7
2.1. Molecular Dynamics . . . 7
2.2. Standard Forces . . . 7
2.3. Simulation Dynamics . . . 9
2.4. Computational Spectroscopy . . . 10
2.5. Self-Diffusivity . . . 12

3. Computational Methodology 15
3.1. Model Setup . . . 15
3.2. Force-Field Generation . . . 16
3.3. Simulation Parameters . . . 17
3.4. Trajectory Analysis . . . 18
3.5. Hardware . . . 20

4. Results and Discussion 21
4.1. Trajectory - Geometry and Stability . . . 21
4.2. IR Spectra . . . 21
4.3. Dielectric Spectra . . . 26
4.4. Diffusion Distribution within the Pores . . . 30
4.5. Discussion . . . 36
4.6. Limitations & Reliability . . . 37

5. Conclusion 39
5.1. Future outlook . . . 39

Bibliography 41

A. Appendix - Force Field: SNUFF 47
A.1. Atom Types . . . 47
A.2. Force Field XML File . . . 47

B. Appendix - Protein Data Bank Files 67
B.1. NU-1000 . . . 67
B.2. Spiropyran & Merocyanine . . . 70
B.3. Solvent and Ions . . . 73

C. Appendix - Python Scripts 77
C.1. Modeling . . . 77
C.2. Force Field Conversion . . . 81

C.3. Simulation 86
C.4. Trajectory Analysis 91

D. Appendix - Replica Results **117**
D.1. Infrared Spectra 117
D.2. Dielectric Spectra 121

1 | Introduction

1.1. Modern Aspects of Material Design

In recent years, the modern approach to material development has shifted towards a design concept, where compounds are specifically tailored to applications on a chemical level [1]. Simulations, data science, and machine learning have became valuable tools for this design process that greatly reduces time and material costs for experimental screenings [2–4]. Most notably, in drug design, screening programs are guided by specific target interactions and can identify the most likely candidates for experimental investigations [5]. In solvent chemistry, a rise in popularity of "designer solvents" such as ionic liquids has demonstrated many advantages of tunable components and emphasized the necessity of methods to accurately predict properties [6]. Similarly, in material sciences a rise in building block-based systems has had an enormous impact on the way material development is approached today. A building block approach opens the door to a wide variety of modifications and therefore to tunability. Plastic polymers are not without reason such a revolutionary invention, with seemingly endless possibilities to modify monomer building blocks and to vary reaction conditions to control the polymer weight distribution, thus leading to a vast landscape of properties [7].

Combining organic building blocks with inorganic components has led to the development of metal organic frameworks (MOFs). These compounds form periodic networks similar to zeolites, but exceeding them in surface area. Additionally, MOFs share the benefit of various modification options and tunability through building block alteration, as well as through post-synthesis modifications. The high porosity of MOFs makes them especially suitable for various technological implementations. Possible applications include gas storage or separation, catalysis, drug transport and more, putting MOFs in a strong position for energy technology innovations [8].

With the vast number of linkers, nodes and modification options for MOF design, the number of synthesized MOFs has increased at a record speed, while the number of possible structures still appears nearly limitless. Since it is impossible to synthesize and test millions of novel structures for a target application, the utilization of computational methods has been advancing in MOF development. Molecular Dynamics (MD) has enabled the prediction of bulk properties and suggested mechanisms behind certain behaviors observed during experiments [9]. Additionally, computer simulations provided the means to calculate various spectra based on collective properties, with the benefit of having all molecule position and orientation available. However, predicting MOF properties is still not a trivial matter. Established force fields are, generally speaking, typically parameterized for biological systems, and the vast landscape of possible MOF variants makes it difficult to find parameters for novel systems. This thesis documents the setup, analysis and interpretation of simulations of NU-1000, a metal organic framework, in the context of explaining its behavior in an electrical field.

1.2. Motivation

Zirconium-based MOFs exhibit interesting properties. NU-1000 is no exception, exhibiting remarkable stability at relatively high temperatures. NU-1000 was developed at Northwestern University by a research group led by Joseph Hupp and Omar Farha, giving the material its university-derived initials [10]. One remarkable trait is the structure's combination of relatively large pores mixed with smaller pores. Many works have focused on the catalytic capabilities of NU-1000, but additional modifications post-synthesis have also been discovered [11]. At the University of Vienna, a research group led by Jia Min Chin has worked on integrating a molecular UV switch, spiropyran, into the pores of NU-1000 (*confer* [12]). They observed that these functionalized NU-1000 rods align in an electrical field, thus allowing controlled orientation of the modified units. The possibility to alternate between two states is a vital and useful property for various applications. However, an unexpected behavior at high frequencies was observed. The Chin group observed alignment of the functionalized NU-1000 crystals after UV treatment even at several different frequencies, presumably due to the activation of spiropyran [13]. Surprisingly, at higher frequencies the NU-1000 rods align in the field regardless of the presumed activation state - in other words, regardless of UV treatment. It is unclear whether this behavior is caused by altered ion mobility or some other mechanism.
Molecular Dynamics can be a valuable tool in this case. Soft matter simulations of bulk solvent and solutions can be used to investigate the mechanics, the behavior of solutes and to calculate spectra, which can in turn be compared to experimental data. Solid phases, like the crystalline lattice of MOFs, can pose a challenge depending on the setup, due to their rigid, periodic structure. To investigate the movement of ions into and out of the pores, as well as compare diffusion coefficients, it was decided to model only a single pore in solution - partly also due to compatible setup options with OpenMM [14]. Open-source tools like OpenMM are advantageous due to the availability and their vast amount of accessible resources.

1.2.1. Purpose and Objective

This thesis presents comprehensive Molecular Dynamics simulations of functionalized NU-1000 Metal-Organic Frameworks (MOFs), including the interpretation of calculated spectral data, as well as the calculation and analysis of diffusion coefficients. This project was carried out in collaboration with Jia Min Chin's research group, who provided experimental data to investigate the mechanisms behind the behavior observed during their experiments and to potentially clarify the structural effects of the spiropyran group modification. To further the understanding of these effects, the objective of this thesis is to set up a soft matter simulation of a NU-1000 pore with the functional group in both its open and closed variants. To achieve this, the residues must be parameterized, or a force field must be adapted from literature, and simulations in OpenMM are produced. Additionally, several scripts for analyzing the generated trajectories are written, and the resulting datasets are finally interpreted.

1.2.2. Contribution

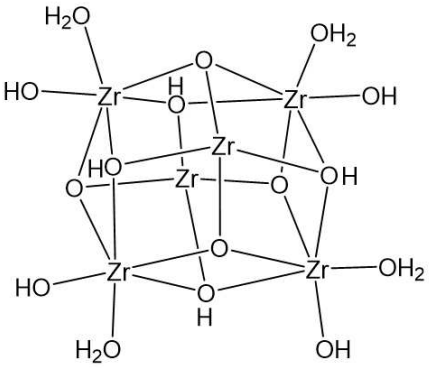
Stable simulations were created using the new, adapted force field for a single large pore structure surrounded by solvent molecules. Infrared and dielectric spectra were calculated for bulk solvent, unmodified, and functionalized NU-1000. The diffusion coefficients of residues inside the pore were compared to those outside the pore. This work represents a baseline for further simulation experiments and force-field refinement.

1.3. Metalorganic Frameworks MOF

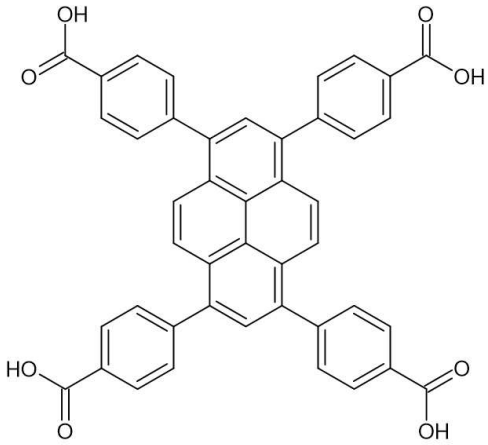
Metal-Organic Frameworks (MOF) consist of metal-containing inorganic units, called secondary building units (SBU) connected by organic linkers to form a periodic lattice. One of the most remarkable properties of these crystalline structures is their high porosity, which is related to their wide range of possible applications such as catalysis, gas adsorption, and separation [8]. Additionally, their unique properties make MOFs promising candidates for advances in functional materials for chemical sensing, drug delivery, and ion exchange. Due to the building-block composition of MOFs, there is a vast range of modification and tunability through different linker structures. Therefore, it is possible to adjust pore sizes to match the desired properties. In theory, the commonly applicable one-pot synthesis method is a major advantage. However, challenges such as ensuring purity, removing solvent guest molecules from the framework, and maintaining the integrity of the network structure can present significant obstacles [15].

1.4. NU-1000

The focus of this thesis is the simulation of NU-1000, a zirconium-based MOF, which is composed of $Zr_6(\mu_3-O)_4(\mu_3-OH)_4(H_2O)_4(OH)_4$ nodes linked by $4,4',4'',4'''$-(pyrene-1,3,6,8-tetrayl)tetrabenzoic acid [16]. The structures are depicted below in Fig. 1.1.



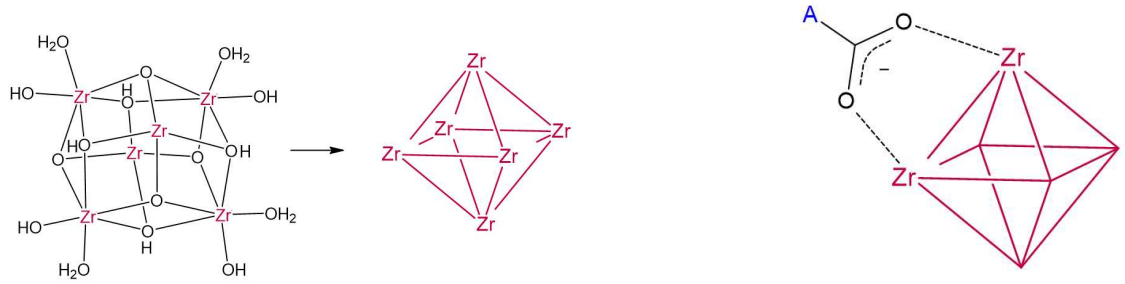
(a) Node: $Zr_6(\mu_3-O)_4(\mu_3-OH)_4(H_2O)_4(OH)_4$



(b) Linker: 4,4',4",4"'-(pyrene-1,3,6,8-tetrayl)-tetrabenzoic acid

Figure 1.1.: The structures of the NU-1000 node (a) and linker (b).

In NU-1000, each node contains six zirconium atoms arranged in an octahedral shape. Each zirconium atom can coordinate to four ligands, while the acid groups of the linkers interact with two neighboring zirconium atoms, one per oxygen (see Fig. 1.2). Due to the water and hydroxy groups on four of the zirconium atoms arranged in a plane, the resulting coordination leads to four linkers binding above the plane and four linkers below it, which results in the characteristic lattice pattern. The coordination of one node and eight linkers is depicted in Fig. 1.3 below.



(a) Simplification of the node only depicting zirconium positions.

(b) Coordination of one linker acid group.

Figure 1.2.: Simplified depiction of a node and the coordination of a single linker.

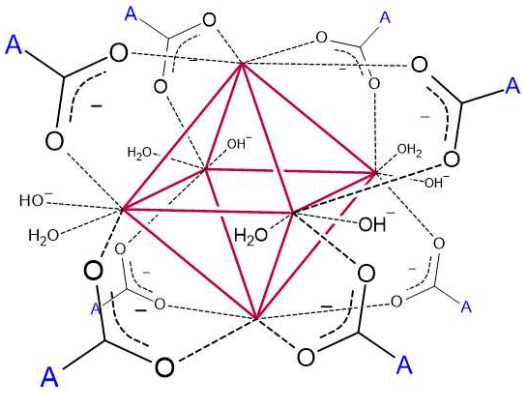
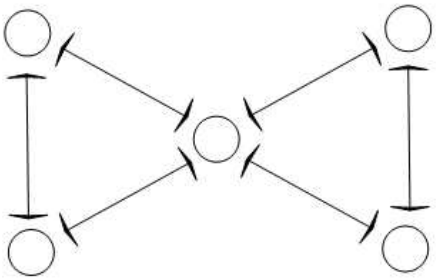
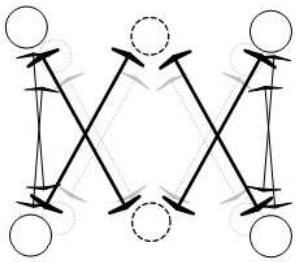


Figure 1.3.: Coordination of one node and eight linkers as found in NU-1000.

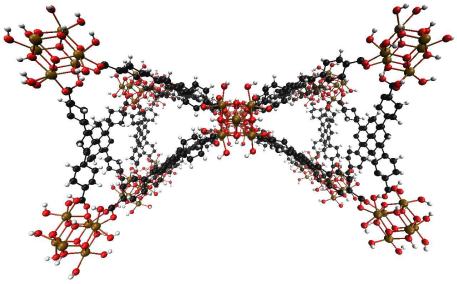
This directional coordination of eight linkers per node leads to hourglass-shaped pore-forming subunits, as shown in Fig. 1.4a and 1.4b below. These subunits form the larger main channels. The pore structure is a 2D trihexagonal tiling pattern, where hexagons are surrounded by equilateral triangles sharing their edges, which can also be called a Kagome network [17]. At each vertex is a zirconium oxide node that is eight-connected, and at each edge are the linkers, which is depicted in Fig. 1.4e. Additionally, there are two kinds of smaller pores orthogonal to the main channels, as shown in Fig. 1.4f.



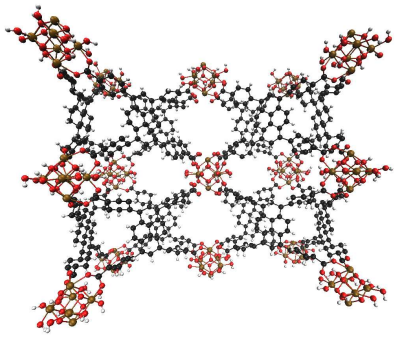
(a) Top view of the pore forming units.



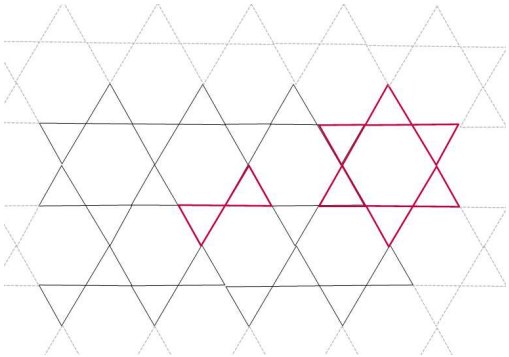
(b) Side view of the pore forming units



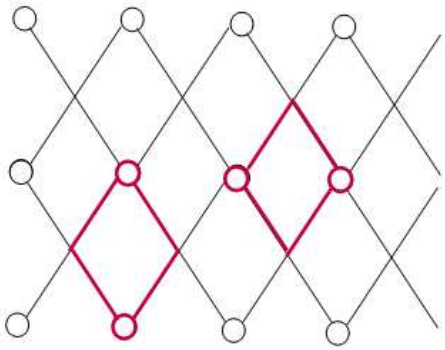
(c) Top view of the pore forming units in VMD.



(d) Side view of the periodic pattern formed by the subunits in VMD.



(e) Top view of the Kagome network with a pore forming subunit and a large pore surrounded by six small pores highlighted in red.



(f) Side view of the periodic pattern formed by the subunits. Two additional kinds of smaller pores at the side of the main channels are marked in red.

Figure 1.4.: (a,b) Simplified depictions of a pore forming subunit. Nodes symbolized by circles and linkers by arrows. (c,d) The two layer pore forming subunit visualized in VMD [18] (e,f) The periodic network formed by these subunits.

NU-1000 is a mesoporous, rigid crystalline polymer incorporating large pores with a size of 31 Å. The material has a high thermal and chemical stability, due to the strong bonds between nodes and linkers. It is a MOF with promising characteristics for gas separation or as a scaffold for heterogeneous catalysis, due to the Lewis acidity of the zirconium nodes, which create secluded reaction sites [19]. One example is the catalysis of styrene oxide to styrene carbonate for the production of polycarbonate monomers, although an isostructural hafnium variant of NU-1000 was discovered that exceeds its reaction rate [20]. Another notable possible application of NU-1000 is the destruction of chemical warfare agents by catalyzing the hydrolysis of phosphonate ester bonds. This concept was tested using the nerve agent simulant dimethyl 4-nitrophenyl phosphate and could help protect against agents containing phosphonate ester bonds, which are among the most toxic substances currently known [11].

1.5. Spiropyran

One of the advantages of NU-1000 is the option to modify it post-synthesis via solvent-assisted ligand incorporation (SALI). The exposed hydroxy groups protruding from the nodes can react with a carboxylic acid or phosphonic acid pendant group of another organic molecule [19]. In the context of this thesis, Jia Min Chin's research group utilized this method to functionalize NU-1000 with 3-(3',3'-dimethyl-6-nitrospiro[chromene-2,2'-indolin]-1'-yl) propionic acid, also known as spiropyran, for their experiments. Spiropyrans are photoswitches that can reversibly open their ring structure under UV irradiation and form their ring-opened isomer, merocyanine. This leads to a distinctive change in the isomer's properties, such as polarity, molecular volume, dipole moment, and emission behavior. The two ring systems in spiropyran are perpendicular to each other, preventing full π-electron delocalization. Merocyanine, on the other hand, forms a positively charged indolium and a negatively charged phenolate via C-O bond cleavage, which together form a coplanar zwitterionic extended conjugated system, as depicted in Fig. 1.5 below. While UV light promotes merocyanine formation, the reverse reaction is encouraged via visible light and/or heat, though solvents, ions, acidity, and even mechanical force can influence which isomer is favored [21].

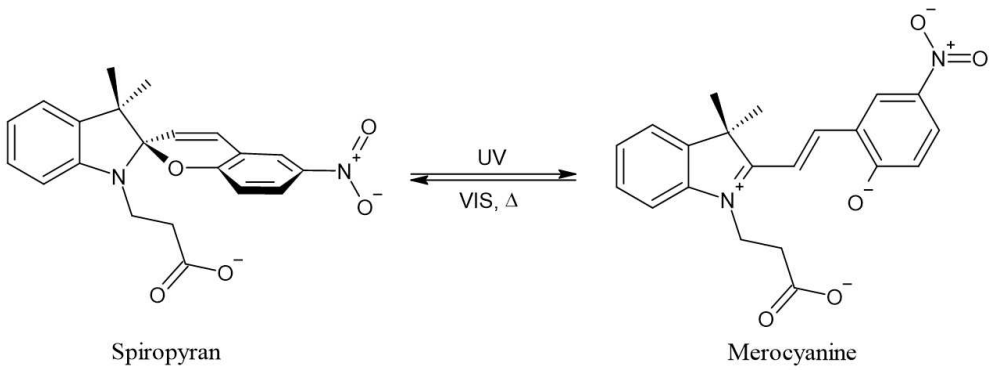


Figure 1.5.: Structure of spiropyran as the ring system opens with UV light and becomes the planar, zwitterionic merocyanine.

2 | Theory

This chapter provides an overview of the core mathematical and theoretical principles relevant to the simulations and calculations that were used to produce the results further discussed in the results chapter.

2.1. Molecular Dynamics

While advances in computer technology and software development have enabled scientists to solve the quantum mechanical calculations of systems with increasing size, it is still a very time-consuming and computationally expensive endeavor, with the major obstacle being accessing the electronic energy of a nuclear configuration. Utilizing the Born-Oppenheimer approximation, the hurdle of solving the electronic Schrödinger equation can be avoided by describing the electronic energy as parametric functions of the nuclear coordinates. Molecules are modeled as balls connected by springs, with Newtonian laws of motion applied according to empirical forces, neglecting quantum aspects of nuclear motion. Due to the classical mechanics applied to nuclear coordinates, these force field methods are called Molecular Mechanics (MM). These methods are efficient, albeit at the cost of accuracy due to the underlying approximations. A solution is to run multiple replications in parallel, allowing for statistical correction and leading to more reliable and correct results. The approximations for intramolecular and intermolecular forces are summed up to obtain the total energy of the system. Molecular Dynamics (MD) allows fast and cost-effective simulations of relatively large systems, though classical MD is limited by static molecule definitions and does not allow the simulation of reactions, polarization effects, charge transfer, or similar dynamic processes [22].

$$U(\vec{R}) = \sum U_{Bonds} + \sum U_{Angles} + \sum U_{Torsion} + \sum_{i}^{N} \sum_{j>i}^{N} [U_{LennardJones} + U_{Coulomb}] \tag{2.1}$$

2.2. Standard Forces

The description of intra- and intermolecular forces is a vital part of any simulation. It should be noted at this point, that different simulation software employ different mathematical approximations for these potentials. Each intramolecular force contains a constant K_X, while intermolecular Lennard-Jones interactions can be scaled via ϵ. OpenMM uses the following standard forces [14].

2.2.1. Intramolecular Potentials

Harmonic Bond Potential

Bonds between two atoms are described using the distance between particles r and the equilibrium distance r_0 in a harmonic potential.

$$U_B(r) = \frac{1}{2} K_B (r - r_0)^2 \tag{2.2}$$

This harmonic function according to Hooke's law is suitable to describe a state near the equilibrium at moderate temperatures. Otherwise, the anharmonic properties of the bonds and the dissociation energy must be considered by using the Morse potential. Under the simulation conditions of this work, the harmonic potential is applicable and reduces computational cost. It should be noted that the equilibrium distance r_0 and the force constant K_B are independent variables, although both are influenced by the types of the atoms involved, the hybridization state and neighboring groups [23].

Harmonic Angle Potential

Similar to the bond forces, the angle potentials are described using a harmonic equation incorporating the angle between the three atoms θ and the equilibrium angle θ_0.

$$U_A(\theta) = \frac{1}{2} K_A(\theta - \theta_0)^2 \tag{2.3}$$

The harmonic angle potentials primarily depend on the nature of the atoms forming the angle. Compared to bond potentials, they are generally weaker.

Periodic Torsion Potential

Proper periodic torsion, as well as improper torsions, are described with the same equation in OpenMM as periodic forces. In this case, the dihedral angle formed by four particles ϕ, the phase shift ϕ_0, and the periodicity n describe the energy contribution.

$$U_T(\phi) = K_T(1 + cos(n\phi - \phi_0)) \tag{2.4}$$

It is noteworthy that, for proper torsions, OpenMM defines the torsion angle ϕ as zero when the first and last atoms are on the same side of the bond formed by the two central atoms, that is in *cis* configuration. Multiple cosine functions are commonly applied, similar to a Fourier cosine series, to reproduce arbitrary torsional energy profiles.

2.2.2. Nonbonded

Lennard-Jones Interaction

The attractive and repulsive properties of the Lennard-Jones potential are described via the distance between particles r, scaled by ϵ.

$$U_{LJ} = 4\epsilon_{ij} \left(\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right) \tag{2.5}$$

Since different variations of this equation exist, it is vital to emphasize that σ, in this context, represents the distance at which the potential energy is zero. It can be derived from the center-to-center distance corresponding to the energy minimum, r_{min}, using the following relationship.

$$\sigma = \frac{r_{min}}{2^{1/6}} \qquad r_{min} = 2r_{vdw} \tag{2.6}$$

While r_{min} is the center-to-center distance at which attractive and repulsive parts of the potential are equal and cancel each other out, so that no net force acts on the particle, the Van der Waals radius, r_{vdw}, describes the radius of two imaginary hard spheres representing the atoms at the distance of their closest approach to one another. Mistaking r_{vdw} for σ would lead to a significant underestimation of the repulsive component of the interaction. This could result in particles approaching each other unnaturally closely, potentially destabilizing structures or even causing the simulation to terminate, see Discussion 4.6. Additionally, in the OpenMM force field, σ and ϵ are atom-type-specific parameters that are combined using the Lorentz-Berthelot mixing rule, as presented in Eq. 2.7 below.

$$\sigma_{ij} = \frac{\sigma_i + \sigma_j}{2} \qquad \epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j} \tag{2.7}$$

Coulomb Interaction

The interaction of charged particles is described using each atom charge q, their distance to each other r, as well as the electric constant ϵ_0, also referred to as the vacuum permittivity.

$$U_C = \frac{1}{4\pi\epsilon_0} \frac{q_i q_j}{r_{ij}} \tag{2.8}$$

When comparing the electrostatic Coulomb interactions to the Lennard-Jones potential, the former exhibits substantially greater range.

2.3. Simulation Dynamics

To perform the simulations, particle positions r_i must be updated based on their initial velocities v_i, mass m_i, and forces f_i acting on them, to generate new sets of positions and velocities after a timestep Δt. In this work, the Nosé-Hoover integrator was used to simulate the system's dynamics. The system variables are updated according to the following set of equations, *vide infra*.

$$v_i(t + \tfrac{\Delta t}{2}) = v_i(t - \tfrac{\Delta t}{2}) + f_i(t)\frac{\Delta t}{m_i} \tag{2.9}$$

$$r_i(t + \tfrac{\Delta t}{2}) = r_i(t) + v_i(t + \tfrac{\Delta t}{2})\frac{\Delta t}{2} \tag{2.10}$$

$$v_i'(t + \tfrac{\Delta t}{2}) = S \times v_i(t + \tfrac{\Delta t}{2}) \tag{2.11}$$

$$r_i(t + \Delta t) = r_i(t + \tfrac{\Delta t}{2}) + v_i'(t + \tfrac{\Delta t}{2})\frac{\Delta t}{2} \tag{2.12}$$

It should be noted that Eq. 2.11 includes a universal scale factor S, which is used for velocity adjustment and functions as the thermostat by employing a multi-timestep algorithm for heat bath propagation [24]. It is often desirable to maintain constant pressure throughout a simulation. The Monte Carlo algorithm creates a random trial volume change at set intervals and scales the box length and center of mass for each molecule accordingly. The new configuration is then evaluated based on the system's potential energy and target pressure via the Metropolis algorithm and is either accepted or rejected. This allows for relatively fast and cost-effective equilibration [25, 26].

2.4. Computational Spectroscopy

Spectroscopy is a routine analysis method in many areas and centers around the absorption, emission, or scattering of electromagnetic radiation by molecules. This includes a wide range of radiation types categorized by wavelength, though fundamentally there is no difference between high-energy gamma and low-energy radio wave radiation, and transitions are continuous. The division rather originates from the different experimental techniques developed over the years. Exposing a molecule to radiation can induce rotational, vibrational, or electronic changes, up to ionization processes, depending on the radiation energy [27].
The prediction of experimental spectra via Molecular Dynamics is a particularly relevant field for model validation, but can also provide insights into the structural contributions and relationships between effects in a complex system, thus helping to interpret experimental data. Additionally, the data can help in future material design [28].

2.4.1. Infrared Spectra Calculation

The interaction of sample materials with infrared light is based on structural vibrations. Therefore, it is also referred to as vibrational spectroscopy. Infrared light at about $100 - 5000\,\mathrm{cm}^{-1}$ excites molecular vibrations, increasing their vibrational amplitude. In an experimental setup, the intensity of the transmitted light through a sample is measured at different frequencies. The traditional scanning process is time-consuming, which is why modern Fourier transform infrared spectroscopy (FTIR) scans a wide spectral range simultaneously to produce high-resolution spectral data. Since functional groups have different characteristic transitions of energy states that can be observed via the absorption of infrared light, information about the structural components can be gained. The term 'fingerprint' describes the characteristic absorption pattern, depicted as a decrease in transmittance at wavelengths corresponding to the state transitions [29]. Infrared spectra can be calculated from simulation data through a Fourier transform of the autocorrelation functions of molecular dipole moments. An equilibrium time-correlation function f_{corr} is the ensemble average of a function f_a with the complex-conjugated form of a function f_b, indicated as f_b^*, see Eq. 2.13 below.

$$f_{corr}(t', t'') = \langle f_a(t') f_b^*(t'') \rangle \qquad with \qquad t' \geqslant t'' \tag{2.13}$$

The angular brackets represent either an ensemble average over initial conditions or an average over time. A definition of the latter is given in Eq. 2.14 below.

$$\langle f_a(t') f_b^*(t'') \rangle = \lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau f_a(t' + t) f_b^*(t'' + t) dt \tag{2.14}$$

The term autocorrelation is used when the two functions f_a and f_b are the same function. Additionally, the time origin t'' can be substituted for $t'' = s$, and the correlation function redefined as a function of only the time difference t, with $t' = s + t$, due to the fact that the equilibrium probability density is independent of time. Therefore, the correlation function is invariant under time translation. This also means the correlation function is stationary with respect to s, which is why it is common to set $s = 0$ and use the following notation [30].

$$f_{autocorr}(t) = \langle f_a(t + s) f_a^*(s) \rangle = \langle f_a(t) f_a^*(0) \rangle \tag{2.15}$$

Applying the autocorrelation and Fourier transform to particle velocities results in power spectra. However, in this case, vibrational spectra are obtained from molecular dipole moments. The autocorrelation of a time-dependent function $f(t)$ can be expressed using the following form of the Wiener-Khintchine theorem, which shows the connection to the frequency domain.

$$\langle f(0)f(t)\rangle = \frac{1}{2\pi}\int \left|\int f(t)e^{-i\omega t}dt\right|^2 e^{i\omega t}d\omega \tag{2.16}$$

With the time derivatives of molecular dipoles, the Fourier transform translates the expression to the frequency domain, producing a spectrum with corresponding infrared intensities. Note that $\vec{r}_i$ is the center of mass for a given molecule i, serving as a reference point to relate the position of all atoms α to in the expression of the dipole moment $\vec{\mu}$. [31]

$$A(\omega) \propto \int_{-\infty}^{\infty} \left\langle \dot{\vec{\mu}}(0)\dot{\vec{\mu}}(t)\right\rangle e^{-i\omega t}dt \tag{2.17}$$

$$with \qquad \vec{\mu}_i = \sum_\alpha q_{i\alpha}\cdot(\vec{r}_{i\alpha}-\vec{r}_i) \qquad and \qquad \dot{\vec{\mu}} = \frac{d\vec{\mu}(t)}{dt} \tag{2.18}$$

While this approach enables easy and fast calculation of infrared spectra, effects caused by anharmonic potentials, mode coupling, and the neglect of quantum contributions can lead to unrealistic band broadening and shifts [32]. The integration interval of the Fourier transform FT is from $-\infty$ to $+\infty$. In the context of infrared spectra calculation of this thesis, the function being integrated is even, which allows the use of the Laplace transform $\mathcal{L}$ instead, if the result is multiplied by a factor of two.

$$\mathcal{L}[f(t)] = \int_0^\infty f(t)e^{-i\omega t}dt = \frac{1}{2}FT[f(t)] \tag{2.19}$$

2.4.2. Dielectric Spectra Calculation

Moving to the THz region of the light frequency, the interactions shift to translational and rotational effects. Similar to the previous section, a variation of the total dipole moment autocorrelation can be utilized to calculate the dielectric spectrum. Using the frequency-dependent generalized dielectric constant $\Sigma(\omega)$, it is possible to express the dielectric susceptibility through application of linear response theory [33].

$$\Sigma(\omega) = \Sigma^*(\omega) - (\epsilon_\infty - 1) = \frac{4\pi}{3Vk_BT}\mathcal{L}\left[-\frac{d}{dt}\langle M_{tot}(0)\cdot M_{tot}(t)\rangle_{eq}\right] \tag{2.20}$$

The total collective dipole moment is the sum of charge times position over all atoms α and molecules i, and is similar to the molecular dipole moment given in Eq. 2.18.

$$M_{tot}(t) = \sum_i\sum_\alpha q_{i\alpha}\cdot\vec{r}_{i\alpha}(t) \qquad with \qquad q_i = \sum_\alpha q_{i\alpha} \tag{2.21}$$

The total collective dipole moment includes rotational and translational contributions. Therefore, the collective rotational dipole moment M_D, which is the part including any motion that is not translational, can be expressed as the difference between the total collective dipole moment M_{tot} and the collective translational dipole moment M_J, as given by the equation below.

$$M_D(t) = M_{tot}(t) - M_J(t) = \sum_i\sum_\alpha q_{i\alpha}(\vec{r}_{i\alpha}(t) - \vec{r}_i(t)) = \sum_i \vec{\mu}_i(t) \tag{2.22}$$

Translational effects stem from charged molecules moving relative to each other. For this reason, only charged species contribute to the collective translational dipole moment M_J. This relationship is illustrated in the following equation, as the summation over all atoms α for atomic charges within a molecule $q_{i\alpha}$ erases translational contributions if the molecular charge q_i results in 0, see Eq. 2.23 below.

$$M_J(t) = \sum_i \sum_\alpha q_{i\alpha} \cdot \vec{r}_i(t) = \sum_i q_i \vec{r}_i(t) \tag{2.23}$$

During a simulation, the position vector $\vec{r}_i$ can undergo dramatic changes when a molecule reaches the periodic boundary and jumps from one end of the simulation box to the other. Therefore, the derivative of $M_J(t)$, the collective dipole current $J(t)$, can be used in the previously established equations as follows. This methodology allows the usage of molecular velocities $\vec{v}$ instead of positions $\vec{r}$, which are not affected by boundary restrictions.

$$J(t) = \frac{dM_J(t)}{dt} = \sum_i q_i \cdot \vec{v}_i(t) \tag{2.24}$$

Analogous to Eq. 2.22, contributions to the time correlation function of the total collective dipole moment Φ_{tot} can be represented as the sum of the time correlation contributions of rotation Φ_D and translation Φ_J, as given below.

$$\Phi_{tot}(t) = \Phi_D(t) + \Phi_J(t) \qquad with \qquad \Phi_{tot} = \langle M_{tot}(0) \cdot M_{tot}(t) \rangle \tag{2.25}$$

However, it has to be stressed that the time correlation contributions do not simply consist of the respective autocorrelation of collective dipole moments, but also include a cross-correlation term of rotational and translational nature, as shown in the following equation.

$$\begin{aligned} \Phi_D(t) &= \langle M_D(0) \cdot M_D(t) \rangle + \langle M_D(0) \cdot M_J(t) \rangle \\ \Phi_J(t) &= \langle M_J(0) \cdot M_J(t) \rangle + \langle M_D(0) \cdot M_J(t) \rangle \end{aligned} \tag{2.26}$$

This is reflected in the frequency location of these contributions in the spectrum. Pure rotational $< M_D(0) M_D(t) >$ contributions can be predominantly found at lower frequencies in dielectric spectra, while purely translational $< J(0) J(t) >$ peaks reside at the higher-frequency end. The overlapping area between these is governed by cross-correlations of rotational-translational coupling $< M_D(0) J(t) >$, though those cross-term effects are situated closer to the translation peaks [33].

2.5. Self-Diffusivity

Based on Einstein's works on self-diffusion of Brownian particles, the Helfand equation for self-diffusion uses an equilibrium ensemble averaged over zero-time variables of the particle positions x_i to calculate diffusion constants [34].

$$D = \frac{1}{2t} \left\langle (\Delta x_i)^2 \right\rangle \qquad with \qquad \Delta x_i = x_i(t) - x_i(0) \tag{2.27}$$

The Mean Square Displacement (MSD) as described in Einstein's formula can be rewritten using center of mass coordinates $\vec{r}_i$.

Note that the ensemble average over time is denoted, as seen with the correlation function Eq. 2.13 above, while an additional average over all molecules i requires division by the total number of molecules of that species N.

$$MSD = \left\langle \frac{1}{N} \sum_{i=1}^{N} |\vec{r}_i(t) - \vec{r}_i(0)|^2 \right\rangle \tag{2.28}$$

Einstein's equation was adapted for the calculation of diffusion coefficients in the dimension d_α, combined with the MSD. The dimensionality d_α is 3 in the given simulation box, which leads to the final expression illustrated in Eq. 2.29 below [35].

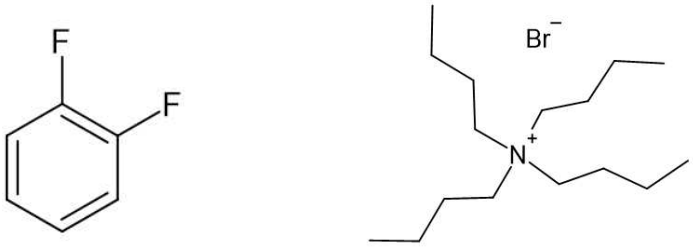
$$D = \frac{1}{2d_\alpha} \lim_{t \to \infty} \frac{d}{dt} \left\langle \frac{1}{N} \sum_{i=1}^{N} |\vec{r}_i(t) - \vec{r}_i(0)|^2 \right\rangle = \frac{1}{6} \frac{d}{dt} MSD \tag{2.29}$$

3 | Computational Methodology

This chapter illustrates the relevant methods applied during the simulations and calculations to produce the results discussed further in the Results chapter. Programs and tools are described here, with according references. Furthermore, the Python scripts mentioned, the base structure files, and the force field used can be found in the Appendix at the end.

3.1. Model Setup

OpenMM requires the force field as well as a structure/coordinate file, in this case a Protein Data Bank (PDB) file, to run a simulation. The PDB files were created by modeling the necessary residue structures - a single node, linker, spiropyran, merocyanine, and the solvent molecule *ortho*-difluorobenzene (DFB), as well as the ion pair tetrabutylammonium (TBA) and bromide (structures depicted in Fig. 3.1 below) - in Avogadro2 version 1.97.0 [36], followed by a geometry optimization utilizing Open Babel version 3.1.1 [37] before exporting the structures in a Protein Data Bank format.



(a) (b)

Figure 3.1.: Structures of (a) solvent *ortho*-difluorobenzene (DFB) and (b) the ion pair tetrabutylammonium (TBA) bromide.

The Avogadro2 output did not yet contain proper residue or atom labels, which needed to be added. The residue PDB files can be found in Appendix B. The larger pore structure was built by importing the required residues. This step was important due to the order in which Avogadro adds residues to the PDB file. This order was exploited to assign missing atom and residue labels, as well as residue numbers, using the order2pdb.py script, see Appendix C.1. The Python program assigns the correct variables in the PDB file based on the first atom of the residue list. In this manner, several variations of a single pore were built. Three node layers connected by two linker layers formed the pore structure, resulting in a total of 36 nodes and 36 linkers, which formed the large pore star shape depicted in Fig. 1.4e. Experimental data provided by the Chin group suggested that approximately three spiropyran molecules would occupy the constructed pore, based on weight percent. Since the distribution inside the pore is unknown, all three possible distributions over the six middle-layer node coordination positions were modeled using either spiropyran or merocyanine, see Fig. 3.2 below. The remaining coordination positions, where the pore structure would periodically continue, were filled with hydroxy groups. Another geometry optimization with Open Babel was conducted before exporting the PDB file.

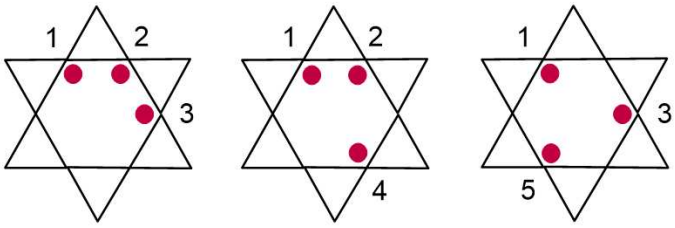


Figure 3.2.: Structure of the pore with spiropyran positions marked or every isomer with 3 groups.

The final simulation boxes were packed using Packmol version 20.15.0 [38]. Packmol creates a starting configuration by placing molecules randomly, while maintaining a minimum distance tolerance to each other, inside the box [39]. The simulation boxes had a side length of 120 Å and were packed with the tolerance set to 2 Å. Each simulation box contained one large, three-layered pore structure. The box was then filled with 5480 DFB molecules. Variations containing either 1 or 10 ion pairs were created to investigate diffusion behavior. Separately, a bulk solvent box was created containing only 5860 DFB molecules as a reference. All box dimensions were set to 120 Å for packing, as documented in the example input file packmol.inp attached in Appendix C.1. One final obstacle was presented by the node connectivity. The eight-coordinated zirconium was not inserted correctly by Packmol, thereby leaving only four bonds per zirconium atom in the final PDB file. This issue was resolved by either rerunning order2pdb.py or manually copying the connectivity of the correct pore PDB file.

3.2. Force-Field Generation

When creating MM simulations of any kind of matter, a good force field is essential for correct results. Availability of such parameters is often limited, which requires either parametrization via empirical methods - possibly mixed with *ab initio* QM calculations - or adaptation of existing force fields. Due to time constraints, the latter is often preferred, which is reflected in the surge of machine learning projects aiming to generate reliable parameters with less expensive calculations. In this thesis, parameters from existing force fields were adapted manually. For zirconium-based MOF parameters, several force fields are available. Aside from the Universal Force Field (UFF) [40] and the specialized UFF4MOF [41], the DREIDING [42], BTW-FF [43], DWES [44], CVFF [45], and TraPPE [46, 47] are all examples of force fields that have been used to describe zirconium-based MOFs. However, none of the published MOF force fields were compatible with the default OpenMM potentials, which were previously discussed in section 2.2, and many describe certain parameters using very different mathematical expressions. For further investigations, it was desirable to match the potentials to internally established force fields for the sake of compatibility. Translating the parameters into a usable form can be challenging when the corresponding equations are not provided and must be extracted from software documentation or other sources due to a lack of documentation. Because of this parametrization dilemma, other authors also used multiple force fields in similar works. One example is the use of a Dreiding force field as a basis for MOF linkers, with unavailable metal parameters taken from the UFF [48].

3.2.1. SNUFF

The **S**piropyran-functionalized **NU**-1000 **F**orce **F**ield (SNUFF) was created using MOF-specific and generalized force fields. A structurally very similar MOF sharing the same zirconium-based nodes is UiO-66, which has been investigated thoroughly with computational methods. A study presenting a UiO-66 force field applied to DL_POLY 2.20 simulations of the MOF and adsorbed gas molecules provided the base parameters for nodes and linkers [49]. For harmonic bond forces, the missing potentials for the hydroxy ligand oxygen and zirconium forces were assumed to be approximately equal to those of the linker ligand oxygen. The missing conjugated carbon-to-carbon forces were substituted using the provided parameters of the terephthalic acid ring carbon bonds. The UFF σ values for non-bonded interactions led to unstable simulations, in which the nodes would collapse. After refinement, the node oxygen and zirconium retained the original parameters, but were doubled and used for the force field, see Discussion 4.6. Since the non-bonded parameters of carbon and hydrogen from the original source compared closely to the parameters of the other residues, the original parameters remained unchanged.

Parametrization of the spiropyran, merocyanine, DFB solvent, and TBA structure was achieved by using the ACPYPE Portal. ACPYPE stands for **A**nte**C**hamber **PY**thon **P**arser interfac**E** and is designed to suggest parameters for unknown molecules. The program is based on Antechamber, an Amber package of auxiliary programs, which assigns force field parameters according to the Generalized Amber Force Field (GAFF). The parameters are provided in CHARMM format and need to be converted to the OpenMM .xml format to be usable. For this task, the convertff.py script based on ParmEd [50] was used, see Appendix C.2. The missing oxygen-to-hydrogen bond and angle parameters of the ligand hydroxy groups, as well as missing angle forces of the linker, were also taken from ACPYPE parameters [51, 52]. Since the nodes still lacked angle forces, some zirconium-oxygen-hydrogen angles were taken from an OpenBabel-optimized node structure [37]. For the bromide ion, the TIP3P [53, 54] standard parameters were used. The water parameters of the SPC/E [55] model were included for test runs.

Atom types and bonds included in the force field .xml file were obtained by running the pdb2xml.py Python script, which translates information from Protein Data Bank format input. Finally, charges were adapted to fit the original DL_POLY publication data. If the sum of pore residues did not balance out, the difference was evenly distributed across a residue molecule. The changes should not pose a significant alteration to the partial charges and their distribution. See Appendix A for the full set of SNUFF parameters.

3.3. Simulation Parameters

All simulations were carried out using OpenMM version 8.1.1, an open-source Python-based molecular dynamics library collection [14]. Two simulation modes were used: one with and one without a Monte Carlo barostat, as described in section 2.3 above. During an isothermal-isobaric NPT simulation with constant number of particles (N), pressure (P), and temperature (T), the Monte Carlo algorithm adjusts the volume to match the set conditions. The aim of running the NPT simulation first is to equilibrate the system. Subsequently, canonical NVT simulations with constant volume (V) instead of pressure were set up using the equilibrated system.

3.3.1. OpenMM Settings

The following settings listed in Tab. 3.1 remained unchanged for all simulations, unless specified otherwise.

Table 3.1.: Settings for all OpenMM simulations conducted. NB = Non-Bonded.

Temperature	300 K	Friction	$1\,ps^{-1}$
NB Method	PME	Constraints	H Bonds
NB Cutoff	1.2 nm	Const. Tolerance	10^{-6}
Ewald Tolerance	0.0005	H Mass	1.5 amu

For the NPT simulations, pressure was set to 1 atm, the Monte Carlo barostat interval was 25, and 1000 equilibration steps preceded the simulation. The following table summarizes all remaining options concerning time intervals, simulation length, and saved data points for all simulations. For NVT-3, which were the simulation runs used for the calculation of infrared spectra, the constraints were set to none. This was done to include the hydrogen effects in the infrared spectra calculated using that set of trajectories. Additionally, but less relevant in this context, ridgidwater was set to false.

Table 3.2.: Detailed simulation settings for each simulation phase. The data listed here is for one of ten trajectories produced per run.

Run	Δt	Steps/Traj.	Time	DCD Steps	DATA Steps
NPT	2.0 fs	50 000	0.1 ns	1000	1000
NVT-1	2.0 fs	500 000	1.0 ns	1000	1000
NVT-2	0.1 fs	5 000	0.5 ps	1	1000
NVT-3	0.1 fs	5 000	0.5 ps	1	1000
NVT-4	0.1 fs	25 000 000	2.5 ns	5000	5000

Each simulation was a succession of the runs listed in Tab. 3.2, while for each run, 10 consecutive trajectories were simulated, leading to a total simulation duration of about 36 ns, unevenly divided over 50 trajectories. The full example scripts npt.py and nvt.py can be found in Appendix C.3 at the end. While the NPT and NVT-1 runs described in the table above were mainly for equilibration purposes, the NVT-2 runs were used for constrained IR tests, which are of lesser relevance. The two main trajectories used for evaluations were NVT-3, the unconstrained IR trajectory, and NVT-4, the trajectory used for more dynamic phenomena, such as dielectric spectra and diffusion.

3.4. Trajectory Analysis

For a first rough estimation of the simulated structures, the pore size of NU-1000 was measured using the VMD version 1.9.4a57 OpenGL Display and the included measuring tool [18]. The distance was measured by selecting pore-facing zirconium atoms on opposite sides of the middle layer and observing the fluctuations over an NVT-4 trajectory.

3.4.1. Calculation of Spectra

The calculation of infrared and dielectric spectra was carried out using the analysis.py script provided in Appendix C.4. The script operates in two modes, each for a different spectrum type. During

IR mode, the NVT-2 or NVT-3 trajectories were used as input files. The code acquires the dipole moments by residue, creates the sum of the total dipole moment, as well as the sum of the MOF and spiropyran contributions, and calculates auto-correlations of these sums. To access the infrared spectrum, a gradient function is followed by a Laplace transform. Since the resulting spectrum is quite noisy, it is necessary to smooth it with a Savitzky-Golay (SavGol) filter included in the script.

The dielectric spectrum is created using the NVT-4 trajectories in a similar manner, by taking the dipole moment by residue and calculating the cross-correlation with the total dipole moment. Then, the fit function MDMD_fit.py is applied to the cross-correlations, and the fit variables τ are saved in a MDMD_fit.json file. For the contribution of ions, the MSD of M_J, based on the center of mass of the involved ions, is also calculated and serves as input for the fit function msdMJ_fit.py, which similarly saves the fit variables τ in a msdMJ_inp.json file. The saved .json files were used as input for the final calculation of the dielectric spectrum with GenDiCon version Lithium.

GenDiCon was developed by the Schröder group at the University of Vienna. The program converts the raw time correlation and fit-function-derived data to dielectric spectra [33]. The Python script write_gdc_inp_file.py was used to generate the GenDiCon input script and is included in Appendix C.4.

3.4.2. Confined Space Analysis

To properly assess the impact of the environment within the NU-1000 pore and compare it to the bulk or surface environment outside the pore, the trajectories were analyzed using the poreVSbulk.py script, see Appendix C.4. The script creates a 15 Å radius sphere in the center of the MOF pore, which is then used to select all atoms within the pore. Due to the fact that, over the duration of the trajectories, some molecules enter or leave the pore volume, the molecules are separated into three groups, namely those that remain inside, those that remain outside the pore over all the examined frames, and those that travel between the two defined spaces. The diffusion distribution functions were calculated for each of the three groups, and diffusion coefficients were evaluated according to Eq. 2.29. The script creates a linear regression of the calculated MSD and uses the slope function to compute the time derivative, which is multiplied by $\frac{1}{6}$ to obtain the diffusion coefficient. For statistically representative results, three replicas, each with ten 2.5 ns trajectories of the NU-1000 in DFB system with ten ion pairs, were used - *id est*, the trajectories used for the dielectric spectrum analysis. The unit of diffusion coefficients is m^2/s, which is noted to clarify the conversion from MSD data given in ps or 10^{-12}s. However, while OpenMM operates on the nm scale, which corresponds to 10^{-9}m, MDAnalysis automatically translates the data to Å, equivalent to 10^{-10}m. This results in the produced coefficients being given in $10^{-8}m^2/s$ as the unit.

3.4.3. Core Libraries for Data Analysis

The trajectory analysis scripts require the following core packages. For the calculation of the spectra, MDAnalysis [56, 57] and NewAnalysis, an extension to MDAnalysis developed by the Computational Biological Chemistry department at the University of Vienna, are required. These packages are necessary to calculate the correlation functions and run the scripts mentioned above. Additionally, NumPy [58] and SciPy [59] functionalities are used. For the calculation of the MSD, the codes use the Tidynamics [60] package. Finally, all figures of spectra and the plots of diffusion coefficients were made using Matplotlib [61] and Seaborn [62].

3.5. Hardware

The simulations were all run on the department's cluster with the cluster head RS10, on GPU partitions with GPU acceleration. The cluster has 15 GPU nodes and 4 CPU nodes available. While most simulations took only about an hour per trajectory, the nvt4 runs were far more time-consuming, with simulation times reaching over 20 h per trajectory, thus requiring a total simulation time of 3 days to a week. Similarly demanding was the trajectory analysis of the confined space within the pore, which took about a week using a cluster CPU partition. An example Slurm [63] script run.sh for running the simulations on the cluster is provided in Appendix C.3 at the end of the document.

4 | Results and Discussion

In the following chapter, the simulation analysis results are presented, including a discussion of the findings, which place the produced data into context with experimental data and literature. Due to the amount of produced data, additional spectra for each replica are provided in Appendix D. This section presents the calculated infrared spectra, dielectric spectra, and diffusion coefficients, as calculated according to the procedures described in section 3.4 Methods before. A summary and discussion are provided at the end of this chapter.

4.1. Trajectory - Geometry and Stability

The pore sizes measured in the middle layer, from pore-facing zirconium atom to the opposite pore-facing zirconium atom, fluctuated during simulations around approximately 32 Å in all simulations. This compares very well to the 31 Å documented in literature [19], which is one indicator that the simulations can fundamentally reproduce a similar environment to real NU-1000 pores. The following depiction, see Fig. 4.1, shows the measurement tool in VMD.

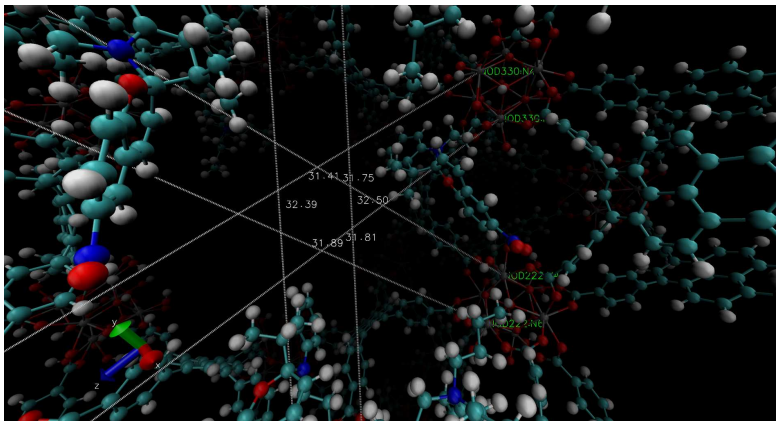


Figure 4.1.: Pore measurement in VMD using opposite, pore-facing zirconium atoms as reference points.

4.2. IR Spectra

For all simulations, the infrared spectra were calculated as previously described in section 3.4.1. In the following section, the results of the bulk DFB solvent, NU-1000 in DFB, and spiropyran/merocyanine-modified NU-1000 in DFB with ten TBABr ion pairs are presented. Additionally, the total spectra are compared to experimental data, and similarities discussed.

4.2.1. Bulk DFB and NU-1000 in DFB

Due to the ratio of solvent to MOF, the following spectra show dominant solvent peaks, as demonstrated in Fig. 4.2, which shows both infrared spectra of the bulk DFB solvent and the single unmodified NU-1000 pore in solution. To highlight the difference, the MOF contribution to the infrared spectra was included in the depiction. Notable solvent peaks can be found at approximately 2520 cm^{-1}, 1790 cm^{-1}, 1705 cm^{-1}, 1590 cm^{-1}, 1515 cm^{-1}, 1300 cm^{-1}, 1210 cm^{-1}, 1025 cm^{-1}, 950 cm^{-1}, 835 cm^{-1}, 755 cm^{-1}, 675 cm^{-1}, and 430 cm^{-1}.

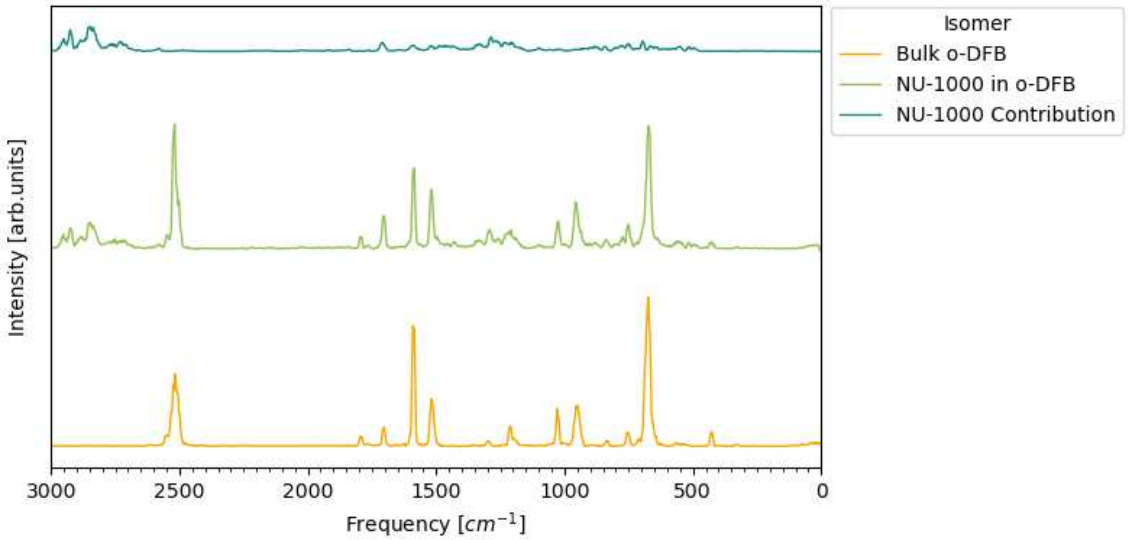


Figure 4.2.: Mean IR spectrum of bulk DFB (yellow) versus mean IR spectrum of NU-1000 in DFB (green) with the MOF contribution depicted separately as well (mint).

The MOF contributions tend to overlap more, creating multiple peaks between 2955 cm^{-1} and 2710 cm^{-1}, from 1515 cm^{-1} to 1410 cm^{-1}, at 1340 cm^{-1} and 1195 cm^{-1}, and from 945 cm^{-1} to 495 cm^{-1}, with a few distinguishable peaks at 2580 cm^{-1}, 1705 cm^{-1}, 1590 cm^{-1}, 1100 cm^{-1} and 1025 cm^{-1}. It is notable that the solvent peak at 2520 cm^{-1}, as well as the multiple peaks caused by the MOF at 2710 cm^{-1} and above, are caused by hydrogen bonds and are not visible in the simulations with constrained hydrogen bonding.

4.2.2. Functionalized NU-1000

The functional spiropyran group did not impact the total infrared spectra in any immediately apparent way. The mean spectra of the different coordination-position isomers are given in Fig. 4.3 and show negligible differences between isomers. The spectra are virtually identical to the NU-1000 in DFB spectrum described above.

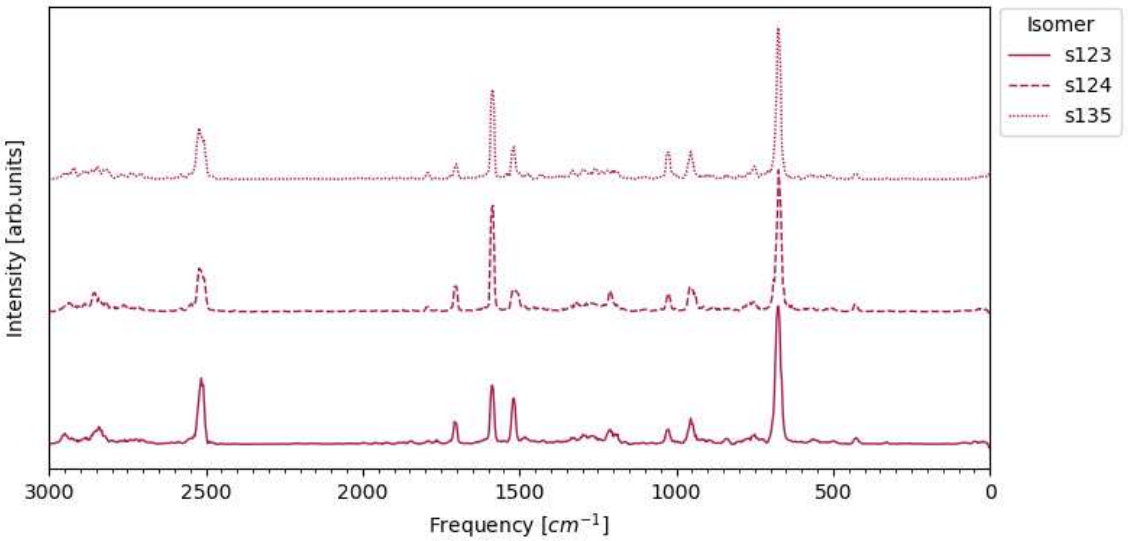


Figure 4.3.: Mean IR Spectra of all spiropyran coordination position isomers.

While the overall peak composition of the spiropyran spectra seem to be the same, there are still differences in the relative intensities of the peaks. One example is the peak at 1590 cm^{-1}, which has a lower maximum for the s123 isomer than for the other two. Similar small differences can be attributed to statistical fluctuations combined with deviations originating from the SavGol filter.
The open merocyanine form had equally little impact on the total IR spectra, as illustrated in Fig. 4.4 below. Both functionalized MOF versions, closed-ring or open-ring, differ very little in their total spectrum.

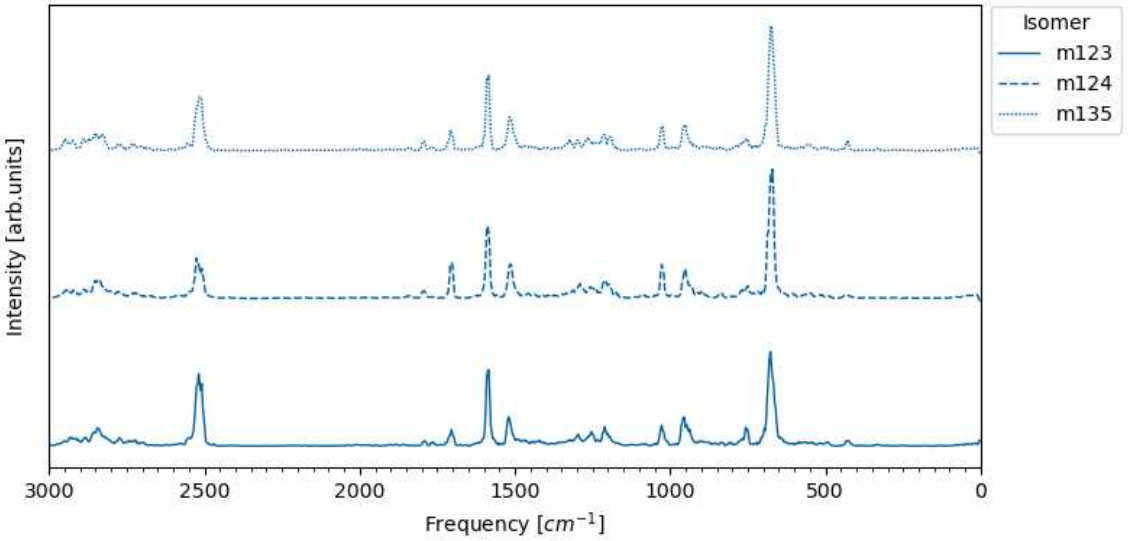


Figure 4.4.: Mean IR Spectra of all merocyanine coordination position isomers.

Mirroring the results for spiropyran, the total spectra of merocyanine-functionalized NU-1000 isomers merely include small deviations in peak intensities, *exempli gratia*, m123 having two small peaks at 1790 cm^{-1} and 1765 cm^{-1}, while the other isomers show one slightly more prominent peak with approximately double the relative intensity at 1790 cm^{-1}.
To further underline the similarity of the calculated IR spectra of functionalized MOF, each spectra averaged over all coordination isomers was overlayed with a small off-set, see Fig. 4.5

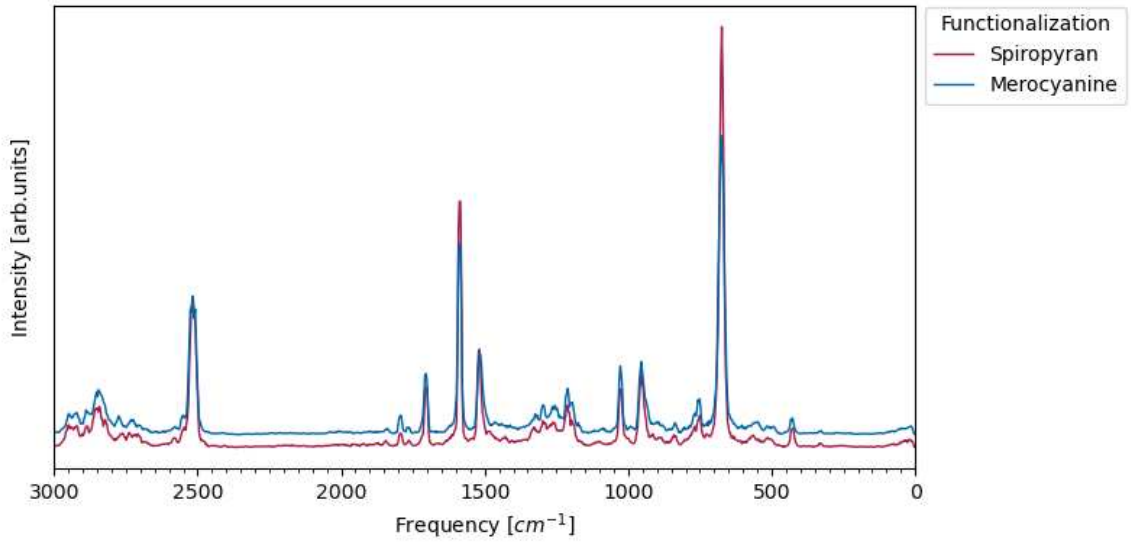


Figure 4.5.: Mean IR Spectra averaged over coordination position isomers for spiropyran (red) and merocyanine (blue).

The averaged spectra include the previously identified solvent peaks at 2520 cm^{-1}, 1790 cm^{-1}, 1705 cm^{-1}, 1590 cm^{-1}, 1515 cm^{-1}, 1300 cm^{-1}, 1210 cm^{-1}, 1025 cm^{-1}, 950 cm^{-1}, 835 cm^{-1}, 755 cm^{-1}, 675 cm^{-1}, and 430 cm^{-1}, as well as the peaks, which were identified as MOF contributions in the first section. Again multiple peaks between 2955 cm^{-1} and 2710 cm^{-1}, some around 1480 cm^{-1} and 1410 cm^{-1}, between 1330 cm^{-1} and 1195 cm^{-1}, and between 945 cm^{-1} and 495 cm^{-1} can be found, with a prominent peak at 1295 cm^{-1}.
However, as with the MOF contribution, the contribution of the spiropyran group was isolated and used for a more meaningful interpretation of the difference between the two forms when comparing calculated and experimental data in the following section (see Fig. 4.8).

4.2.3. Experimental Data

The experimental data provided by Flora Schöfbeck of the Chin group was obtained using an FTIR system called Tensor 37 (Bruker). The sample consisted of dried MOF powder. First, the experimental data for NU-1000 without modification was compared to infrared spectra derived from simulations, see Fig. 4.6 below. To discern peaks originating from NU-1000, the MOF contribution to the calculated spectra was included in the graph, with scaling applied for improved visibility.

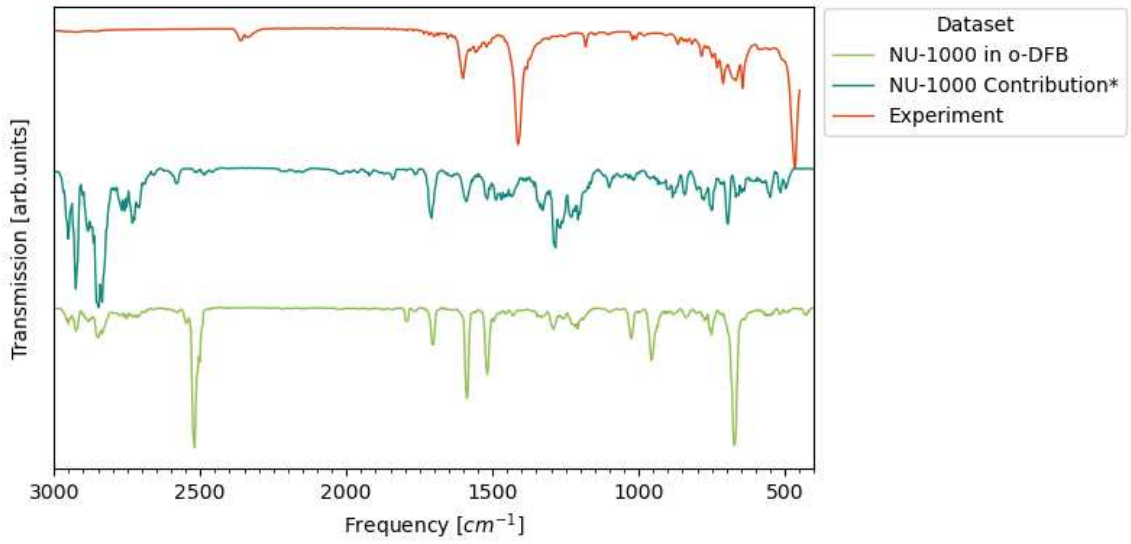


Figure 4.6.: Experimental (orange) and simulated (green) infrared spectra for NU-1000, plus the separated NU-1000 contribution (mint). *The MOF contribution was scaled for a better comparison with experimental data.

The experimental spectrum includes major peaks at 2360 cm^{-1}, 2340 cm^{-1}, 1600 cm^{-1}, 1555 cm^{-1}, 1520 cm^{-1}, 1415 cm^{-1}, 1180 cm^{-1}, 1025 cm^{-1}, 1010 cm^{-1}, 785 cm^{-1}, 710 cm^{-1}, 675 cm^{-1}, 645 cm^{-1}, and 465 cm^{-1}. While there is considerable overlap between some peaks from simulation results, for instance the peak at about 1600 cm^{-1} and multiple peaks between 710 cm^{-1} and 645 cm^{-1}, other peaks appear to be either shifted or absent.
Two additional experimental data sets were provided, one with a spiropyran-functionalized sample exposed to white light (WL), and the other to ultraviolet light (UV), representing the merocyanine form. The three experimental data sets are depicted in Fig. 4.7 below.

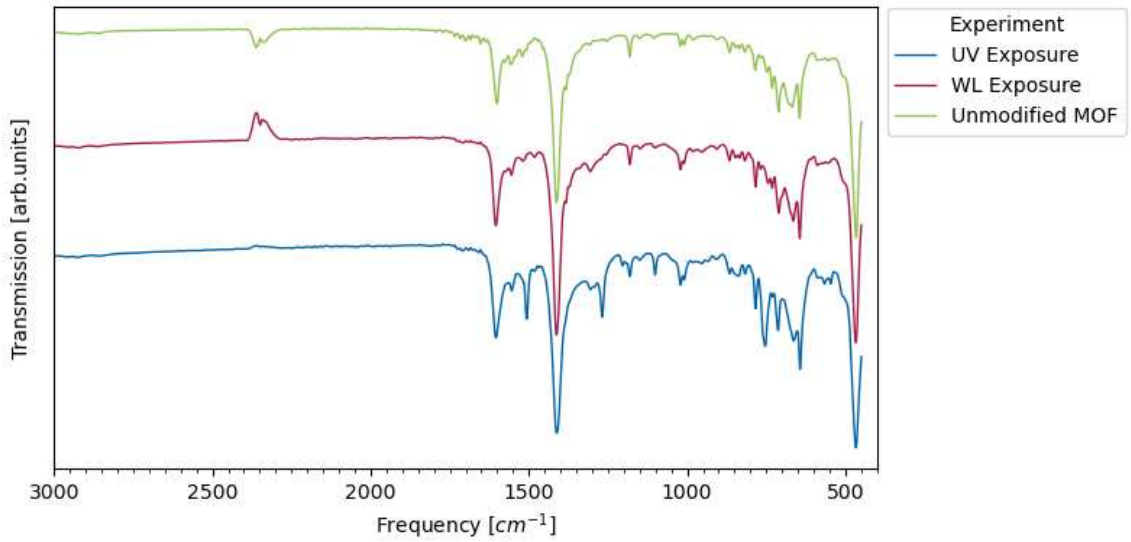


Figure 4.7.: All experimental infrared spectra. Unmodified NU-1000 (green), functionalized MOF after WL exposure (red), functionalized NU-1000 after UV treatment (blue).

The comparison of these experimental datasets reveals prominent peaks that appear exclusively in the UV-treated sample. The most significant occur at 1505 cm^{-1}, 1270 cm^{-1}, 1205 cm^{-1}, 1100 cm^{-1}, and

755 cm^{-1}. Additionally, three smaller peaks at 590 cm^{-1}, 565 cm^{-1}, and 545 cm^{-1} significantly increase in intensity for the UV sample. The behavior around 2360 cm^{-1} and 2335 cm^{-1} remains unclear, since unmodified NU-1000 exhibits some absorption in the region, but the spiropyran-functionalized version shows an unnatural negative peak that almost entirely disappears in the UV dataset, with only a faint deviation remaining. Utilizing one core advantage of computational methods, the contributions of the inserted functional groups were isolated and plotted in Fig. 4.8. This was necessary because the simulated total infrared spectra are dominated by solvent and MOF contributions, which can obscure subtle but relevant changes caused by the ring opening.

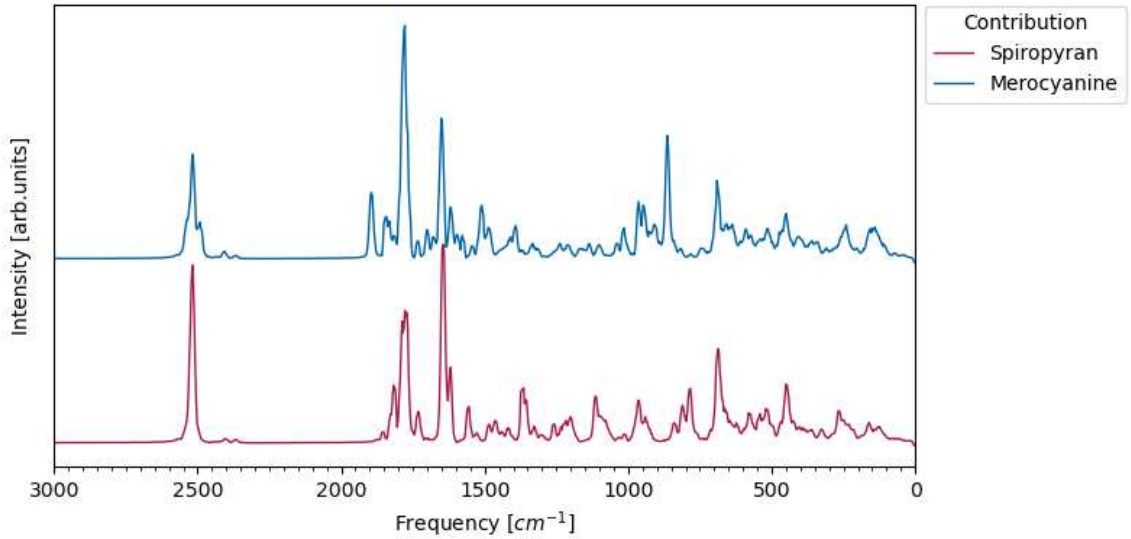


Figure 4.8.: Compared infrared contributions of spiropyrane (red) and merocyanine (blue) simulations.

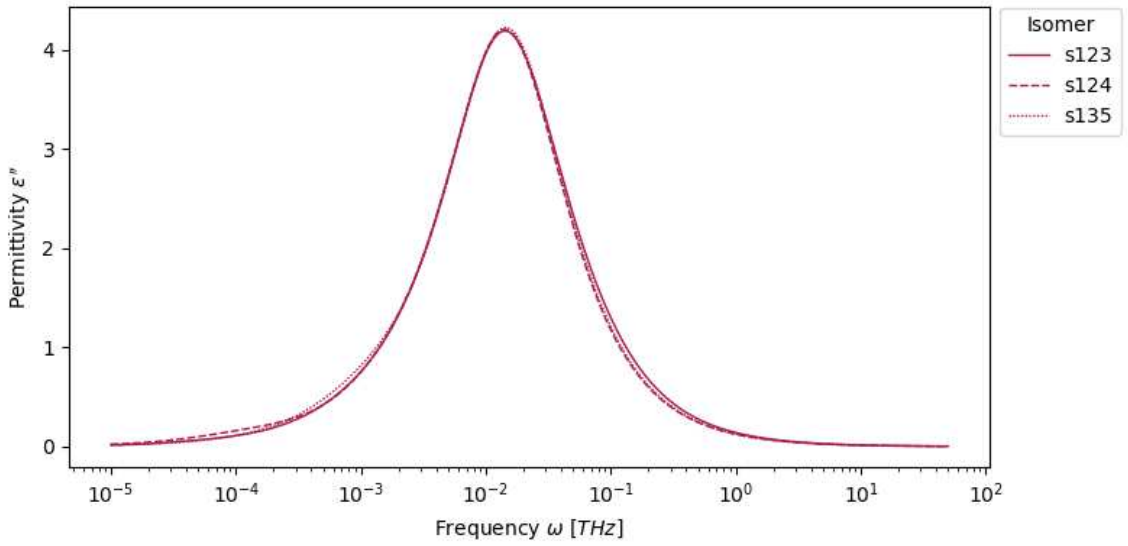
The split fractions reveal that the infrared spectrum of merocyanine, which parallels the UV data, contains additional peaks at 2490 cm^{-1}, a major one at 1895 cm^{-1}, 1845 cm^{-1}, and four smaller peaks at 1700 cm^{-1}, 1680 cm^{-1}, 1595 cm^{-1}, and 1575 cm^{-1}, as well as peaks at 1510 cm^{-1}, 1395 cm^{-1},910 cm^{-1}, and a major peak at 865 cm^{-1}. Unexpected additional peaks, exclusive to the spiropyran contributions, corresponding to the WL-exposed sample, appear in five positions at 1555 cm^{-1}, 1365 cm^{-1}, 1110 cm^{-1}, 815 cm^{-1}, and 785 cm^{-1}. Although one of the characteristic UV peaks in the experimental infrared spectra was matched exactly at 1505 cm^{-1}, and several extra peaks were visible in the merocyanine contribution spectrum as expected, their positions deviated from those observed experimentally.

4.3. Dielectric Spectra

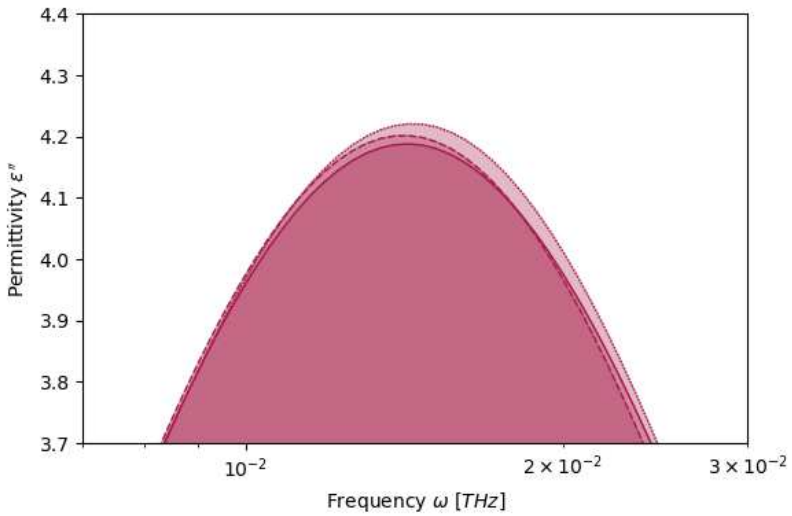
Similar to the previous section, the results of dielectric spectra calculations from simulations are presented in the following passage. The first set of results discussed is the total dielectric spectra of all positional isomers of functionalized NU-1000. Then, the total spectra of the unmodified MOF and the DFB bulk solvent are used to contrast the average spectra of the spiropyran and merocyanine isomers. Finally, the contributions of the functional group are examined independently to identify possible correlations. For all spectra, only the imaginary part ϵ'', was plotted.

4.3.1. Functionalized NU-1000 Coordination Isomers

The total dielectric permittivity spectra of spiropyran-modified NU-1000 for all three coordination position isomers are shown in the following Fig. 4.9, and exhibit a major peak at $1.4 \cdot 10^{-2}$ THz. A closer inspection of the peak maxima reveals that all coordination position isomers produce very similar results. The value of ϵ'' was 4.19 for m123, 4.20 for s124, and 4.22 for s135.



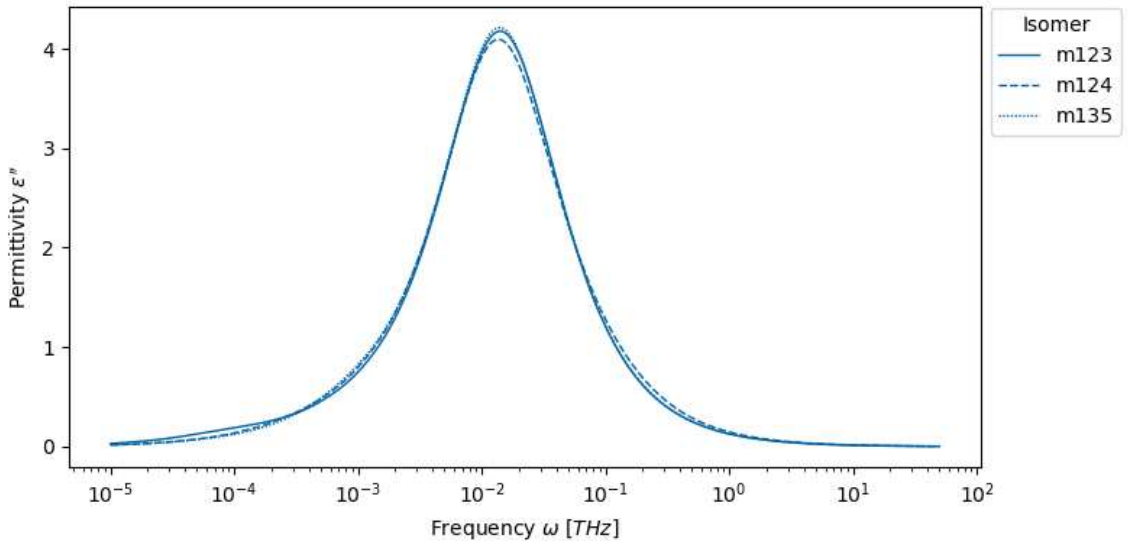
(a) Full dielectric spectra.



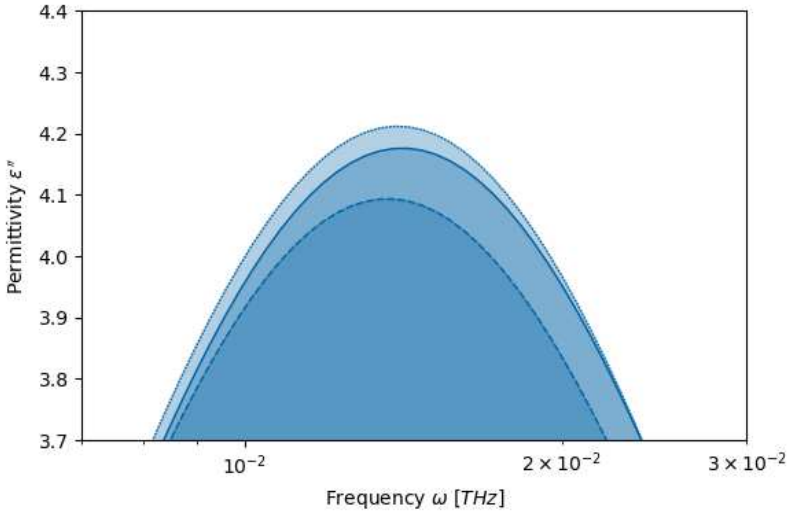
(b) Closer view of the peak maxima.

Figure 4.9.: Average imaginary part of the dielectric permittivity ϵ'' of all spiropyran functionalized NU-1000 simulations.

Similarly, the dielectric data for merocyanine-functionalized NU-1000 (open form) is plotted below, see Fig. 4.10. The peak maxima vary slightly more, though all peak positions remain at $1.4 \cdot 10^{-2}$ THz, with maximum values of 4.18 for m123, 4.09 for m124, and 4.21 for m135.



(a) Full dielectric spectra.



(b) Closer view of the peak maxima.

Figure 4.10.: Average imaginary part of the dielectric permittivity ϵ'' of all merocyanine functionalized NU-1000 simulations.

4.3.2. Comparison with Bulk Solvent

Comparing the total dielectric spectra for all simulated systems, as shown in Fig. 4.11, reveals that the unmodified MOF produces results very similar to the functionalized versions. These peaks are all located at approximately $1.4 \cdot 10^{-2}$ THz. The peak maxima of the overall replica-averaged spectra are 4.18 for unmodified NU-1000, 4.20 for the spiropyran-modified MOF, and 4.16 for the merocyanine version. However, the bulk DFB solvent spectrum is shifted to $1.5 \cdot 10^{-2}$ THz, with a maximum at 5.05. This indicates the expected behavior, that total translational and rotational motion is reduced by introducing the MOF structure into the system.

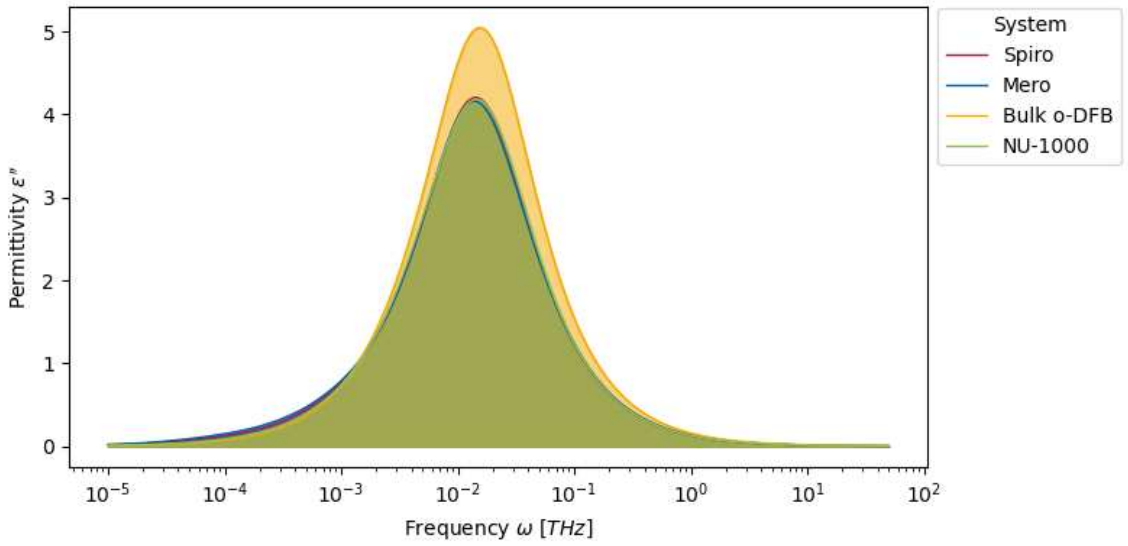


Figure 4.11.: Total average ϵ'' spectra of bulk DFB simulations (yellow), unmodified NU-1000 (green), spiropyran modified NU-1000 (red) and meroyanine in NU-1000 (blue).

4.3.3. Spiropyran and Merocyanine Contribution

To clarify the effect of the spiropyran group in its closed and open form, the contributions were plotted separately, as previously shown in the IR section. The average contributions of the spiropyran residues are depicted in Fig. 4.12 below. For spiropyran, the peaks were located at $7.0 \cdot 10^{-5}$ THz with a maximum of 0.0186 for s123, at $7.2 \cdot 10^{-5}$ THz with a maximum of 0.0057 for s124, and at $1.0 \cdot 10^{-4}$ THz with a peak height of 0.0057 for s135. The merocyanine residues exhibited peaks at $7.0 \cdot 10^{-5}$ THz with a maximum of 0.0180 for m123, at $1.1 \cdot 10^{-4}$ THz reaching 0.0078 for m124, and at $7.1 \cdot 10^{-5}$ THz with a maximum of 0.0135 for m135.

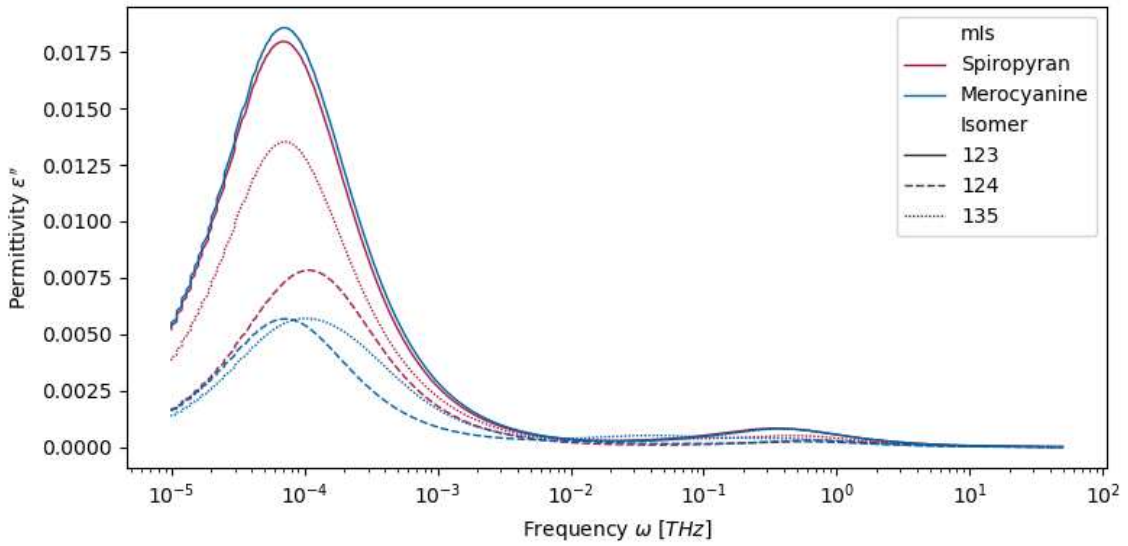


Figure 4.12.: Dielectric permittivity contributions of the closed (red) and open (blue) spiropyran residue within the NU-1000.

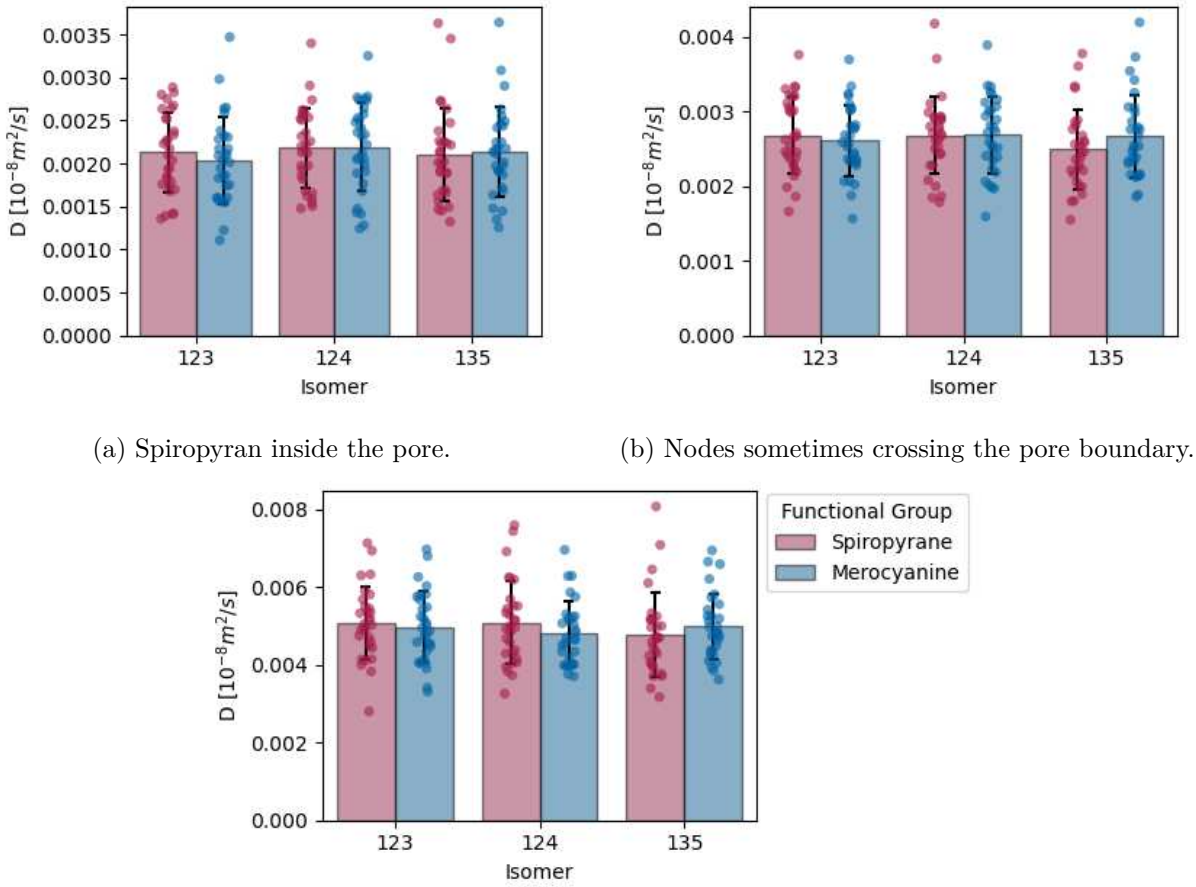
Though no clear correlation between the open and closed forms can be derived from these results, the coordination position appears to influence the dielectric contributions, with the average ϵ'' contribution being notably higher for isomers where the functional groups are all neighboring. However, the small sample size of three residues per simulation and the larger variability across individual replicas undermine such assumptions. Further investigation would be necessary to draw a reliable conclusion.

4.4. Diffusion Distribution within the Pores

The final results section presents the calculated diffusion coefficients, grouped by residue type, which were obtained as described in section 3.4.2 of the Methods. The first set includes all MOF residues, which are connected to the extended framework. Next, the solvent diffusion is compared to the established baseline. Finally, ion movement is analyzed and discussed in the context of these results.

4.4.1. MOF Structure Residues

The bar graphs shown in Fig. 4.13 display the diffusion coefficients of the spiropyran molecules and the zirconium nodes. Unlike the other residues, spiropyran remained confined within the pore space throughout the simulations, while the center of mass of the zirconium nodes consistently never stayed inside the pore region over each trajectory. Literature commonly estimates diffusion coefficients to be around $10^{-5}\mathrm{m}^2/\mathrm{s}$ for gases, approximately $10^{-9}\mathrm{m}^2/\mathrm{s}$ for liquids, and even lower for solids. However, for solids, the values can vary significantly depending on temperature. One example is hydrogen diffusion in iron, which is roughly estimated at $10^{-13}\mathrm{m}^2/\mathrm{s}$. Another example is carbon diffusion in austenite, face-centered cubic γ-iron, with diffusion coefficients around $10^{-35}\mathrm{m}^2/\mathrm{s}$ [64].



(a) Spiropyran inside the pore.

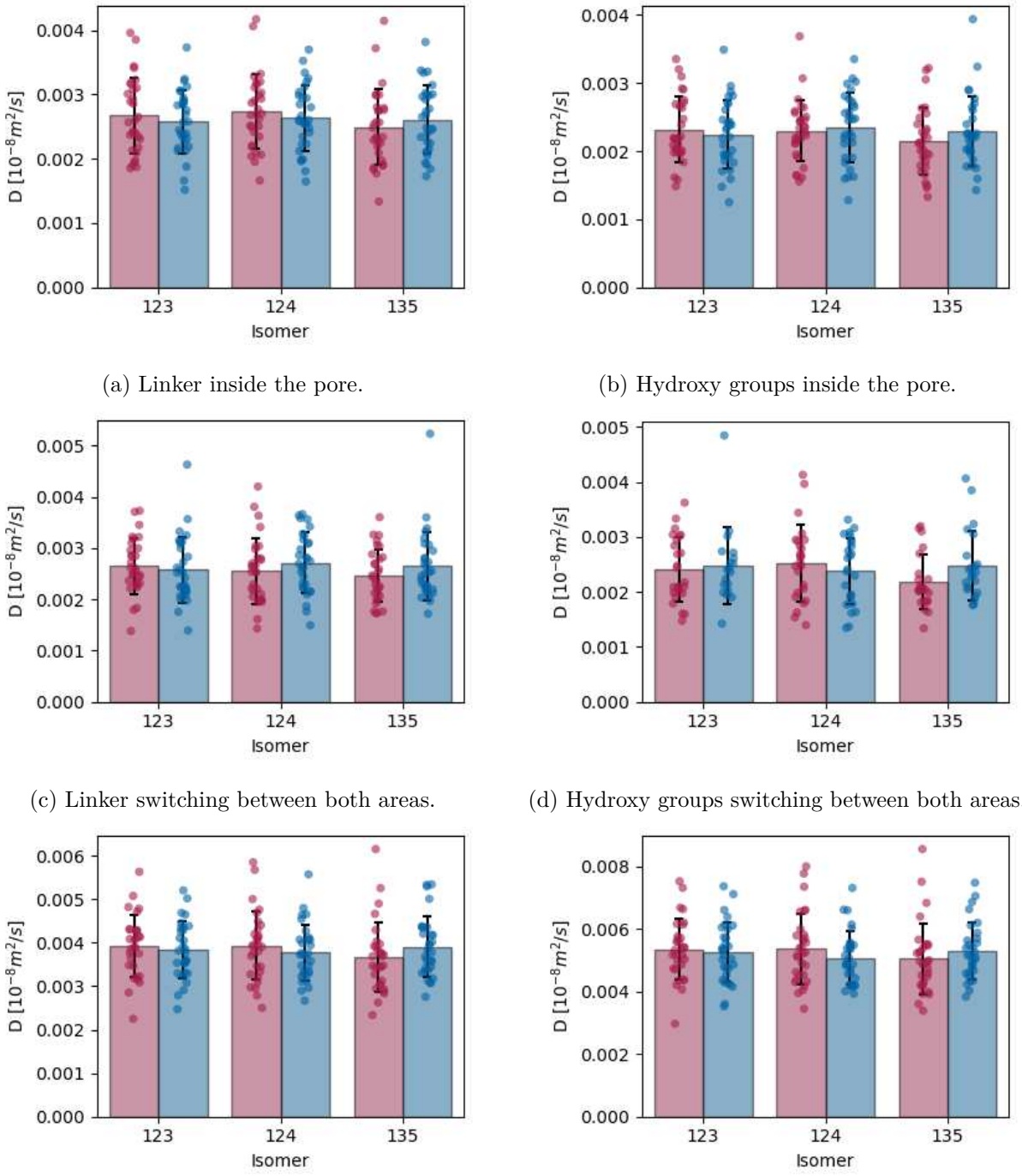
(b) Nodes sometimes crossing the pore boundary.

(c) Nodes outside the pore.

Figure 4.13.: Diffusion coefficients of nodes and spiropyran, each grouped by closed (red) and open (blue) form, as well as coordination position isomer.

The average diffusion coefficients of the spiropyran residue aggregated around $2 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$, which coincides with literature estimates for values between those of liquids and solids. Similar results were found for the nodes that crossed the pore boundary, though those were positioned closer to $3 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$. The nodes outside the pore yielded slightly higher values, around $5 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$, which is likely due to the increased mobility of the nodes at the outer edges of the MOF structure. The merocyanine form had no significant impact on diffusion coefficients, as expected.

A similar pattern was observed for the remaining MOF residues, namely the linkers and hydroxy groups that filled empty coordination bonds, as illustrated in Fig. 4.14 below.



(a) Linker inside the pore.

(b) Hydroxy groups inside the pore.

(c) Linker switching between both areas.

(d) Hydroxy groups switching between both areas.

(e) Linker outside the pore.

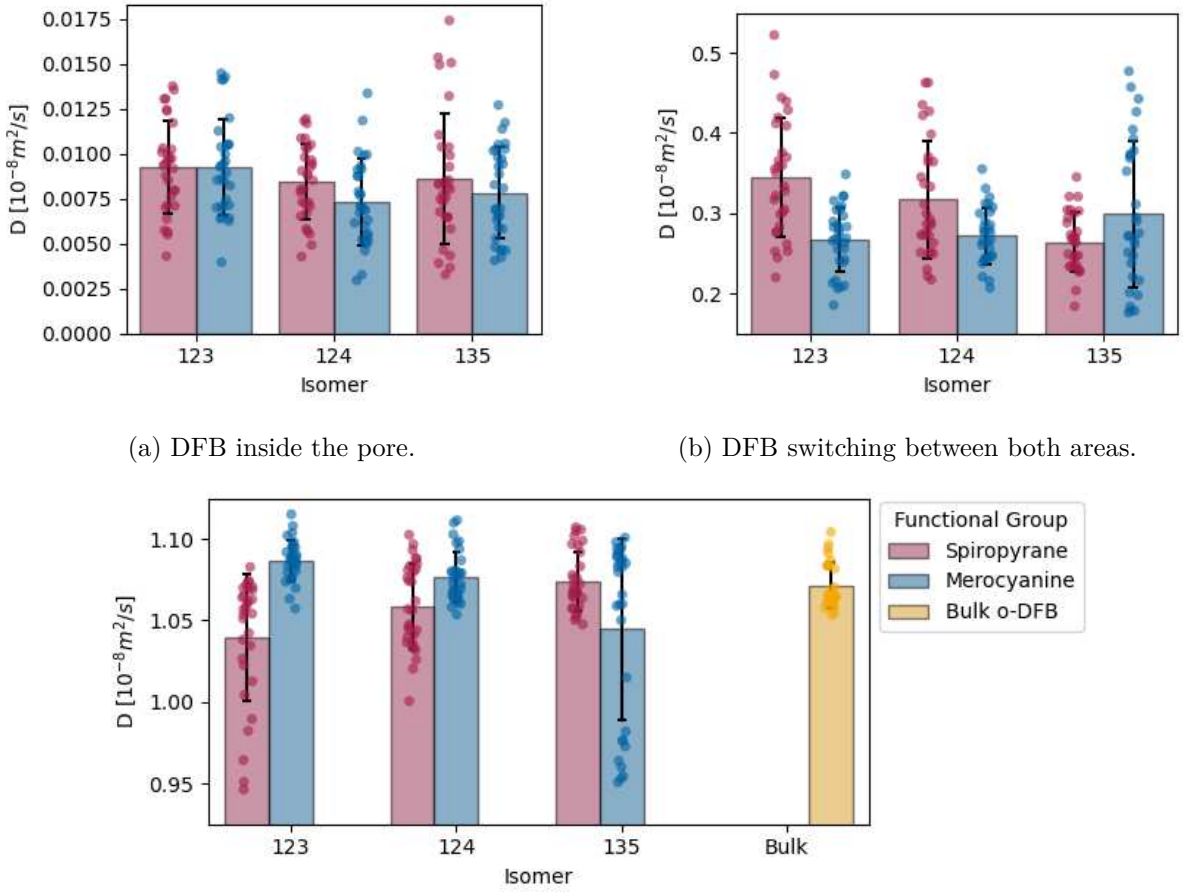
(f) Hydroxy groups outside the pore.

Figure 4.14.: Diffusion coefficients of linker and terminal hydroxy groups integrated in the MOF structure, each categorized by spiropyran (red) and merocyanine (blue) form, as well as coordination position isomer.

Akin to the nodes, average diffusion coefficients for linkers and hydroxy groups located inside or transitioning between the pore area fluctuated between 2 - 3 $\cdot 10^{-11}$m^2/s. Linkers outside the pore yielded coefficients close to $4 \cdot 10^{-11}$m^2/s and hydroxy groups outside the pore averaged approximately $5 \cdot 10^{-11}$m^2/s, most probably due to increased mobility around the outer edges of the structure.

4.4.2. Solvent DFB

The diffusivity of *o*-difluorobenzene is given in Fig. 4.15 below. In addition to the diffusion coefficients obtained from the functionalized NU-1000 simulations, the diffusion values for the bulk solvent simulations are also presented.



(a) DFB inside the pore.

(b) DFB switching between both areas.

(c) DFB outside the pore boundary.

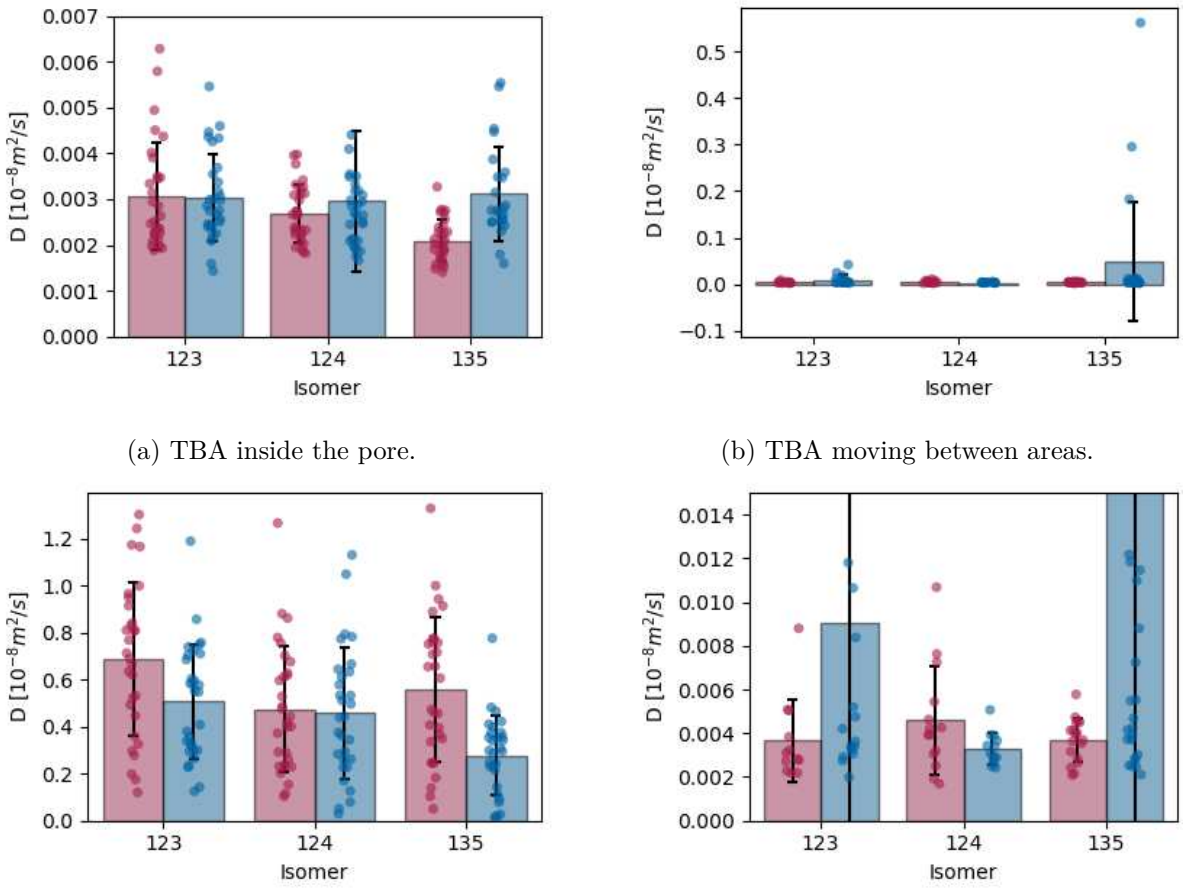
Figure 4.15.: Diffusion coefficients of DFB solvent residues, each grouped by spiropyran (red) and merocyanine (blue) functionalization, as well as coordination position isomer. Additionally, the bulk solvent data (yellow) was included for outside of the pore behavior (c).

The diffusion coefficients inside the pore fluctuated around $8 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$, which is three to four times faster than those of the MOF components themselves, but still remain within the same order of magnitude (10^{-11}), and about ten times slower than literature values for DFB would suggest. A reference value at 298 K of $8.5 \cdot 10^{-10}\mathrm{m}^2/\mathrm{s}$ was used for this assessment [65].
A stark increase can be seen with the mixed-positioned DFB residues, yielding diffusion coefficients around $3 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$, which is already marginally faster than expected literature values. The average diffusion outside the pore area was much higher, with values between 1.05 and $1.10 \cdot 10^{-8}\mathrm{m}^2/\mathrm{s}$. This means the coefficients are nearly thirteen times higher than the literature reference, placing these results between expected values for gases and liquids, but relatively close to liquids. One reason for this discrepancy might be insufficiently described non-bonded interactions, leading to underestimated densities.
Of greater relevance, the trajectory length for pore diffusion is limited, due to the fact that statistically all molecules leave or enter the pore at some point, and therefore it is not possible to use the full set of

trajectories in that case. However, since this limitation does not apply to the bulk simulation, diffusion coefficients were also calculated using the MSD over all 10 trajectories of one replica, corresponding to a total simulation time of 25 ns. These resulted in an average diffusion coefficient of $2.593 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$ (Replica 1,2,3: 2.539, 2.544, 2.695), which is approximately three times higher than the reference values, but much closer to $0.85 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$. This suggests that, due to the short 2.5 ns trajectories, the calculated diffusion coefficients are systematically overestimated, as the bulk simulation average of $10.72 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$ is still roughly four times higher, and thus faster, than the bulk simulation average obtained from the full 25 ns of all trajectories per replica. Nonetheless, the results calculated from short trajectories were used for the evaluation to maintain a consistent procedure throughout all systems, and make results comparable.

4.4.3. TBA and Br Ions

The final, and most interesting in the context of electromagnetic field effects, datasets are the diffusion coefficients of the ion pairs present in the investigated system. As one of the ions in solution, the mobility of tetrabutylammonium is a point of interest for the overall alignment effect, and the corresponding diffusion coefficients are depicted in Fig. 4.16 below.



(a) TBA inside the pore.

(b) TBA moving between areas.

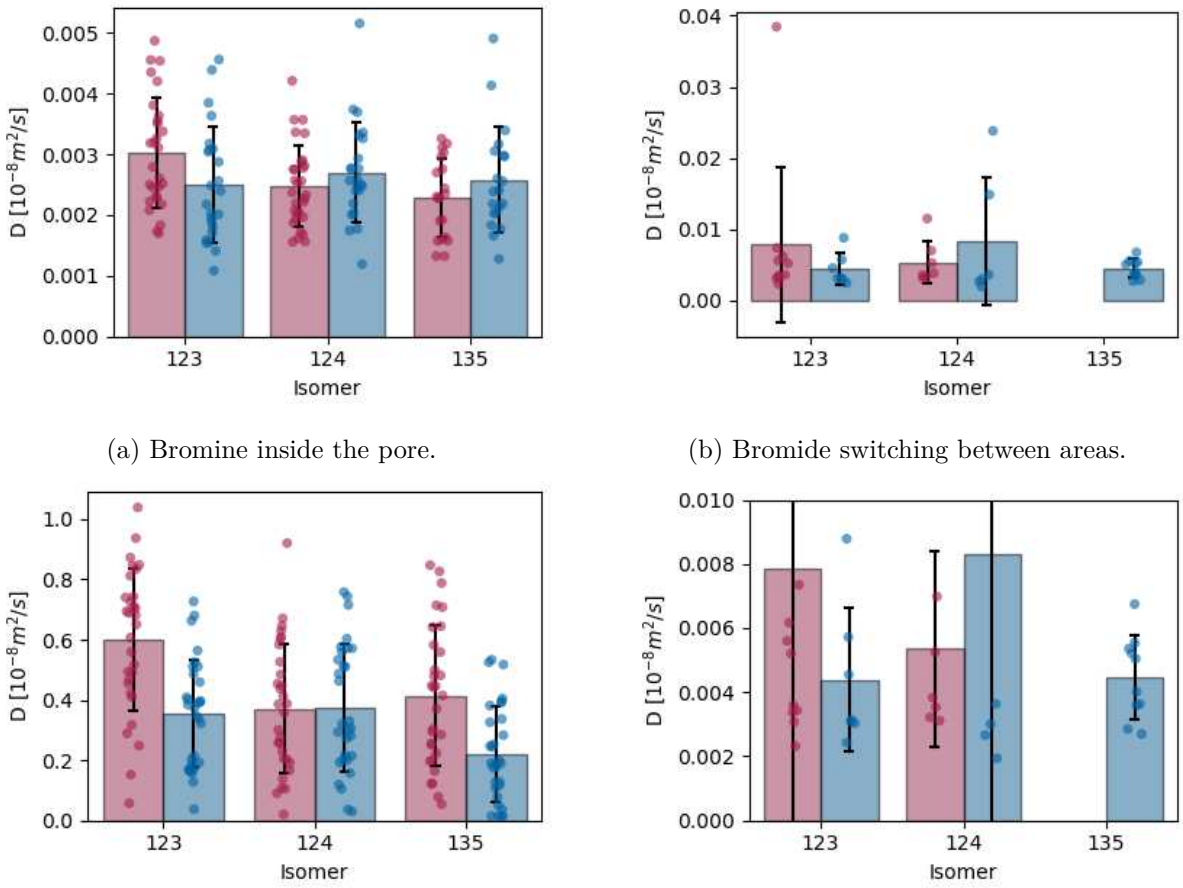
(c) TBA outside the pore.

(d) Closer view of TBA switching areas.

Figure 4.16.: Diffusion coefficients of TBA ions, each grouped by closed (red) and open (blue) spiropyran form, as well as coordination position isomer.

The average diffusion coefficients of TBA inside the pore generally fluctuated around $3 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$, thus moving surprisingly slowly and at approximately the same rate as the rest of the MOF structure.

It should be noted that for the m124 bar, one data point at $10.27 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$ was cropped out of the illustration for the inside of the pore plot (a). Similar to previously discussed cases, the TBA residues moving between the inside and outside of the pore produce higher coefficients. However, in this case, the dispersion of data points is larger for merocyanine simulations m123 and m135, which is most likely due to poor statistics, since only a few molecules are observed. Much more mobility can be seen in the dataset of TBA residues outside the pore, with diffusion coefficients around $5 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$, which is nearly fifty times higher than within the pore and about half the value of solvent coefficients.
In Fig. 4.17, the coefficient values for the bromine ions are illustrated, as with the previous residues.



(a) Bromine inside the pore.

(b) Bromide switching between areas.

(c) Bromide outside the pore.

(d) Closer view of bromide switching areas.

Figure 4.17.: Diffusion coefficients of bromine ions, each grouped by spiropyran (red) and merocyanine (blue) form, as well as coordination position isomer.

Even though bromine ions are smaller than TBA, the average diffusion coefficients inside the pore fluctuated between $2 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$ and $3 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$, just like TBA and all fixed structure components, thus reinforcing the conclusion that once inside the pore, the salt molecules are affected by the confined space in a way that slows movement to approximately the same level as the surrounding MOF structure. Analogous to TBA, the average bromine diffusion coefficients of those molecules that switch between the pore and the outside are slightly higher and show a larger distribution relative to the dataset inside the pore, with average values around $6 \cdot 10^{-11}\mathrm{m}^2/\mathrm{s}$. One remarkable exception is the spiropyran s135 configuration, where not a single bromine ion left or entered the pore during any trajectory.
Lastly, the average diffusion coefficients outside of the pore boundary fluctuate around $4 \cdot 10^{-9}\mathrm{m}^2/\mathrm{s}$, very similar to the TBA results. Given that bromine ions are relatively small compared to TBA or

the solvent molecules, but the reduction in mobility when comparing bromine ions inside and outside the pore is approximately the same as for TBA, these diffusion coefficients suggest that the reduced mobility is most likely not caused by steric effects. However, verifying the likely interaction between ions and merocyanine was not possible.
Furthermore, in none of the datasets for any residue was a clear trend concerning coordination position isomers established. Similarly, no clear difference or trend was visible when comparing spiropyran and merocyanine forms. This matter is discussed in more detail in the following sections 4.5 and 4.6.

4.5. Discussion

In summary, the overall geometry of the modeled pore structure was stable throughout the simulations, and the pore size was consistent with the literature, suggesting decent parametrization for attractive and repulsive forces within the large pore structure.
The infrared spectra were recreated, and several peaks matching with experimental data were identified. Nonetheless, calculated spectra still deviated significantly. However, it is worth pointing out that the force field was not optimized for recreating spectral properties, and having some indicative peaks for the spiropyran-to-merocyanine transition visible is a positive result. It is not unlikely that the general force field used by ACPYPE is not suitable for the perfect recreation of experimental spectra. On the other hand, massive optimization efforts to recreate specific properties with a force field can hypothetically negatively impact overall results of other calculated properties, which could be of more interest.
The dielectric spectra gave some insight into the overall molecular rotational effects within the system, clearly showing the decreasing effect of introducing the large MOF pore structure to the system. The contribution of the spiropyran and merocyanine groups was less conclusive, though. The data somewhat indicates a steric influence if all functional groups are neighboring, but no clear difference between the open and closed forms and no real trend could be identified with certainty. Of course, the small amount of spiropyran residues per simulation is not advantageous.
The calculated diffusion coefficients revealed that the pore structure moves at magnitudes slower than the bulk solvent, as expected. Additionally, the separate analysis of the behavior inside and outside the pore clearly showed that any solvent or ion is significantly slowed down. Most interestingly, the coefficients of ions inside the pore were basically the same as the rest of the MOF structure, suggesting relevant interactions between the pore environment and ions that go beyond the steric confinement effects that DFB is subjected to. In contrast, when comparing the values of diffusion coefficients to experimental data, the results were less satisfying. Overall, the coefficients were much higher than what real-world examples would suggest. This problem can be mainly attributed to the small trajectory size available for this method, see 4.6. However, even using the full trajectory of the bulk simulation yields higher diffusion coefficients than experimental references. This is most likely due to intermolecular interactions not being properly parameterized for this application, again indicating that the general force field approach of ACPYPE does not suffice to replicate experimental data.

4.6. Limitations & Reliability

The actual NU-1000 crystal structure is far larger in reality. The system size required to simulate a whole rod is not realistically feasible with these methods at this point. Additionally, periodicity of the lattice structure was also disregarded. Furthermore, the small number of functional groups inside the pore and the number of ions, though generally based on reality, were unfavorable for the statistics of a simulation approach. The limited pore structure was necessary to enable the observation of ions and solvent migrating into and out of the pore area for the purpose of this project. Nonetheless, it is of little surprise that a non-periodic imitation with a distorted MOF-to-outer-surface area ratio is suboptimal for an accurate recreation of experimental data.
The method for generating diffusion coefficients is also limited by this circumstance. Using longer trajectories would lead to all molecules that are not part of the pore structure eventually passing through both pore area and bulk area. Thus, generating the separation was only possible using shorter simulation segments in this case. Though another alternative would be to create a boundary between the pore and outside, it would also take away the possibility to observe ions entering or leaving and of visualizing preferred occupation areas. The trade-off was reduced statistical reliability on top of the small number of relevant residues being present. The impact was showcased in the bulk DFB example, where using all 10 trajectories, in other words, 25 ns of simulation time, led to far more realistic results. The difference is also clearly visible when plotting the MSD for both methods, see Fig. 4.18 below.

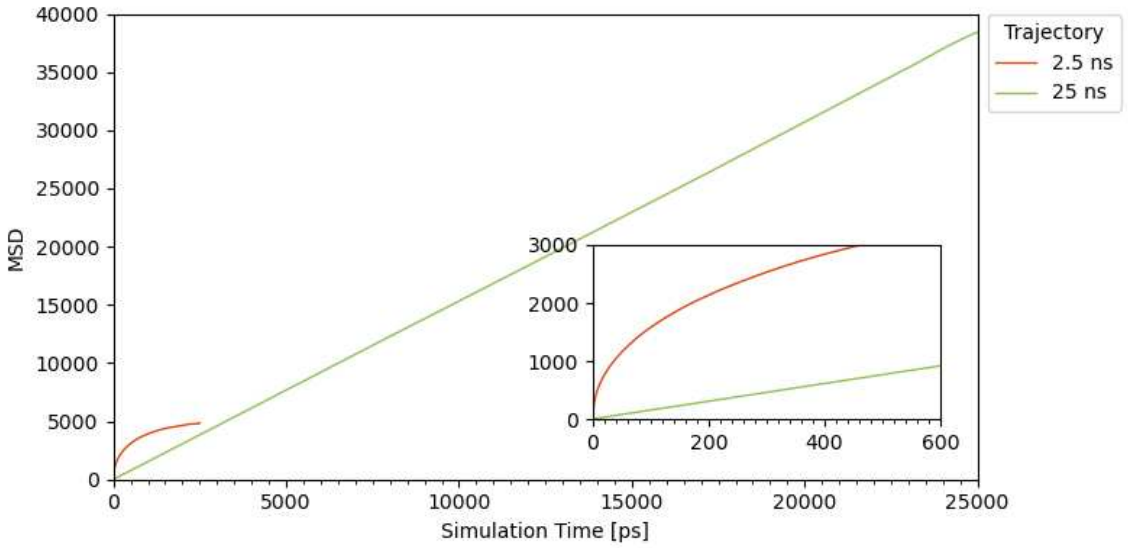


Figure 4.18.: MSD of bulk DFB calculated during trajectory analysis using one 2.5 ns trajectory (orange) or over 10 trajectories for a total length of 25 ns (green).

The MSD for the short trajectory is non-linear, and the initial slope is much steeper than the linear result for the long trajectory. Still, the diffusion results calculated from these non-linear MSDs should contain some informational value if viewed in relation to each other.
As a comparison with literature reference data showed, even with a proper sample size of ten times the trajectory length, the solvent still moved too quickly, suggesting that the generalized force field used to parameterize the DFB was at least somewhat insufficient to properly replicate the solvent behavior.
Even if the generalized force field quality is disregarded, a similar concern exists for the NU-1000 parameters. The source material available was parameters for UiO-66, which is coordinated with 12 ligands, fully filling all binding positions with linkers. Thus, the parametrization was originally

conducted for a structure that completely lacks the OH/H_2O groups bound to the zirconium nodes, as seen in NU-1000. Additionally, the terephthalic acid linkers in UiO-66 served as a base for the parameters of the NU-1000 linkers, under the assumption that the behavior would be similar. While it was the most practical approach, and the most effective given time constraints, it has to be noted that an individual parametrization utilizing experimental data, quantum mechanical calculations, and/or machine learning could lead to a more refined and better-performing force field.
Lastly, during force field refinement test simulations, there were initially problems with collapsing nodes, unless the literature Lennard-Jones potential value σ was doubled for node parameters. While this produced stable simulations, it should be mentioned that the authors most likely provided the Van der Waals radius r_{vdw}, which can be translated to an OpenMM σ as previously discussed in Eq. 2.6, leading to the following factor.

$$r_{vdw} \cdot 1.78179743628 = \sigma \tag{4.1}$$

The deviation was initially not identified because the other parameters provided by the authors for the linkers were in a very similar range as would be expected for σ. Additionally, the authors of the original publication did only refer to the parameters as LJ terms, not specifying, whether they referred to the Van der Waals radius or zero-energy distance, but actually described non-bonded interactions with the same equation as provided in this work (see Eq. 2.5) [49]. Though the estimation produced decent behavior, there is a deviation by a factor of approximately 0.9, which could be relevant in other studies. Here, the main point of investigation was focused on the interaction of spiropyran isomers, which should not be impacted by this change.

5 | Conclusion

All things considered, the project was a partial success. The force field SNUFF, adapted and refined from literature, was sufficient to create stable simulations in OpenMM, which reproduced the geometry and pore size of NU-1000 very accurately. Furthermore, several infrared and dielectric spectra were calculated from the simulation data. Although the calculated infrared spectra did not compare as well to experimental data as desired, the separate contributions of functional spiropyran or merocyanine groups provided some insight. Moreover, extensive testing of the created system variations in the context of diffusion revealed the large impact of confined space effects within the pore on molecule mobility. However, the diffusion coefficients calculated from MSD did not compare well to experimental data found in the literature. Nonetheless, given the limitations of such a pore analysis concerning trajectory length, the results obtained were still relevant for comparison with each other. The initial point of interest was the different behavior of spiropyran-functionalized and merocyanine-functionalized NU-1000. The conducted computational experiments were not sufficient to produce a significant difference between the two isomers. Additionally, verification of the created force field was not entirely conclusive either, since the general force field parameters used by ACPYPE seemed insufficient to replicate experimental data for the bulk solvent. In conclusion, the created force field and pore analysis method worked to an extent, but require refinement to reproduce experimental data. Just as important, the limitation of small sample size within a pore leading to less-than-optimal statistics needs to be overcome in future investigations. Still, this work serves as a solid foundation for future inquiries.

5.1. Future outlook

The question of proper parameterization remains, and there are several approaches, which could solve or, at least, improve the current state. Quantum mechanical calculations are time- and cost-intensive, though of course they are the first option to create a more refined force field. A more cost-efficient approach might be to turn to machine learning methods, such as the recently developed MACE software [66, 67]. Another option is to utilize the Open Force Field toolkit for small molecules, which is designed to avoid atom types by using a direct chemical perception method [68].
Finally, diffusion within the pores should be re-evaluated to improve statistical robustness. This could be achieved by dropping the small pore model and transitioning to a periodic model in which ions and solvent exist only within the periodic pore structure, thus allowing for longer trajectories to be used in the calculation. Bulk simulations containing ions could serve as a baseline in this case. Similarly, the number of spiropyran groups inside the pore might also be increased in further investigations. The positions of these groups within the pores are currently not experimentally verified, and the ratio used in this work assumed an even distribution throughout the structure. It is entirely possible that during the solvent-assisted ligand incorporation (SALI), the spiropyran groups primarily coordinated near pore ends. A higher number of functional groups might reveal more about their effect on ion mobility and perhaps resolve the question of the alignment of functionalized NU-1000 in an electromagnetic field.

Bibliography

[1] S. G. Arturo, L. J. Broadbelt, P. J. Dauenhauer, and A. Govind Rajan, ACS Engineering Au **4**, 293 (2024), https://doi.org/10.1021/acsengineeringau.4c00014, URL https://doi.org/10.1021/acsengineeringau.4c00014.

[2] L. Zhang, H. Mao, Q. Liu, and R. Gani, Current Opinion in Chemical Engineering **27**, 22 (2020), ISSN 2211-3398, frontiers of Chemical Engineering: Chemical Product Design, URL https://www.sciencedirect.com/science/article/pii/S2211339819300504.

[3] N. Villavicencio and M. N. Groves, ACS Engineering Au **4**, 381 (2024), https://doi.org/10.1021/acsengineeringau.3c00068, URL https://doi.org/10.1021/acsengineeringau.3c00068.

[4] M. J. Buehler, ACS Engineering Au **4**, 241 (2024), https://doi.org/10.1021/acsengineeringau.3c00058, URL https://doi.org/10.1021/acsengineeringau.3c00058.

[5] H. Chen and J. Bajorath, Scientific reports **13**, 7412 (2023), ISSN 2045-2322, URL https://europepmc.org/articles/PMC10164739.

[6] D. Dhiman, M. Alhammadi, H. Kim, R. Umapathi, Y. S. Huh, and P. Venkatesu, Advanced Therapeutics **7**, 2400090 (2024), https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/pdf/10.1002/adtp.202400090, URL https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/abs/10.1002/adtp.202400090.

[7] H. Zhou, Y. Fang, and H. Gao, ACS Engineering Au **4**, 231 (2024), https://doi.org/10.1021/acsengineeringau.3c00056, URL https://doi.org/10.1021/acsengineeringau.3c00056.

[8] H. Furukawa, K. E. Cordova, M. O'Keeffe, and O. M. Yaghi, Science **341**, 1230444 (2013), https://www.science.org/doi/pdf/10.1126/science.1230444, URL https://www.science.org/doi/abs/10.1126/science.1230444.

[9] X. Yin and C. E. Gounaris, Computers & Chemical Engineering **167**, 108022 (2022), ISSN 0098-1354, URL https://www.sciencedirect.com/science/article/pii/S0098135422003568.

[10] J. E. Mondloch, W. Bury, D. Fairen-Jimenez, S. Kwon, E. J. DeMarco, M. H. Weston, A. A. Sarjeant, S. T. Nguyen, P. C. Stair, R. Q. Snurr, et al., Journal of the American Chemical Society **135**, 10294 (2013), pMID: 23829224, https://doi.org/10.1021/ja4050828, URL https://doi.org/10.1021/ja4050828.

[11] J. E. Mondloch, M. J. Katz, W. C. Isley III, P. Ghosh, P. Liao, W. Bury, G. W. Wagner, M. G. Hall, J. B. DeCoste, G. W. Peterson, et al., **14**, 512 (2015), ISSN 1476-4660, URL https://doi.org/10.1038/nmat4238.

[12] K. Healey, W. Liang, P. D. Southon, T. L. Church, and D. M. D'Alessandro, J. Mater. Chem. A **4**, 10816 (2016), URL http://dx.doi.org/10.1039/C6TA04160D.

[13] B. S. Lukyanov and M. B. Lukyanova, Chemistry of Heterocyclic Compounds **41**, 281 (2005), ISSN 1573-8353, URL https://doi.org/10.1007/s10593-005-0148-x.

[14] P. Eastman, R. Galvelis, R. P. Peláez, C. R. A. Abreu, S. E. Farr, E. Gallicchio, A. Gorenko, M. M. Henry, F. Hu, J. Huang, et al., The Journal of Physical Chemistry B **128**, 109 (2024), pMID: 38154096, https://doi.org/10.1021/acs.jpcb.3c06662, URL https://doi.org/10.1021/acs.jpcb.3c06662.

[15] O. K. Farha and J. T. Hupp, Accounts of Chemical Research **43**, 1166 (2010), pMID: 20608672, https://doi.org/10.1021/ar1000617, URL https://doi.org/10.1021/ar1000617.

[16] F. P. Kinik, A. Ortega-Guerrero, D. Ongari, C. P. Ireland, and B. Smit, Chem. Soc. Rev. **50**, 3143 (2021), URL http://dx.doi.org/10.1039/D0CS00424C.

[17] M. Mekata, Physics Today **56**, 12 (2003), ISSN 0031-9228, https://pubs.aip.org/physicstoday/article-pdf/56/2/12/11152056/12_1_online.pdf, URL https://doi.org/10.1063/1.1564329.

[18] W. Humphrey, A. Dalke, and K. Schulten, Journal of Molecular Graphics **14**, 33 (1996).

[19] T. C. Wang, N. A. Vermeulen, I. S. Kim, A. B. F. Martinson, J. F. Stoddart, J. T. Hupp, and O. K. Farha, **11**, 149 (2016), ISSN 1750-2799, URL https://www.nature.com/articles/nprot.2016.001.

[20] H. Beyzavi, R. C. Klet, S. Tussupbayev, J. Borycz, N. A. Vermeulen, C. J. Cramer, J. F. Stoddart, J. T. Hupp, and O. K. Farha, Journal of the American Chemical Society **136**, 15861 (2014), pMID: 25357020, https://doi.org/10.1021/ja508626n, URL https://doi.org/10.1021/ja508626n.

[21] M. Mandal, D. Banik, A. Karak, S. K. Manna, and A. K. Mahapatra, ACS Omega **7**, 36988 (2022), https://doi.org/10.1021/acsomega.2c04969, URL https://doi.org/10.1021/acsomega.2c04969.

[22] F. Jensen, *Introduction to computational chemistry* (Wiley, Chichester, 2017), third edition ed., ISBN 9781118825990.

[23] D. A. McQuarrie, *Quantum chemistry* (University Science Books, Sausalito, Calif., 2008), 2nd ed., ISBN 9781891389504.

[24] G. J. Martyna, M. L. Klein, and M. Tuckerman, The Journal of Chemical Physics **97**, 2635 (1992), ISSN 0021-9606, https://pubs.aip.org/aip/jcp/article-pdf/97/4/2635/19001249/2635_1_online.pdf, URL https://doi.org/10.1063/1.463940.

[25] K.-H. Chow and D. M. Ferguson, Computer Physics Communications **91**, 283 (1995), ISSN 0010-4655, URL https://www.sciencedirect.com/science/article/pii/0010465595000590.

[26] J. Åqvist, P. Wennerström, M. Nervall, S. Bjelic, and B. O. Brandsdal, Chemical Physics Letters **384**, 288 (2004), ISSN 0009-2614, URL https://www.sciencedirect.com/science/article/pii/S0009261403021687.

[27] J. M. Hollas, *Modern spectroscopy* (Wiley & Sons, Chichester, 2007), 4th ed., ISBN 9780470844168.

[28] T. L. C. Jansen, The Journal of Chemical Physics **155**, 170901 (2021), ISSN 0021-9606, https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0064092/20029749/170901_1_5.0064092.pdf, URL https://doi.org/10.1063/5.0064092.

[29] D. C. Harris and M. D. Bertolucci, *Symmetry and spectroscopy : an introduction to vibrational and electronic spectroscopy*, Dover classics of science and mathematics (Dover Publ., New York, NY, 1989), unabridged, corr. republ., 1. publ. ed., ISBN 9780486661445.

[30] J.-P. Hansen and I. R. McDonald, in *Theory of Simple Liquids (Third Edition)*, edited by J.-P. Hansen and I. R. McDonald (Academic Press, Burlington, 2006), pp. 178–218, third edition ed., ISBN 978-0-12-370535-8, URL https://www.sciencedirect.com/science/article/pii/B9780123705358500094.

[31] M. Thomas, M. Brehm, R. Fligg, P. Vöhringer, and B. Kirchner, **15**, 6608 (2013), ISSN 1463-9084, URL https://pubs.rsc.org/en/content/articlelanding/2013/cp/c3cp44302g.

[32] J. Hudecová, K. H. Hopmann, and P. Bouř, The Journal of Physical Chemistry B **116**, 336 (2012), pMID: 22132857, https://doi.org/10.1021/jp208785a, URL https://doi.org/10.1021/jp208785a.

[33] C. Schröder and O. Steinhauser, **132**, 244109 (2010), ISSN 0021-9606, URL https://doi.org/10.1063/1.3432620.

[34] E. Helfand, Phys. Rev. **119**, 1 (1960), URL https://link.aps.org/doi/10.1103/PhysRev.119.1.

[35] E. J. Maginn, R. A. Messerly, D. J. Carlson, D. R. Roe, and J. R. Elliot, Living Journal of Computational Molecular Science **1**, 6324 (2018), URL https://livecomsjournal.org/index.php/livecoms/article/view/v1i1e6324.

[36] M. D. Hanwell, D. E. Curtis, D. C. Lonie, T. Vandermeersch, E. Zurek, and G. R. Hutchison, Journal of cheminformatics **4**, 17 (2012), ISSN 1758-2946, URL https://doi.org/10.1186/1758-2946-4-17.

[37] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, Journal of Cheminformatics **3**, 33 (2011), ISSN 1758-2946, URL https://doi.org/10.1186/1758-2946-3-33.

[38] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez, Journal of Computational Chemistry **30**, 2157 (2009), https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21224, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21224.

[39] J. M. Martínez and L. Martínez, Journal of Computational Chemistry **24**, 819 (2003), https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.10216, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.10216.

[40] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, **114**, 10024 (1992), ISSN 0002-7863, 1520-5126, URL https://pubs.acs.org/doi/abs/10.1021/ja00051a040.

[41] M. A. Addicoat, N. Vankova, I. F. Akter, and T. Heine, **10**, 880 (2014), ISSN 1549-9618, URL https://doi.org/10.1021/ct400952t.

[42] S. L. Mayo, B. D. Olafson, and W. A. Goddard, **94**, 8897 (1990), ISSN 0022-3654, 1541-5740, URL https://pubs.acs.org/doi/abs/10.1021/j100389a010.

[43] J. K. Bristow, D. Tiana, and A. Walsh, **10**, 4644 (2014), ISSN 1549-9618, URL https://doi.org/10.1021/ct500515h.

[44] D. Dubbeldam, K. S. Walton, D. E. Ellis, and R. Q. Snurr, Angewandte Chemie International Edition **46**, 4496 (2007), https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/pdf/10.1002/anie.200700218, URL https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/abs/10.1002/anie.200700218.

[45] P. Dauber-Osguthorpe, V. A. Roberts, D. J. Osguthorpe, J. Wolff, M. Genest, and A. T. Hagler, Proteins: Structure, Function, and Bioinformatics **4**, 31 (1988), https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/pdf/10.1002/prot.340040106, URL https://onlinelibrary-wiley-com.uaccess.univie.ac.at/doi/abs/10.1002/prot.340040106.

[46] M. G. Martin and J. I. Siepmann, The Journal of Physical Chemistry B **102**, 2569 (1998), https://doi.org/10.1021/jp972543+, URL https://doi.org/10.1021/jp972543+.

[47] B. L. Eggimann, A. J. Sunnarborg, H. D. Stern, A. P. Bliss, and J. I. S. and, Molecular Simulation **40**, 101 (2014), https://doi.org/10.1080/08927022.2013.842994, URL https://doi.org/10.1080/08927022.2013.842994.

[48] G. Ercakir, G. O. Aksu, C. Altintas, and S. Keskin, ACS Engineering Au **3**, 488 (2023), https://doi.org/10.1021/acsengineeringau.3c00039, URL https://doi.org/10.1021/acsengineeringau.3c00039.

[49] Q. Yang, A. D. Wiersum, H. Jobic, V. Guillerm, C. Serre, P. L. Llewellyn, and G. Maurin, **115**, 13768 (2011), ISSN 1932-7447, URL https://doi.org/10.1021/jp202633t.

[50] M. R. Shirts, C. Klein, J. M. Swails, J. Yin, M. K. Gilson, D. L. Mobley, D. A. Case, and E. D. Zhong, Journal of Computer-Aided Molecular Design **31**, 147 (2017), ISSN 1573-4951, URL https://doi.org/10.1007/s10822-016-9977-1.

[51] A. W. Sousa da Silva and W. F. Vranken, BMC Research Notes **5**, 367 (2012), ISSN 1756-0500, URL https://doi.org/10.1186/1756-0500-5-367.

[52] L. Kagami, A. Wilter, A. Diaz, and W. Vranken, Bioinformatics **39**, btad350 (2023), ISSN 1367-4811, https://academic.oup.com/bioinformatics/article-pdf/39/6/btad350/50599896/btad350.pdf, URL https://doi.org/10.1093/bioinformatics/btad350.

[53] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein, The Journal of Chemical Physics **79**, 926 (1983), ISSN 0021-9606, https://pubs.aip.org/aip/jcp/article-pdf/79/2/926/18942849/926_1_online.pdf, URL https://doi.org/10.1063/1.445869.

[54] I. S. Joung and T. E. I. Cheatham, The Journal of Physical Chemistry B **112**, 9020 (2008), pMID: 18593145, https://doi.org/10.1021/jp8001614, URL https://doi.org/10.1021/jp8001614.

[55] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma, The Journal of Physical Chemistry **91**, 6269 (1987), https://doi.org/10.1021/j100308a038, URL https://doi.org/10.1021/j100308a038.

[56] N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein, Journal of Computational Chemistry **32**, 2319 (2011), https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21787, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21787.

[57] Richard J. Gowers, Max Linke, Jonathan Barnoud, Tyler J. E. Reddy, Manuel N. Melo, Sean L. Seyler, Jan Domański, David L. Dotson, Sébastien Buchoux, Ian M. Kenney, et al., in *Proceedings of the 15th Python in Science Conference*, edited by Sebastian Benthall and Scott Rostrup (2016), pp. 98 – 105.

[58] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al., Nature **585**, 357 (2020), URL https://doi.org/10.1038/s41586-020-2649-2.

[59] T. E. Oliphant, Computing in Science & Engineering **9**, 10 (2007).

[60] P. de Buyl, Journal of Open Source Software **3**, 877 (2018).

[61] J. D. Hunter, Computing in Science & Engineering **9**, 90 (2007).

[62] M. L. Waskom, Journal of Open Source Software **6**, 3021 (2021), URL https://doi.org/10.21105/joss.03021.

[63] M. A. Jette and T. Wickberg, in *Job Scheduling Strategies for Parallel Processing: 26th Workshop, JSSPP 2023, St. Petersburg, FL, USA, May 19, 2023, Revised Selected Papers*, edited by D. Klusáček, J. Corbalán, and G. P. Rodrigo (Springer Nature Switzerland, Cham, 2023), p. 3–23, ISBN 978-3-031-43943-8, URL https://doi-org.uaccess.univie.ac.at/10.1007/978-3-031-43943-8_1.

[64] E. L. Cussler, *Values of Diffusion Coefficients* (Cambridge University Press, 2009), p. 117–160, Cambridge Series in Chemical Engineering.

[65] D. J. Codling, G. Zheng, T. Stait-Gardner, and W. S. Price, Journal of Molecular Liquids **198**, 392 (2014), ISSN 0167-7322, URL https://www.sciencedirect.com/science/article/pii/S0167732214003481.

[66] I. Batatia, D. P. Kovacs, G. N. C. Simm, C. Ortner, and G. Csanyi, in *Advances in Neural Information Processing Systems*, edited by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho (2022), URL https://openreview.net/forum?id=YPpSngE-ZU.

[67] I. Batatia, S. Batzner, D. P. Kovács, A. Musaelian, G. N. C. Simm, R. Drautz, C. Ortner, B. Kozinsky, and G. Csányi, *The design space of e(3)-equivariant atom-centered interatomic potentials* (2022), 2205.06643.

[68] D. L. Mobley, C. C. Bannan, A. Rizzi, C. I. Bayly, J. D. Chodera, V. T. Lim, N. M. Lim, K. A. Beauchamp, D. R. Slochower, M. R. Shirts, et al., Journal of Chemical Theory and Computation **14**, 6076 (2018), pMID: 30351006, https://doi.org/10.1021/acs.jctc.8b00640, URL https://doi.org/10.1021/acs.jctc.8b00640.

A | Appendix - Force Field: SNUFF

A.1. Atom Types

For a better overview of the adapted forces, the atom types were labeled in the following representation of a single node connected to a linker, see Fig. A.1.

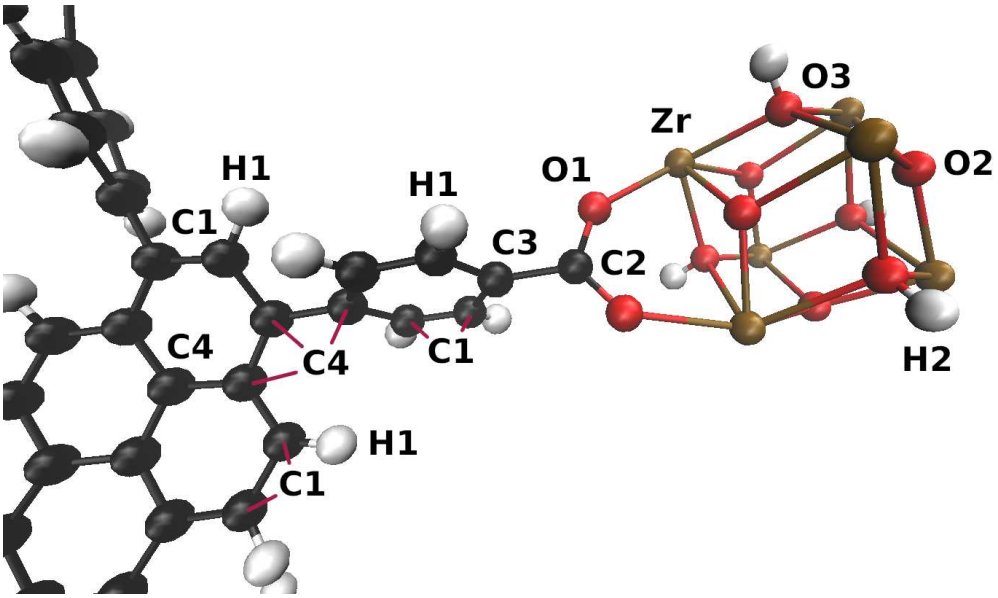


Figure A.1.: Atom types of the node and linker structures.

The OH groups used to occupy empty coordination positions were assigned with the atom types O4 and H2.

A.2. Force Field XML File

```
<ForceField>
  <Info>
   <Source>The 'S'piropyran-functionalized 'NU'-1000 'F'orce 'F'ield SNUFF was
    created as part of the master thesis 'Molecular dynamics simulaion of
    functionalized NU-1000 MOF' </Source>
   <Reference> J. Phys. Chem. C 2011, 115, 28, 13768-13774, https://doi.org
    /10.1021/jp202633t </Reference>
  </Info>
 <AtomTypes>
  <Type name="O1" class="Ligand" element="O" mass="15.999"/>
  <Type name="O2" class="NodeO" element="O" mass="15.999"/>
  <Type name="O3" class="NodeO" element="O" mass="15.999"/>
  <Type name="O4" class="Ligand" element="O" mass="15.999"/>
  <Type name="H1" class="2" element="H" mass="1.008"/>
  <Type name="H2" class="2" element="H" mass="1.008"/>
  <Type name="C1" class="3" element="C" mass="12.011"/>
  <Type name="C2" class="3" element="C" mass="12.011"/>
  <Type name="C3" class="3" element="C" mass="12.011"/>
  <Type name="C4" class="3" element="C" mass="12.011"/>
  <Type name="Zr" class="Zr" element="Zr" mass="91.224"/>
  <!-- water -->
  <Type name="O" class="waterO" element="O" mass="15.999"/>
  <Type name="H" class="waterH" element="H" mass="1.008"/>
  <!-- solvent o-DFB -->
  <Type name="SF" class="solventF" element="F" mass="18.998"/>
```

```
23   <Type name="SC" class="solventC" element="C"  mass="12.011"/>
24   <Type name="SH" class="solventH" element="H"  mass="1.008"/>
25   <!-- Spiropyran P (closed ring) -->
26   <Type name="PN1" class="PN1" element="N" mass="14.007"/>
27   <Type name="PO1" class="PO1" element="O" mass="15.999"/>
28   <Type name="PC1" class="PC1" element="C" mass="12.011"/>
29   <Type name="PC2" class="PC2" element="C" mass="12.011"/>
30   <Type name="PC3" class="PC3" element="C" mass="12.011"/>
31   <Type name="PC4" class="PC4" element="C" mass="12.011"/>
32   <Type name="PO2" class="PO2" element="O" mass="15.999"/>
33   <Type name="PN2" class="PN2" element="N" mass="14.007"/>
34   <Type name="PC5" class="PC5" element="C" mass="12.011"/>
35   <Type name="PH1" class="PH1" element="H" mass="1.008"/>
36   <Type name="PH2" class="PH2" element="H" mass="1.008"/>
37   <Type name="PH3" class="PH3" element="H" mass="1.008"/>
38   <!-- Merocyanine M (spiro with open ring) -->
39   <Type name="PC6" class="PC6" element="C" mass="12.011"/>
40   <Type name="PC7" class="PC7" element="C" mass="12.011"/>
41   <!-- Spiropyran H -->
42   <Type name="POH" class="POH" element="O" mass="15.999"/>
43   <Type name="PHO" class="PHO" element="H" mass="1.008"/>
44   <!-- TBA Br -->
45   <Type name="BN1" class="BN1" element="N" mass="14.007"/>
46   <Type name="BC1" class="BC3" element="C" mass="12.011"/>
47   <Type name="BH1" class="BHC" element="H" mass="1.008"/>
48   <Type name="BH2" class="BHX" element="H" mass="1.008"/>
49   <Type name="tip3p_standard-Br-" class="tip3p_standard-Br-" element="Br" mass="
       79.9"/>
50  </AtomTypes>
51
52  <Residues>
53   <Residue name="HOH">
54    <Atom name="WO1" type="O"/>
55    <Atom name="WH1" type="H"/>
56    <Atom name="WH2" type="H"/>
57    <Bond atomName1="WO1" atomName2="WH1"/>
58    <Bond atomName1="WO1" atomName2="WH2"/>
59   </Residue>
60   <Residue name="TOH">
61    <Atom name="TH1" type="H2" charge="0.4095 "/>
62    <Atom name="TO1" type="O4" charge="-0.6455"/>
63    <Bond atomName1="TH1" atomName2="TO1"/>
64    <ExternalBond atomName="TO1"/>
65   </Residue>
66   <Residue name="TER">
67    <Atom name="TH" type="H2" charge="0.472"/>
68    <ExternalBond atomName="TH"/>
69   </Residue>
70   <Residue name="BRO">
71     <Atom name="BR" type="tip3p_standard-Br-" charge="-1.0"/>
72   </Residue>
73   <Residue name="TBA">
74    <Atom name="B1"  type="BN1" charge="-0.604788"/>
75    <Atom name="B2"  type="BC1" charge="0.111819 "/>
76    <Atom name="B3"  type="BC1" charge="0.111819 "/>
77    <Atom name="B4"  type="BC1" charge="0.111819 "/>
78    <Atom name="B5"  type="BC1" charge="0.111819 "/>
79    <Atom name="B6"  type="BC1" charge="-0.101131"/>
80    <Atom name="B7"  type="BH1" charge="0.047219 "/>
81    <Atom name="B8"  type="BH1" charge="0.047219 "/>
82    <Atom name="B9"  type="BH1" charge="0.047219 "/>
83    <Atom name="B10" type="BH2" charge="0.094969 "/>
84    <Atom name="B11" type="BH2" charge="0.094969 "/>
85    <Atom name="B12" type="BC1" charge="-0.101131"/>
86    <Atom name="B13" type="BH1" charge="0.047219 "/>
87    <Atom name="B14" type="BH1" charge="0.047219 "/>
```

```
88    <Atom name="B15" type="BH1" charge="0.047219 "/>
89    <Atom name="B16" type="BH2" charge="0.094969 "/>
90    <Atom name="B17" type="BH2" charge="0.094969 "/>
91    <Atom name="B18" type="BC1" charge="-0.101131"/>
92    <Atom name="B19" type="BH1" charge="0.047219 "/>
93    <Atom name="B20" type="BH1" charge="0.047219 "/>
94    <Atom name="B21" type="BH1" charge="0.047219 "/>
95    <Atom name="B22" type="BH2" charge="0.094969 "/>
96    <Atom name="B23" type="BH2" charge="0.094969 "/>
97    <Atom name="B24" type="BC1" charge="-0.101131"/>
98    <Atom name="B25" type="BH1" charge="0.047219 "/>
99    <Atom name="B26" type="BH1" charge="0.047219 "/>
100   <Atom name="B27" type="BH1" charge="0.047219 "/>
101   <Atom name="B28" type="BH2" charge="0.094969 "/>
102   <Atom name="B29" type="BH2" charge="0.094969 "/>
103   <Atom name="B30" type="BC1" charge="-0.081631"/>
104   <Atom name="B31" type="BH1" charge="0.063719 "/>
105   <Atom name="B32" type="BH1" charge="0.063719 "/>
106   <Atom name="B33" type="BC1" charge="-0.081631"/>
107   <Atom name="B34" type="BH1" charge="0.063719 "/>
108   <Atom name="B35" type="BH1" charge="0.063719 "/>
109   <Atom name="B36" type="BC1" charge="-0.081631"/>
110   <Atom name="B37" type="BH1" charge="0.063719 "/>
111   <Atom name="B38" type="BH1" charge="0.063719 "/>
112   <Atom name="B39" type="BC1" charge="-0.081631"/>
113   <Atom name="B40" type="BH1" charge="0.063719 "/>
114   <Atom name="B41" type="BH1" charge="0.063719 "/>
115   <Atom name="B42" type="BC1" charge="-0.101331"/>
116   <Atom name="B43" type="BH1" charge="0.057219 "/>
117   <Atom name="B44" type="BH1" charge="0.057219 "/>
118   <Atom name="B45" type="BC1" charge="-0.101331"/>
119   <Atom name="B46" type="BH1" charge="0.057219 "/>
120   <Atom name="B47" type="BH1" charge="0.057219 "/>
121   <Atom name="B48" type="BC1" charge="-0.101331"/>
122   <Atom name="B49" type="BH1" charge="0.057219 "/>
123   <Atom name="B50" type="BH1" charge="0.057219 "/>
124   <Atom name="B51" type="BC1" charge="-0.101331"/>
125   <Atom name="B52" type="BH1" charge="0.057219 "/>
126   <Atom name="B53" type="BH1" charge="0.057219 "/>
127   <Bond atomName1="B1" atomName2="B2"/>
128   <Bond atomName1="B1" atomName2="B3"/>
129   <Bond atomName1="B1" atomName2="B4"/>
130   <Bond atomName1="B1" atomName2="B5"/>
131   <Bond atomName1="B2" atomName2="B6"/>
132   <Bond atomName1="B2" atomName2="B10"/>
133   <Bond atomName1="B2" atomName2="B11"/>
134   <Bond atomName1="B3" atomName2="B24"/>
135   <Bond atomName1="B3" atomName2="B28"/>
136   <Bond atomName1="B3" atomName2="B29"/>
137   <Bond atomName1="B4" atomName2="B12"/>
138   <Bond atomName1="B4" atomName2="B16"/>
139   <Bond atomName1="B4" atomName2="B17"/>
140   <Bond atomName1="B5" atomName2="B18"/>
141   <Bond atomName1="B5" atomName2="B22"/>
142   <Bond atomName1="B5" atomName2="B23"/>
143   <Bond atomName1="B6" atomName2="B30"/>
144   <Bond atomName1="B6" atomName2="B31"/>
145   <Bond atomName1="B6" atomName2="B32"/>
146   <Bond atomName1="B7" atomName2="B51"/>
147   <Bond atomName1="B8" atomName2="B51"/>
148   <Bond atomName1="B9" atomName2="B51"/>
149   <Bond atomName1="B12" atomName2="B33"/>
150   <Bond atomName1="B12" atomName2="B34"/>
151   <Bond atomName1="B12" atomName2="B35"/>
152   <Bond atomName1="B13" atomName2="B42"/>
153   <Bond atomName1="B14" atomName2="B42"/>
```

```
154     <Bond atomName1="B15" atomName2="B42"/>
155     <Bond atomName1="B18" atomName2="B36"/>
156     <Bond atomName1="B18" atomName2="B37"/>
157     <Bond atomName1="B18" atomName2="B38"/>
158     <Bond atomName1="B19" atomName2="B45"/>
159     <Bond atomName1="B20" atomName2="B45"/>
160     <Bond atomName1="B21" atomName2="B45"/>
161     <Bond atomName1="B24" atomName2="B39"/>
162     <Bond atomName1="B24" atomName2="B40"/>
163     <Bond atomName1="B24" atomName2="B41"/>
164     <Bond atomName1="B25" atomName2="B48"/>
165     <Bond atomName1="B26" atomName2="B48"/>
166     <Bond atomName1="B27" atomName2="B48"/>
167     <Bond atomName1="B30" atomName2="B51"/>
168     <Bond atomName1="B30" atomName2="B52"/>
169     <Bond atomName1="B30" atomName2="B53"/>
170     <Bond atomName1="B33" atomName2="B42"/>
171     <Bond atomName1="B33" atomName2="B43"/>
172     <Bond atomName1="B33" atomName2="B44"/>
173     <Bond atomName1="B36" atomName2="B45"/>
174     <Bond atomName1="B36" atomName2="B46"/>
175     <Bond atomName1="B36" atomName2="B47"/>
176     <Bond atomName1="B39" atomName2="B48"/>
177     <Bond atomName1="B39" atomName2="B49"/>
178     <Bond atomName1="B39" atomName2="B50"/>
179   </Residue>
180
181   <Residue name="DFB">
182     <Atom name="S1" type="SF" charge="-0.126900"/>
183     <Atom name="S2" type="SC" charge=" 0.092900"/>
184     <Atom name="S3" type="SC" charge="-0.143000"/>
185     <Atom name="S4" type="SC" charge=" 0.092900"/>
186     <Atom name="S5" type="SC" charge="-0.120000"/>
187     <Atom name="S6" type="SC" charge="-0.120000"/>
188     <Atom name="S7" type="SC" charge="-0.143000"/>
189     <Atom name="S8" type="SH" charge=" 0.154000"/>
190     <Atom name="S9" type="SH" charge=" 0.154000"/>
191     <Atom name="S10" type="SF" charge="-0.126900"/>
192     <Atom name="S11" type="SH" charge=" 0.143000"/>
193     <Atom name="S12" type="SH" charge=" 0.143000"/>
194     <Bond atomName1="S1" atomName2="S2"/>
195     <Bond atomName1="S2" atomName2="S3"/>
196     <Bond atomName1="S2" atomName2="S4"/>
197     <Bond atomName1="S3" atomName2="S5"/>
198     <Bond atomName1="S3" atomName2="S9"/>
199     <Bond atomName1="S4" atomName2="S7"/>
200     <Bond atomName1="S4" atomName2="S10"/>
201     <Bond atomName1="S5" atomName2="S6"/>
202     <Bond atomName1="S5" atomName2="S11"/>
203     <Bond atomName1="S6" atomName2="S7"/>
204     <Bond atomName1="S6" atomName2="S12"/>
205     <Bond atomName1="S7" atomName2="S8"/>
206   </Residue>
207   <!-- Merocyanine = Spiropyran open -->
208   <Residue name="MER">
209     <AllowPatch name="MER_H"/>
210     <Atom name="M1" type="PN1"  charge="0.321334 "/>
211     <Atom name="M2" type="PO1"  charge="-0.241266"/>
212     <Atom name="M3" type="PO1"  charge="-0.241266"/>
213     <Atom name="M4" type="PC6"  charge="-0.266866"/>
214     <Atom name="M5" type="PC7"  charge="0.062234 "/>
215     <Atom name="M6" type="PC6"  charge="-0.027766"/>
216     <Atom name="M7" type="PC7"  charge="-0.231766"/>
217     <Atom name="M8" type="PC7"  charge="-0.287766"/>
218     <Atom name="M9" type="PC5"  charge="0.554334 "/>
219     <Atom name="M10" type="PC2" charge="0.122234 "/>
```

```
220     <Atom name="M11" type="PC2" charge="-0.317766"/>
221     <Atom name="M12" type="PC3" charge="0.263034 "/>
222     <Atom name="M13" type="PO1" charge="-0.593866"/>
223     <Atom name="M14" type="PN2" charge="-0.427565"/>
224     <Atom name="M15" type="PC4" charge="0.020735 "/>
225     <Atom name="M16" type="PC1" charge="-0.126066"/>
226     <Atom name="M17" type="PC1" charge="0.112834 "/>
227     <Atom name="M18" type="PC4" charge="0.207034 "/>
228     <Atom name="M19" type="PC4" charge="-0.212166"/>
229     <Atom name="M20" type="PC5" charge="0.917834 "/>
230     <Atom name="M21" type="PC4" charge="-0.068366"/>
231     <Atom name="M22" type="PC4" charge="-0.068366"/>
232     <Atom name="M23" type="PC1" charge="-0.092766"/>
233     <Atom name="M24" type="PC1" charge="-0.137766"/>
234     <Atom name="M25" type="PC1" charge="-0.100766"/>
235     <Atom name="M26" type="PC1" charge="-0.089766"/>
236     <Atom name="M27" type="PO1" charge="-0.826566"/>
237     <Atom name="M28" type="PO1" charge="-0.826566"/>
238     <Atom name="M29" type="PH1" charge="0.152234 "/>
239     <Atom name="M30" type="PH1" charge="0.159234 "/>
240     <Atom name="M31" type="PH1" charge="0.155234 "/>
241     <Atom name="M32" type="PH1" charge="0.114234 "/>
242     <Atom name="M33" type="PH1" charge="0.207234 "/>
243     <Atom name="M34" type="PH2" charge="0.094434 "/>
244     <Atom name="M35" type="PH2" charge="0.094434 "/>
245     <Atom name="M36" type="PH3" charge="0.064434 "/>
246     <Atom name="M37" type="PH3" charge="0.064434 "/>
247     <Atom name="M38" type="PH3" charge="0.058934 "/>
248     <Atom name="M39" type="PH3" charge="0.058934 "/>
249     <Atom name="M40" type="PH3" charge="0.058934 "/>
250     <Atom name="M41" type="PH3" charge="0.058934 "/>
251     <Atom name="M42" type="PH3" charge="0.058934 "/>
252     <Atom name="M43" type="PH3" charge="0.058934 "/>
253     <Atom name="M44" type="PH1" charge="0.138234 "/>
254     <Atom name="M45" type="PH1" charge="0.136234 "/>
255     <Atom name="M46" type="PH1" charge="0.151234 "/>
256     <Atom name="M47" type="PH1" charge="0.246234 "/>
257     <Bond atomName1="M1" atomName2="M2"/>
258     <Bond atomName1="M1" atomName2="M3"/>
259     <Bond atomName1="M1" atomName2="M4"/>
260     <Bond atomName1="M4" atomName2="M5"/>
261     <Bond atomName1="M4" atomName2="M6"/>
262     <Bond atomName1="M5" atomName2="M8"/>
263     <Bond atomName1="M5" atomName2="M29"/>
264     <Bond atomName1="M6" atomName2="M7"/>
265     <Bond atomName1="M6" atomName2="M30"/>
266     <Bond atomName1="M7" atomName2="M9"/>
267     <Bond atomName1="M7" atomName2="M31"/>
268     <Bond atomName1="M8" atomName2="M9"/>
269     <Bond atomName1="M8" atomName2="M10"/>
270     <Bond atomName1="M9" atomName2="M13"/>
271     <Bond atomName1="M10" atomName2="M11"/>
272     <Bond atomName1="M10" atomName2="M32"/>
273     <Bond atomName1="M11" atomName2="M12"/>
274     <Bond atomName1="M11" atomName2="M33"/>
275     <Bond atomName1="M12" atomName2="M14"/>
276     <Bond atomName1="M12" atomName2="M15"/>
277     <Bond atomName1="M14" atomName2="M17"/>
278     <Bond atomName1="M14" atomName2="M18"/>
279     <Bond atomName1="M15" atomName2="M16"/>
280     <Bond atomName1="M15" atomName2="M21"/>
281     <Bond atomName1="M15" atomName2="M22"/>
282     <Bond atomName1="M16" atomName2="M17"/>
283     <Bond atomName1="M16" atomName2="M23"/>
284     <Bond atomName1="M17" atomName2="M26"/>
285     <Bond atomName1="M18" atomName2="M19"/>
```

```
286     <Bond atomName1="M18" atomName2="M34"/>
287     <Bond atomName1="M18" atomName2="M35"/>
288     <Bond atomName1="M19" atomName2="M20"/>
289     <Bond atomName1="M19" atomName2="M36"/>
290     <Bond atomName1="M19" atomName2="M37"/>
291     <Bond atomName1="M20" atomName2="M27"/>
292     <Bond atomName1="M20" atomName2="M28"/>
293     <Bond atomName1="M21" atomName2="M38"/>
294     <Bond atomName1="M21" atomName2="M39"/>
295     <Bond atomName1="M21" atomName2="M40"/>
296     <Bond atomName1="M22" atomName2="M41"/>
297     <Bond atomName1="M22" atomName2="M42"/>
298     <Bond atomName1="M22" atomName2="M43"/>
299     <Bond atomName1="M23" atomName2="M24"/>
300     <Bond atomName1="M23" atomName2="M44"/>
301     <Bond atomName1="M24" atomName2="M25"/>
302     <Bond atomName1="M24" atomName2="M45"/>
303     <Bond atomName1="M25" atomName2="M26"/>
304     <Bond atomName1="M25" atomName2="M46"/>
305     <Bond atomName1="M26" atomName2="M47"/>
306     <ExternalBond atomName="M27"/>
307     <ExternalBond atomName="M28"/>
308   </Residue>
309   <!-- Spiropyran (closed) -->
310   <Residue name="SPI">
311     <AllowPatch name="SPI_H"/>
312     <Atom name="P1" type="PN1"  charge="0.333434 "/>
313     <Atom name="P2" type="PO1"  charge="-0.217766"/>
314     <Atom name="P3" type="PO1"  charge="-0.217766"/>
315     <Atom name="P4" type="PC1"  charge="-0.220966"/>
316     <Atom name="P5" type="PC1"  charge="-0.002766"/>
317     <Atom name="P6" type="PC1"  charge="-0.023766"/>
318     <Atom name="P7" type="PC1"  charge="-0.167766"/>
319     <Atom name="P8" type="PC1"  charge="-0.137566"/>
320     <Atom name="P9" type="PC1"  charge="0.226334 "/>
321     <Atom name="P10" type="PC2" charge="-0.089966"/>
322     <Atom name="P11" type="PC3" charge="-0.144966"/>
323     <Atom name="P12" type="PC4" charge="0.461434 "/>
324     <Atom name="P13" type="PO2" charge="-0.321666"/>
325     <Atom name="P14" type="PN2" charge="-0.628766"/>
326     <Atom name="P15" type="PC4" charge="0.026534 "/>
327     <Atom name="P16" type="PC1" charge="-0.152066"/>
328     <Atom name="P17" type="PC1" charge="0.155834 "/>
329     <Atom name="P18" type="PC4" charge="0.221034 "/>
330     <Atom name="P19" type="PC4" charge="-0.199166"/>
331     <Atom name="P20" type="PC5" charge="0.919834 "/>
332     <Atom name="P21" type="PC4" charge="-0.066866"/>
333     <Atom name="P22" type="PC4" charge="-0.066866"/>
334     <Atom name="P23" type="PC1" charge="-0.087766"/>
335     <Atom name="P24" type="PC1" charge="-0.173766"/>
336     <Atom name="P25" type="PC1" charge="-0.093766"/>
337     <Atom name="P26" type="PC1" charge="-0.133766"/>
338     <Atom name="P27" type="PO1" charge="-0.832066"/>
339     <Atom name="P28" type="PO1" charge="-0.832066"/>
340     <Atom name="P29" type="PH1" charge="0.176234 "/>
341     <Atom name="P30" type="PH1" charge="0.176234 "/>
342     <Atom name="P31" type="PH1" charge="0.176234 "/>
343     <Atom name="P32" type="PH1" charge="0.150234 "/>
344     <Atom name="P33" type="PH1" charge="0.167234 "/>
345     <Atom name="P34" type="PH2" charge="0.060434 "/>
346     <Atom name="P35" type="PH2" charge="0.060434 "/>
347     <Atom name="P36" type="PH3" charge="0.049934 "/>
348     <Atom name="P37" type="PH3" charge="0.049934 "/>
349     <Atom name="P38" type="PH3" charge="0.050101 "/>
350     <Atom name="P39" type="PH3" charge="0.050101 "/>
351     <Atom name="P40" type="PH3" charge="0.050101 "/>
```

```
    <Atom name="P41" type="PH3" charge="0.050101 "/>
    <Atom name="P42" type="PH3" charge="0.050101 "/>
    <Atom name="P43" type="PH3" charge="0.050101 "/>
    <Atom name="P44" type="PH1" charge="0.125234 "/>
    <Atom name="P45" type="PH1" charge="0.125234 "/>
    <Atom name="P46" type="PH1" charge="0.139234 "/>
    <Atom name="P47" type="PH1" charge="0.238234 "/>
    <Bond atomName1="P1" atomName2="P2"/>
    <Bond atomName1="P1" atomName2="P3"/>
    <Bond atomName1="P1" atomName2="P4"/>
    <Bond atomName1="P4" atomName2="P5"/>
    <Bond atomName1="P4" atomName2="P6"/>
    <Bond atomName1="P5" atomName2="P8"/>
    <Bond atomName1="P5" atomName2="P29"/>
    <Bond atomName1="P6" atomName2="P7"/>
    <Bond atomName1="P6" atomName2="P30"/>
    <Bond atomName1="P7" atomName2="P9"/>
    <Bond atomName1="P7" atomName2="P31"/>
    <Bond atomName1="P8" atomName2="P9"/>
    <Bond atomName1="P8" atomName2="P10"/>
    <Bond atomName1="P9" atomName2="P13"/>
    <Bond atomName1="P10" atomName2="P11"/>
    <Bond atomName1="P10" atomName2="P32"/>
    <Bond atomName1="P11" atomName2="P12"/>
    <Bond atomName1="P11" atomName2="P33"/>
    <Bond atomName1="P12" atomName2="P13"/>
    <Bond atomName1="P12" atomName2="P14"/>
    <Bond atomName1="P12" atomName2="P15"/>
    <Bond atomName1="P14" atomName2="P17"/>
    <Bond atomName1="P14" atomName2="P18"/>
    <Bond atomName1="P15" atomName2="P16"/>
    <Bond atomName1="P15" atomName2="P21"/>
    <Bond atomName1="P15" atomName2="P22"/>
    <Bond atomName1="P16" atomName2="P17"/>
    <Bond atomName1="P16" atomName2="P23"/>
    <Bond atomName1="P17" atomName2="P26"/>
    <Bond atomName1="P18" atomName2="P19"/>
    <Bond atomName1="P18" atomName2="P34"/>
    <Bond atomName1="P18" atomName2="P35"/>
    <Bond atomName1="P19" atomName2="P20"/>
    <Bond atomName1="P19" atomName2="P36"/>
    <Bond atomName1="P19" atomName2="P37"/>
    <Bond atomName1="P20" atomName2="P27"/>
    <Bond atomName1="P20" atomName2="P28"/>
    <Bond atomName1="P21" atomName2="P38"/>
    <Bond atomName1="P21" atomName2="P39"/>
    <Bond atomName1="P21" atomName2="P40"/>
    <Bond atomName1="P22" atomName2="P41"/>
    <Bond atomName1="P22" atomName2="P42"/>
    <Bond atomName1="P22" atomName2="P43"/>
    <Bond atomName1="P23" atomName2="P24"/>
    <Bond atomName1="P23" atomName2="P44"/>
    <Bond atomName1="P24" atomName2="P25"/>
    <Bond atomName1="P24" atomName2="P45"/>
    <Bond atomName1="P25" atomName2="P26"/>
    <Bond atomName1="P25" atomName2="P46"/>
    <Bond atomName1="P26" atomName2="P47"/>
    <ExternalBond atomName="P27"/>
    <ExternalBond atomName="P28"/>
  </Residue>

  <Residue name="LNK">
    <AllowPatch name="LNK_ONE"/>
    <AllowPatch name="LNK_TWO"/>
    <AllowPatch name="LNK_THR"/>
    <AllowPatch name="LNK_FOU"/>
```

```
418     <AllowPatch name="LNK_FIV"/>
419     <AllowPatch name="LNK_SIX"/>
420     <AllowPatch name="LNK_SEV"/>
421     <AllowPatch name="LNK_EIG"/>
422     <Atom name="L1"  type="C4" charge="-0.064703 "/>
423     <Atom name="L2"  type="C1" charge="-0.064702 "/>
424     <Atom name="L3"  type="C4" charge="-0.064703 "/>
425     <Atom name="L4"  type="C3" charge="-0.081703 "/>
426     <Atom name="L5"  type="C4" charge="-0.064703 "/>
427     <Atom name="L6"  type="C1" charge="-0.064702 "/>
428     <Atom name="L7"  type="C4" charge="-0.064703 "/>
429     <Atom name="L8"  type="H1" charge="0.133297  "/>
430     <Atom name="L9"  type="C4" charge="-0.064703 "/>
431     <Atom name="L10" type="C4" charge="-0.064703 "/>
432     <Atom name="L11" type="C1" charge="-0.064702 "/>
433     <Atom name="L12" type="C1" charge="-0.064702 "/>
434     <Atom name="L13" type="H1" charge="0.133297  "/>
435     <Atom name="L14" type="C4" charge="-0.064703 "/>
436     <Atom name="L15" type="C4" charge="-0.064703 "/>
437     <Atom name="L16" type="C4" charge="-0.064703 "/>
438     <Atom name="L17" type="C1" charge="-0.064702 "/>
439     <Atom name="L18" type="C4" charge="-0.064703 "/>
440     <Atom name="L19" type="C4" charge="-0.064703 "/>
441     <Atom name="L20" type="C1" charge="-0.064702 "/>
442     <Atom name="L21" type="C1" charge="-0.064702 "/>
443     <Atom name="L22" type="H1" charge="0.133297  "/>
444     <Atom name="L23" type="H1" charge="0.133297  "/>
445     <Atom name="L24" type="C1" charge="-0.064702 "/>
446     <Atom name="L25" type="H1" charge="0.133297  "/>
447     <Atom name="L26" type="H1" charge="0.133297  "/>
448     <Atom name="L27" type="C4" charge="-0.064703 "/>
449     <Atom name="L28" type="H1" charge="0.133297  "/>
450     <Atom name="L29" type="H1" charge="0.133297  "/>
451     <Atom name="L30" type="H1" charge="0.133297  "/>
452     <Atom name="L31" type="O1" charge="-0.585703 "/>
453     <Atom name="L32" type="H1" charge="0.133297  "/>
454     <Atom name="L33" type="C3" charge="-0.081703 "/>
455     <Atom name="L34" type="H1" charge="0.133297  "/>
456     <Atom name="L35" type="H1" charge="0.133297  "/>
457     <Atom name="L36" type="H1" charge="0.133297  "/>
458     <Atom name="L37" type="H1" charge="0.133297  "/>
459     <Atom name="L38" type="C1" charge="-0.064702 "/>
460     <Atom name="L39" type="C1" charge="-0.064702 "/>
461     <Atom name="L40" type="C1" charge="-0.064702 "/>
462     <Atom name="L41" type="H1" charge="0.133297  "/>
463     <Atom name="L42" type="C1" charge="-0.064702 "/>
464     <Atom name="L43" type="C1" charge="-0.064702 "/>
465     <Atom name="L44" type="C1" charge="-0.064702 "/>
466     <Atom name="L45" type="C3" charge="-0.081703 "/>
467     <Atom name="L46" type="C3" charge="-0.081703 "/>
468     <Atom name="L47" type="H1" charge="0.133297  "/>
469     <Atom name="L48" type="C1" charge="-0.064702 "/>
470     <Atom name="L49" type="C1" charge="-0.064702 "/>
471     <Atom name="L50" type="O1" charge="-0.585703 "/>
472     <Atom name="L51" type="O1" charge="-0.585703 "/>
473     <Atom name="L52" type="O1" charge="-0.585703 "/>
474     <Atom name="L53" type="C1" charge="-0.064702 "/>
475     <Atom name="L54" type="C4" charge="-0.064703 "/>
476     <Atom name="L55" type="H1" charge="0.133297  "/>
477     <Atom name="L56" type="C4" charge="-0.064703 "/>
478     <Atom name="L57" type="C1" charge="-0.064702 "/>
479     <Atom name="L58" type="H1" charge="0.133297  "/>
480     <Atom name="L59" type="C1" charge="-0.064702 "/>
481     <Atom name="L60" type="H1" charge="0.133297  "/>
482     <Atom name="L61" type="H1" charge="0.133297  "/>
483     <Atom name="L62" type="C1" charge="-0.064702 "/>
```

```
484     <Atom name="L63" type="C1" charge="-0.064702 "/>
485     <Atom name="L64" type="H1" charge="0.133297  "/>
486     <Atom name="L65" type="C1" charge="-0.064702 "/>
487     <Atom name="L66" type="H1" charge="0.133297  "/>
488     <Atom name="L67" type="C2" charge="0.630297  "/>
489     <Atom name="L68" type="C2" charge="0.630297  "/>
490     <Atom name="L69" type="O1" charge="-0.585703 "/>
491     <Atom name="L70" type="C2" charge="0.630297  "/>
492     <Atom name="L71" type="O1" charge="-0.585703 "/>
493     <Atom name="L72" type="O1" charge="-0.585703 "/>
494     <Atom name="L73" type="O1" charge="-0.585703 "/>
495     <Atom name="L74" type="C2" charge="0.630297  "/>
496     <Bond atomName1="L1" atomName2="L2"/>
497     <Bond atomName1="L1" atomName2="L9"/>
498     <Bond atomName1="L1" atomName2="L18"/>
499     <Bond atomName1="L2" atomName2="L12"/>
500     <Bond atomName1="L2" atomName2="L13"/>
501     <Bond atomName1="L3" atomName2="L15"/>
502     <Bond atomName1="L3" atomName2="L7"/>
503     <Bond atomName1="L3" atomName2="L9"/>
504     <Bond atomName1="L4" atomName2="L57"/>
505     <Bond atomName1="L4" atomName2="L21"/>
506     <Bond atomName1="L4" atomName2="L70"/>
507     <Bond atomName1="L5" atomName2="L15"/>
508     <Bond atomName1="L5" atomName2="L6"/>
509     <Bond atomName1="L5" atomName2="L54"/>
510     <Bond atomName1="L6" atomName2="L16"/>
511     <Bond atomName1="L6" atomName2="L23"/>
512     <Bond atomName1="L7" atomName2="L16"/>
513     <Bond atomName1="L7" atomName2="L12"/>
514     <Bond atomName1="L8" atomName2="L17"/>
515     <Bond atomName1="L9" atomName2="L14"/>
516     <Bond atomName1="L10" atomName2="L14"/>
517     <Bond atomName1="L10" atomName2="L11"/>
518     <Bond atomName1="L10" atomName2="L27"/>
519     <Bond atomName1="L11" atomName2="L18"/>
520     <Bond atomName1="L11" atomName2="L25"/>
521     <Bond atomName1="L12" atomName2="L26"/>
522     <Bond atomName1="L14" atomName2="L17"/>
523     <Bond atomName1="L15" atomName2="L20"/>
524     <Bond atomName1="L16" atomName2="L19"/>
525     <Bond atomName1="L17" atomName2="L20"/>
526     <Bond atomName1="L18" atomName2="L56"/>
527     <Bond atomName1="L19" atomName2="L38"/>
528     <Bond atomName1="L19" atomName2="L40"/>
529     <Bond atomName1="L20" atomName2="L22"/>
530     <Bond atomName1="L21" atomName2="L37"/>
531     <Bond atomName1="L21" atomName2="L63"/>
532     <Bond atomName1="L24" atomName2="L46"/>
533     <Bond atomName1="L24" atomName2="L38"/>
534     <Bond atomName1="L24" atomName2="L61"/>
535     <Bond atomName1="L27" atomName2="L39"/>
536     <Bond atomName1="L27" atomName2="L48"/>
537     <Bond atomName1="L28" atomName2="L43"/>
538     <Bond atomName1="L29" atomName2="L42"/>
539     <Bond atomName1="L30" atomName2="L38"/>
540     <Bond atomName1="L31" atomName2="L70"/>
541     <Bond atomName1="L32" atomName2="L40"/>
542     <Bond atomName1="L33" atomName2="L65"/>
543     <Bond atomName1="L33" atomName2="L62"/>
544     <Bond atomName1="L33" atomName2="L67"/>
545     <Bond atomName1="L34" atomName2="L62"/>
546     <Bond atomName1="L35" atomName2="L63"/>
547     <Bond atomName1="L36" atomName2="L57"/>
548     <Bond atomName1="L39" atomName2="L42"/>
549     <Bond atomName1="L39" atomName2="L47"/>
```

```
550     <Bond atomName1="L40" atomName2="L44"/>
551     <Bond atomName1="L41" atomName2="L48"/>
552     <Bond atomName1="L42" atomName2="L45"/>
553     <Bond atomName1="L43" atomName2="L57"/>
554     <Bond atomName1="L43" atomName2="L54"/>
555     <Bond atomName1="L44" atomName2="L46"/>
556     <Bond atomName1="L44" atomName2="L66"/>
557     <Bond atomName1="L45" atomName2="L49"/>
558     <Bond atomName1="L45" atomName2="L74"/>
559     <Bond atomName1="L46" atomName2="L68"/>
560     <Bond atomName1="L48" atomName2="L49"/>
561     <Bond atomName1="L49" atomName2="L64"/>
562     <Bond atomName1="L50" atomName2="L68"/>
563     <Bond atomName1="L51" atomName2="L68"/>
564     <Bond atomName1="L52" atomName2="L67"/>
565     <Bond atomName1="L53" atomName2="L60"/>
566     <Bond atomName1="L53" atomName2="L56"/>
567     <Bond atomName1="L53" atomName2="L65"/>
568     <Bond atomName1="L54" atomName2="L63"/>
569     <Bond atomName1="L55" atomName2="L59"/>
570     <Bond atomName1="L56" atomName2="L59"/>
571     <Bond atomName1="L58" atomName2="L65"/>
572     <Bond atomName1="L59" atomName2="L62"/>
573     <Bond atomName1="L67" atomName2="L72"/>
574     <Bond atomName1="L69" atomName2="L74"/>
575     <Bond atomName1="L70" atomName2="L71"/>
576     <Bond atomName1="L73" atomName2="L74"/>
577     <ExternalBond atomName="L31"/>
578     <ExternalBond atomName="L50"/>
579     <ExternalBond atomName="L51"/>
580     <ExternalBond atomName="L52"/>
581     <ExternalBond atomName="L69"/>
582     <ExternalBond atomName="L71"/>
583     <ExternalBond atomName="L72"/>
584     <ExternalBond atomName="L73"/>
585   </Residue>
586   <Residue name="NOD">
587     <Atom name="N1" type="Zr"  charge=" 1.968"/>
588     <Atom name="N2" type="Zr"  charge=" 1.968"/>
589     <Atom name="N3" type="Zr"  charge=" 1.968"/>
590     <Atom name="N4" type="Zr"  charge=" 1.968"/>
591     <Atom name="N5" type="Zr"  charge=" 1.968"/>
592     <Atom name="N6" type="Zr"  charge=" 1.968"/>
593     <Atom name="N7" type="O2"  charge="−0.992"/>
594     <Atom name="N8" type="O2"  charge="−0.992"/>
595     <Atom name="N9" type="O3"  charge="−0.992"/>
596     <Atom name="N10" type="O3" charge="−0.992"/>
597     <Atom name="N11" type="O3" charge="−0.992"/>
598     <Atom name="N12" type="O3" charge="−0.992"/>
599     <Atom name="N13" type="O2" charge="−0.992"/>
600     <Atom name="N14" type="O2" charge="−0.992"/>
601     <Atom name="N15" type="H2" charge=" 0.448"/>
602     <Atom name="N16" type="H2" charge=" 0.448"/>
603     <Atom name="N17" type="H2" charge=" 0.448"/>
604     <Atom name="N18" type="H2" charge=" 0.448"/>
605     <Bond atomName1="N1" atomName2="N14"/>
606     <Bond atomName1="N1" atomName2="N11"/>
607     <Bond atomName1="N1" atomName2="N8"/>
608     <Bond atomName1="N1" atomName2="N9"/>
609     <Bond atomName1="N2" atomName2="N7"/>
610     <Bond atomName1="N2" atomName2="N10"/>
611     <Bond atomName1="N2" atomName2="N13"/>
612     <Bond atomName1="N2" atomName2="N12"/>
613     <Bond atomName1="N3" atomName2="N10"/>
614     <Bond atomName1="N3" atomName2="N11"/>
615     <Bond atomName1="N3" atomName2="N13"/>
```

```
    <Bond atomName1="N3" atomName2="N8"/>
    <Bond atomName1="N4" atomName2="N7"/>
    <Bond atomName1="N4" atomName2="N12"/>
    <Bond atomName1="N4" atomName2="N14"/>
    <Bond atomName1="N4" atomName2="N9"/>
    <Bond atomName1="N5" atomName2="N8"/>
    <Bond atomName1="N5" atomName2="N7"/>
    <Bond atomName1="N5" atomName2="N9"/>
    <Bond atomName1="N5" atomName2="N10"/>
    <Bond atomName1="N6" atomName2="N11"/>
    <Bond atomName1="N6" atomName2="N13"/>
    <Bond atomName1="N6" atomName2="N12"/>
    <Bond atomName1="N6" atomName2="N14"/>
    <Bond atomName1="N9" atomName2="N15"/>
    <Bond atomName1="N10" atomName2="N16"/>
    <Bond atomName1="N11" atomName2="N17"/>
    <Bond atomName1="N12" atomName2="N18"/>
    <ExternalBond atomName="N1"/>
    <ExternalBond atomName="N1"/>
    <ExternalBond atomName="N1"/>
    <ExternalBond atomName="N1"/>
    <ExternalBond atomName="N2"/>
    <ExternalBond atomName="N2"/>
    <ExternalBond atomName="N2"/>
    <ExternalBond atomName="N2"/>
    <ExternalBond atomName="N3"/>
    <ExternalBond atomName="N3"/>
    <ExternalBond atomName="N3"/>
    <ExternalBond atomName="N3"/>
    <ExternalBond atomName="N4"/>
    <ExternalBond atomName="N4"/>
    <ExternalBond atomName="N4"/>
    <ExternalBond atomName="N4"/>
    <ExternalBond atomName="N5"/>
    <ExternalBond atomName="N5"/>
    <ExternalBond atomName="N5"/>
    <ExternalBond atomName="N5"/>
    <ExternalBond atomName="N6"/>
    <ExternalBond atomName="N6"/>
    <ExternalBond atomName="N6"/>
    <ExternalBond atomName="N6"/>
  </Residue>
 </Residues>

 <Patches>
  <Patch name="LNK_ONE">
   <RemoveExternalBond atomName="L31"/>
  </Patch>
  <Patch name="LNK_TWO">
   <RemoveExternalBond atomName="L50"/>
  </Patch>
  <Patch name="LNK_THR">
   <RemoveExternalBond atomName="L52"/>
  </Patch>
  <Patch name="LNK_FOU">
   <RemoveExternalBond atomName="L69"/>
  </Patch>
  <Patch name="LNK_FIV">
   <RemoveExternalBond atomName="L71"/>
  </Patch>
  <Patch name="LNK_SIX">
   <RemoveExternalBond atomName="L51"/>
  </Patch>
  <Patch name="LNK_SEV">
   <RemoveExternalBond atomName="L72"/>
  </Patch>
```

```
<Patch name="LNK_EIG">
 <RemoveExternalBond atomName="L73"/>
</Patch>
<Patch name="SPI_H">
 <RemoveExternalBond atomName="P27"/>
 <RemoveExternalBond atomName="P28"/>
 <AddAtom name="P48" type="PHO" charge="0.424000"/>
 <AddBond atomName1="P27" atomName2="P48"/>
 <ChangeAtom name="P27" type="POH" charge="-0.580100"/>
 <ChangeAtom name="P1" type="PN1" charge=" 0.319200"/>
 <ChangeAtom name="P2" type="PO1" charge="-0.211000"/>
 <ChangeAtom name="P3" type="PO1" charge="-0.211000"/>
 <ChangeAtom name="P4" type="PC1" charge="-0.216200"/>
 <ChangeAtom name="P5" type="PC1" charge="-0.013000"/>
 <ChangeAtom name="P6" type="PC1" charge="-0.030000"/>
 <ChangeAtom name="P7" type="PC1" charge="-0.169000"/>
 <ChangeAtom name="P8" type="PC1" charge="-0.153800"/>
 <ChangeAtom name="P9" type="PC1" charge=" 0.186100"/>
 <ChangeAtom name="P10" type="PC2" charge="-0.080200"/>
 <ChangeAtom name="P11" type="PC3" charge="-0.182200"/>
 <ChangeAtom name="P12" type="PC4" charge=" 0.451200"/>
 <ChangeAtom name="P13" type="PO2" charge="-0.329900"/>
 <ChangeAtom name="P14" type="PN2" charge="-0.648000"/>
 <ChangeAtom name="P15" type="PC4" charge=" 0.015300"/>
 <ChangeAtom name="P16" type="PC1" charge="-0.132300"/>
 <ChangeAtom name="P17" type="PC1" charge=" 0.128600"/>
 <ChangeAtom name="P18" type="PC4" charge=" 0.205800"/>
 <ChangeAtom name="P19" type="PC4" charge="-0.185400"/>
 <ChangeAtom name="P20" type="PC5" charge=" 0.622100"/>
 <ChangeAtom name="P21" type="PC4" charge="-0.082100"/>
 <ChangeAtom name="P22" type="PC4" charge="-0.082100"/>
 <ChangeAtom name="P23" type="PC1" charge="-0.081000"/>
 <ChangeAtom name="P24" type="PC1" charge="-0.158000"/>
 <ChangeAtom name="P25" type="PC1" charge="-0.098000"/>
 <ChangeAtom name="P26" type="PC1" charge="-0.163000"/>
 <ChangeAtom name="P28" type="PO1" charge="-0.486000"/>
 <ChangeAtom name="P29" type="PH1" charge=" 0.173000"/>
 <ChangeAtom name="P30" type="PH1" charge=" 0.176000"/>
 <ChangeAtom name="P31" type="PH1" charge=" 0.168000"/>
 <ChangeAtom name="P32" type="PH1" charge=" 0.149000"/>
 <ChangeAtom name="P33" type="PH1" charge=" 0.158000"/>
 <ChangeAtom name="P34" type="PH2" charge=" 0.067700"/>
 <ChangeAtom name="P35" type="PH2" charge=" 0.067700"/>
 <ChangeAtom name="P36" type="PH3" charge=" 0.079700"/>
 <ChangeAtom name="P37" type="PH3" charge=" 0.079700"/>
 <ChangeAtom name="P38" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P39" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P40" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P41" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P42" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P43" type="PH3" charge=" 0.046700"/>
 <ChangeAtom name="P44" type="PH1" charge=" 0.136000"/>
 <ChangeAtom name="P45" type="PH1" charge=" 0.135000"/>
 <ChangeAtom name="P46" type="PH1" charge=" 0.134000"/>
 <ChangeAtom name="P47" type="PH1" charge=" 0.136000"/>
</Patch>
<Patch name="MER_H">
 <RemoveExternalBond atomName="M27"/>
 <RemoveExternalBond atomName="M28"/>
 <AddAtom name="M48" type="PHO" charge="0.457000"/>
 <AddBond atomName1="M27" atomName2="M48"/>
 <ChangeAtom name="M27" type="POH" charge="-0.579100"/>
 <ChangeAtom name="M1" type="PN1" charge=" 0.301100"/>
 <ChangeAtom name="M2" type="PO1" charge="-0.215000"/>
 <ChangeAtom name="M3" type="PO1" charge="-0.215000"/>
 <ChangeAtom name="M4" type="PC6" charge="-0.212100"/>
```

```
748     <ChangeAtom name="M5" type="PC7" charge=" 0.021000"/>
749     <ChangeAtom name="M6" type="PC6" charge="-0.037000"/>
750     <ChangeAtom name="M7" type="PC7" charge="-0.226000"/>
751     <ChangeAtom name="M8" type="PC7" charge="-0.249000"/>
752     <ChangeAtom name="M9" type="PC5" charge=" 0.556100"/>
753     <ChangeAtom name="M10" type="PC2" charge=" 0.103000"/>
754     <ChangeAtom name="M11" type="PC2" charge="-0.309000"/>
755     <ChangeAtom name="M12" type="PC3" charge=" 0.220800"/>
756     <ChangeAtom name="M13" type="PO1" charge="-0.616100"/>
757     <ChangeAtom name="M14" type="PN2" charge="-0.524800"/>
758     <ChangeAtom name="M15" type="PC4" charge=" 0.022500"/>
759     <ChangeAtom name="M16" type="PC1" charge="-0.135300"/>
760     <ChangeAtom name="M17" type="PC1" charge=" 0.129600"/>
761     <ChangeAtom name="M18" type="PC4" charge=" 0.223800"/>
762     <ChangeAtom name="M19" type="PC4" charge="-0.215400"/>
763     <ChangeAtom name="M20" type="PC5" charge=" 0.631100"/>
764     <ChangeAtom name="M21" type="PC4" charge="-0.082100"/>
765     <ChangeAtom name="M22" type="PC4" charge="-0.082100"/>
766     <ChangeAtom name="M23" type="PC1" charge="-0.078000"/>
767     <ChangeAtom name="M24" type="PC1" charge="-0.148000"/>
768     <ChangeAtom name="M25" type="PC1" charge="-0.096000"/>
769     <ChangeAtom name="M26" type="PC1" charge="-0.146000"/>
770     <ChangeAtom name="M28" type="PO1" charge="-0.501000"/>
771     <ChangeAtom name="M29" type="PH1" charge=" 0.160000"/>
772     <ChangeAtom name="M30" type="PH1" charge=" 0.170000"/>
773     <ChangeAtom name="M31" type="PH1" charge=" 0.161000"/>
774     <ChangeAtom name="M32" type="PH1" charge=" 0.126000"/>
775     <ChangeAtom name="M33" type="PH1" charge=" 0.173000"/>
776     <ChangeAtom name="M34" type="PH2" charge=" 0.078700"/>
777     <ChangeAtom name="M35" type="PH2" charge=" 0.078700"/>
778     <ChangeAtom name="M36" type="PH3" charge=" 0.091200"/>
779     <ChangeAtom name="M37" type="PH3" charge=" 0.091200"/>
780     <ChangeAtom name="M38" type="PH3" charge=" 0.049867"/>
781     <ChangeAtom name="M39" type="PH3" charge=" 0.049867"/>
782     <ChangeAtom name="M40" type="PH3" charge=" 0.049867"/>
783     <ChangeAtom name="M41" type="PH3" charge=" 0.049867"/>
784     <ChangeAtom name="M42" type="PH3" charge=" 0.049867"/>
785     <ChangeAtom name="M43" type="PH3" charge=" 0.049867"/>
786     <ChangeAtom name="M44" type="PH1" charge=" 0.141000"/>
787     <ChangeAtom name="M45" type="PH1" charge=" 0.141000"/>
788     <ChangeAtom name="M46" type="PH1" charge=" 0.142000"/>
789     <ChangeAtom name="M47" type="PH1" charge=" 0.149000"/>
790   </Patch>
791 </Patches>
792
793 <HarmonicBondForce>
794   <Bond type1="C1" type2="H1" length="0.1080" k="304106.80"/>
795   <Bond type1="C2" type2="O1" length="0.1273" k="451872.00"/>
796   <Bond type1="C2" type2="C3" length="0.1487" k="293928.30"/>
797   <Bond type1="C1" type2="C1" length="0.1393" k="401664.00"/>
798   <Bond type1="C1" type2="C3" length="0.1393" k="401664.00"/>
799   <Bond type1="Zr" type2="O2" length="0.2098" k="107733.80"/>
800   <Bond type1="Zr" type2="O1" length="0.2232" k="287290.20"/>
801   <!--Assumptions, O3 approx O2, C4 approx C1, H2 Acpype -->
802   <Bond type1="Zr" type2="O3" length="0.2098" k="107733.80"/>
803   <Bond type1="C4" type2="C4" length="0.1393" k="401664.00"/>
804   <Bond type1="C4" type2="C1" length="0.1393" k="401664.00"/>
805   <Bond type1="H2" type2="O3" length="0.0973" k="310787.52"/>
806   <Bond type1="H2" type2="O1" length="0.0973" k="310787.52"/>
807   <!--O4 for node hydroxy ligands, O4 Zr bond like O1 -->
808   <Bond type1="H2" type2="O4" length="0.0973" k="310787.52"/>
809   <Bond type1="Zr" type2="O4" length="0.2232" k="287290.20"/>
810   <!-- water spce -->
811   <Bond type1="O" type2="H" length="0.1000" k="376560.00"/>
812   <!-- o-DFB from acpype -->
813   <Bond type1="SC" type2="SF" length="0.1349" k="299407.04"/>
```

```
814   <Bond type1="SC" type2="SC" length="0.1398" k="385848.48"/>
815   <Bond type1="SC" type2="SH" length="0.1086" k="289365.44"/>
816   <!-- Spiro from acpype -->
817   <Bond type1="PN1" type2="PO1" length="0.1226" k="620738.24"/>
818   <Bond type1="PC1" type2="PN1" length="0.1469" k="269198.56"/>
819   <Bond type1="PC1" type2="PC1" length="0.1398" k="385848.48"/>
820   <Bond type1="PC1" type2="PH1" length="0.1086" k="289365.44"/>
821   <Bond type1="PC1" type2="PC2" length="0.1476" k="302335.84"/>
822   <Bond type1="PC1" type2="PO2" length="0.1370" k="315138.88"/>
823   <Bond type1="PC3" type2="PC2" length="0.1346" k="457980.64"/>
824   <Bond type1="PC2" type2="PH1" length="0.1088" k="286604.00"/>
825   <Bond type1="PC3" type2="PC4" length="0.1510" k="273466.24"/>
826   <Bond type1="PC3" type2="PH1" length="0.1088" k="287106.08"/>
827   <Bond type1="PC4" type2="PO2" length="0.1432" k="258236.48"/>
828   <Bond type1="PC4" type2="PN2" length="0.1464" k="273298.88"/>
829   <Bond type1="PC4" type2="PC4" length="0.1538" k="251793.12"/>
830   <Bond type1="PC1" type2="PN2" length="0.1386" k="349698.72"/>
831   <Bond type1="PC4" type2="PC1" length="0.1516" k="268612.80"/>
832   <Bond type1="PC4" type2="PH2" length="0.1097" k="276646.08"/>
833   <Bond type1="PC5" type2="PC4" length="0.1524" k="261918.40"/>
834   <Bond type1="PC4" type2="PH3" length="0.1097" k="276646.08"/>
835   <Bond type1="PC5" type2="PO1" length="0.1218" k="533627.36"/>
836   <!-- Spiro open -->
837   <Bond type1="PC2" type2="PC2" length="0.1457" k="320327.04"/>
838   <Bond type1="PC3" type2="PN2" length="0.1387" k="348276.16"/>
839   <Bond type1="PC6" type2="PN1" length="0.1429" k="304762.56"/>
840   <Bond type1="PC6" type2="PC7" length="0.1373" k="419153.12"/>
841   <Bond type1="PC6" type2="PC6" length="0.1428" k="351288.64"/>
842   <Bond type1="PC7" type2="PC7" length="0.1428" k="351288.64"/>
843   <Bond type1="PC7" type2="PH1" length="0.1084" k="292126.88"/>
844   <Bond type1="PC6" type2="PH1" length="0.1084" k="292126.88"/>
845   <Bond type1="PC5" type2="PC7" length="0.1468" k="310452.80"/>
846   <Bond type1="PC7" type2="PC2" length="0.1366" k="429278.40"/>
847   <!-- Spiro H -->
848   <Bond type1="PC5" type2="POH" length="0.1351" k="334803.68"/>
849   <Bond type1="PHO" type2="POH" length="0.0973" k="310787.52"/>
850   <!-- TBA Br -->
851   <Bond type1="BC1" type2="BN1" length="0.1511" k="237065.44"/>
852   <Bond type1="BC1" type2="BC1" length="0.1538" k="251793.12"/>
853   <Bond type1="BC1" type2="BH2" length="0.1091" k="283424.16"/>
854   <Bond type1="BC1" type2="BH1" length="0.1097" k="276646.08"/>
855 </HarmonicBondForce>
856
857 <HarmonicAngleForce>
858   <Angle type1="O1" type2="C2" type3="O1" angle="2.18166" k="1213.360"/>
859   <Angle type1="C3" type2="C2" type3="O1" angle="2.04727" k="456.013"/>
860   <Angle type1="C1" type2="C3" type3="C1" angle="2.09440" k="753.120"/>
861   <Angle type1="C3" type2="C1" type3="C1" angle="2.09440" k="753.120"/>
862   <Angle type1="C1" type2="C3" type3="C2" angle="2.09440" k="290.201"/>
863   <Angle type1="C3" type2="C1" type3="H1" angle="2.09440" k="309.616"/>
864   <Angle type1="C1" type2="C1" type3="H1" angle="2.09440" k="309.616"/>
865   <Angle type1="Zr" type2="O1" type3="C2" angle="2.37016" k="231.637"/>
866   <Angle class1="NodeO" class2="Zr" class3="NodeO" angle="1.24093" k="115.776"/>
867   <Angle class1="NodeO" class2="Zr" class3="NodeO" angle="1.75231" k="115.776"/>
868   <!-- extra O1-Zr-O1 + O4 like O1 -->
869   <Angle class1="Ligand" type2="Zr" class3="Ligand" angle="1.32645" k="115.776"/>
870   <Angle class1="Ligand" type2="Zr" class3="Ligand" angle="1.42942" k="115.776"/>
871   <Angle class1="Ligand" type2="Zr" class3="Ligand" angle="1.21824" k="115.776"/>
872   <Angle class1="Ligand" type2="Zr" class3="Ligand" angle="1.29852" k="115.776"/>
873   <!-- OX-Zr-O1 use O3 values , O2-Zr-O2 is O3-Zr-O3 + O4 like O1 -->
874   <Angle class1="NodeO" type2="Zr" class3="Ligand" angle="1.49575" k="115.776"/>
875   <Angle class1="NodeO" type2="Zr" class3="Ligand" angle="2.16944" k="115.776"/>
876   <Angle class1="NodeO" type2="Zr" class3="Ligand" angle="2.22704" k="115.776"/>
877   <Angle class1="NodeO" type2="Zr" class3="Ligand" angle="2.50455" k="115.776"/>
878   <Angle class1="NodeO" type2="Zr" class3="Ligand" angle="2.62672" k="115.776"/>
879   <!-- Acpype linker -->
```

```
<Angle type1="C4" type2="C4" type3="C4" angle="2.11377" k="535.552"/>
<Angle type1="C4" type2="C4" type3="C1" angle="2.11377" k="535.552"/>
<Angle type1="C1" type2="C4" type3="C1" angle="2.06612" k="558.146"/>
<Angle type1="C4" type2="C1" type3="C1" angle="2.10644" k="554.798"/>
<Angle type1="C4" type2="C1" type3="C4" angle="1.94639" k="575.7184"/>
<Angle type1="C4" type2="C1" type3="H1" angle="2.09195" k="401.664"/>
<Angle type1="C2" type2="O1" type3="H2" angle="1.85965" k="417.563"/>
<!-- Geometry optimized OpenBabel, same force as in all node angles -->
<Angle type1="Zr" type2="O2" type3="Zr" angle="1.71915" k="115.776"/>
<Angle type1="Zr" type2="O3" type3="Zr" angle="1.73486" k="115.776"/>
<Angle type1="H2" type2="O3" type3="Zr" angle="2.06647" k="115.776"/>
<!-- O4 H2 angle from acpype acid angle -->
<Angle type1="Zr" type2="O4" type3="H2" angle="1.85965" k="417.563"/>
<!-- water spce -->
<Angle type1="H" type2="O" type3="H" angle="1.91055" k="460.240"/>
<!-- o-DFB from acpype -->
<Angle type1="SF" type2="SC" type3="SC" angle="2.07624" k="561.4928"/>
<Angle type1="SC" type2="SC" type3="SC" angle="2.09474" k="557.3088"/>
<Angle type1="SC" type2="SC" type3="SH" angle="2.09230" k="403.3376"/>
<!-- Spiro from acpype -->
<Angle type1="PN1" type2="PC1" type3="PC1" angle="2.07712" k="558.9824"/>
<Angle type1="PO1" type2="PN1" type3="PO1" angle="2.18306" k="641.8256"/>
<Angle type1="PO1" type2="PN1" type3="PC1" angle="2.05530" k="574.8816"/>
<Angle type1="PC1" type2="PC1" type3="PC1" angle="2.09474" k="557.3088"/>
<Angle type1="PC1" type2="PC1" type3="PH1" angle="2.09230" k="403.3376"/>
<Angle type1="PC1" type2="PC1" type3="PC2" angle="2.10871" k="539.7360"/>
<Angle type1="PC1" type2="PC1" type3="PO2" angle="2.08043" k="582.4128"/>
<Angle type1="PC1" type2="PC2" type3="PC3" angle="2.12546" k="546.4304"/>
<Angle type1="PC1" type2="PC2" type3="PH1" angle="2.00940" k="393.2960"/>
<Angle type1="PC1" type2="PO2" type3="PC4" angle="2.05879" k="523.0000"/>
<Angle type1="PC2" type2="PC3" type3="PC4" angle="2.14937" k="535.5520"/>
<Angle type1="PC2" type2="PC3" type3="PH1" angle="2.10225" k="414.2160"/>
<Angle type1="PC3" type2="PC2" type3="PH1" angle="2.09335" k="415.0528"/>
<Angle type1="PC3" type2="PC4" type3="PO2" angle="1.89473" k="573.2080"/>
<Angle type1="PC3" type2="PC4" type3="PN2" angle="1.92702" k="559.8192"/>
<Angle type1="PC3" type2="PC4" type3="PC4" angle="1.94709" k="530.5312"/>
<Angle type1="PC4" type2="PC3" type3="PH1" angle="2.01900" k="384.0912"/>
<Angle type1="PC4" type2="PN2" type3="PC1" angle="2.09405" k="530.5312"/>
<Angle type1="PC4" type2="PN2" type3="PC4" angle="1.99858" k="528.8576"/>
<Angle type1="PC1" type2="PC4" type3="PC4" angle="1.95599" k="528.0208"/>
<Angle type1="PC4" type2="PC4" type3="PC4" angle="1.94622" k="526.3472"/>
<Angle type1="PO2" type2="PC4" type3="PN2" angle="1.90468" k="595.8016"/>
<Angle type1="PO2" type2="PC4" type3="PC4" angle="1.88443" k="569.0240"/>
<Angle type1="PN2" type2="PC4" type3="PC4" angle="1.92789" k="553.9616"/>
<Angle type1="PN2" type2="PC1" type3="PC1" angle="2.11098" k="571.5344"/>
<Angle type1="PN2" type2="PC4" type3="PH2" angle="1.91620" k="415.0528"/>
<Angle type1="PC4" type2="PC1" type3="PC1" angle="2.10783" k="531.3680"/>
<Angle type1="PC4" type2="PC4" type3="PH3" angle="1.91637" k="387.4384"/>
<Angle type1="PC4" type2="PC4" type3="PC5" angle="1.93801" k="529.6944"/>
<Angle type1="PC4" type2="PC4" type3="PH2" angle="1.91218" k="388.2752"/>
<Angle type1="PC4" type2="PC5" type3="PO1" angle="2.15025" k="564.0032"/>
<Angle type1="PC5" type2="PC4" type3="PH3" angle="1.89839" k="392.4592"/>
<Angle type1="PO1" type2="PC5" type3="PO1" angle="2.27329" k="651.8672"/>
<Angle type1="PH2" type2="PC4" type3="PH2" angle="1.89298" k="328.0256"/>
<Angle type1="PH3" type2="PC4" type3="PH3" angle="1.87763" k="329.6992"/>
<!-- Mero -->
<Angle type1="PN1" type2="PC6" type3="PC7" angle="2.24606" k="549.7776"/>
<Angle type1="PN1" type2="PC6" type3="PC6" angle="2.19824" k="545.5936"/>
<Angle type1="PO1" type2="PN1" type3="PC6" angle="2.05059" k="586.5968"/>
<Angle type1="PC6" type2="PC7" type3="PC7" angle="1.99299" k="570.6976"/>
<Angle type1="PC6" type2="PC7" type3="PH1" angle="2.12511" k="405.8480"/>
<Angle type1="PC6" type2="PC6" type3="PC7" angle="1.99299" k="570.6976"/>
<Angle type1="PC6" type2="PC6" type3="PH1" angle="2.11307" k="394.1328"/>
<Angle type1="PC7" type2="PC7" type3="PC5" angle="2.14134" k="532.2048"/>
<Angle type1="PC7" type2="PC7" type3="PC2" angle="2.14187" k="551.4512"/>
<Angle type1="PC6" type2="PC7" type3="PC5" angle="2.11796" k="544.7568"/>
```

```
946   <Angle type1="PC7" type2="PC6" type3="PH1" angle="2.12511" k="405.848"/>
947   <Angle type1="PC7" type2="PC5" type3="PC7" angle="2.02179" k="540.5728"/>
948   <Angle type1="PC7" type2="PC5" type3="PO1" angle="2.16299" k="578.2288"/>
949   <Angle type1="PC7" type2="PC7" type3="PH1" angle="2.11307" k="394.1328"/>
950   <Angle type1="PC7" type2="PC2" type3="PC2" angle="2.17032" k="541.4096"/>
951   <Angle type1="PC7" type2="PC2" type3="PH1" angle="2.00626" k="419.2368"/>
952   <Angle type1="PC5" type2="PC7" type3="PH1" angle="2.03575" k="392.4592"/>
953   <Angle type1="PC5" type2="PC7" type3="PC2" angle="2.11150" k="528.0208"/>
954   <Angle type1="PC2" type2="PC2" type3="PC3" angle="2.15129" k="547.2672"/>
955   <Angle type1="PC2" type2="PC2" type3="PH1" angle="2.03593" k="394.9696"/>
956   <Angle type1="PC2" type2="PC3" type3="PN2" angle="2.10679" k="582.4128"/>
957   <Angle type1="PC3" type2="PN2" type3="PC1" angle="2.22634" k="529.6944"/>
958   <Angle type1="PC3" type2="PN2" type3="PC4" angle="2.15915" k="522.1632"/>
959   <Angle type1="PC3" type2="PC4" type3="PC1" angle="1.96332" k="532.2048"/>
960   <Angle type1="PN2" type2="PC3" type3="PC4" angle="2.02825" k="558.1456"/>
961   <!-- Spiro H -->
962   <Angle type1="PC4" type2="PC5" type3="POH" angle="1.96751" k="572.3712"/>
963   <Angle type1="PC5" type2="POH" type3="PHO" angle="1.85965" k="417.5632"/>
964   <Angle type1="POH" type2="PC5" type3="PO1" angle="2.13105" k="635.1312"/>
965   <!-- TBA Br -->
966   <Angle type1="BN1" type2="BC1" type3="BC1" angle="1.99334" k="537.2256"/>
967   <Angle type1="BN1" type2="BC1" type3="BH2" angle="1.88513" k="406.6848"/>
968   <Angle type1="BC1" type2="BN1" type3="BC1" angle="1.91393" k="523.8368"/>
969   <Angle type1="BC1" type2="BC1" type3="BC1" angle="1.94622" k="526.3472"/>
970   <Angle type1="BC1" type2="BC1" type3="BH1" angle="1.91637" k="387.4384"/>
971   <Angle type1="BC1" type2="BC1" type3="BH2" angle="1.92964" k="386.6016"/>
972   <Angle type1="BH1" type2="BC1" type3="BH1" angle="1.87763" k="329.6992"/>
973   <Angle type1="BH2" type2="BC1" type3="BH2" angle="1.91550" k="328.0256"/>
974  </HarmonicAngleForce>
975
976  <PeriodicTorsionForce>
977   <Proper type1="Zr" type2="O1" type3="C2" type4="C3" periodicity1="2" phase1="
       3.14159" k1="86.837"/>
978   <!-- Node from BTW-FF https://pubs.acs.org/doi/10.1021/ct500515h -->
979   <!-- <Proper type1="O2" type2="Zr" type3="O1" type4="C2" periodicity1="2" phase1=
       "3.14159" k1="8.63578"/> -->
980   <!-- <Proper type1="Zr" type2="O1" type3="C2" type4="O1" periodicity1="2" phase1=
       "3.14159" k1="8.43913"/> -->
981   <!-- <Proper type1="O1" type2="Zr" type3="O1" type4="C2" periodicity1="2" phase1=
       "3.14159" k1="3.59824"/> -->
982   <!-- <Proper type1="O1" type2="Zr" type3="O2" type4="Zr" periodicity1="2" phase1=
       "3.14159" k1="4.18400"/> -->
983   <!-- <Proper type1="C2" type2="O1" type3="Zr" type4="O3" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
984   <!-- <Proper type1="H2" type2="O3" type3="Zr" type4="O1" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
985   <!-- <Proper type1="H2" type2="O3" type3="Zr" type4="O1" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
986   <!-- <Proper type1="Zr" type2="O3" type3="Zr" type4="O1" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
987   <!-- <Proper type1="Zr" type2="O3" type3="Zr" type4="O3" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
988   <!-- <Proper type1="Zr" type2="O3" type3="Zr" type4="O2" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
989   <!-- <Proper type1="H2" type2="O3" type3="Zr" type4="O3" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
990   <!-- <Proper type1="Zr" type2="O2" type3="Zr" type4="O3" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
991   <!-- <Proper type1="Zr" type2="O2" type3="Zr" type4="O2" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
992   <!-- <Proper type1="H2" type2="O3" type3="Zr" type4="O2" periodicity1="2" phase1=
       "3.14159" k1="20.92000"/> -->
993   <!-- Linker -->
994   <Proper type1="O1" type2="C2" type3="C3" type4="C1" periodicity1="2" phase1="
       3.14159" k1="10.460"/>
```

```
995  <Proper type1="H1" type2="C1" type3="C3" type4="C2" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
996  <Proper type1="C2" type2="C3" type3="C1" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
997  <Proper type1="C1" type2="C3" type3="C1" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
998  <Proper type1="C1" type2="C3" type3="C1" type4="H1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
999  <Proper type1="C3" type2="C1" type3="C1" type4="H1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1000 <Proper type1="C3" type2="C1" type3="C1" type4="C3" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1001 <Proper type1="H1" type2="C1" type3="C1" type4="H1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1002 <!-- C4 approx C3 -->
1003 <Proper type1="H1" type2="C1" type3="C1" type4="C4" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1004 <Proper type1="C3" type2="C1" type3="C1" type4="C4" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1005 <Proper type1="C1" type2="C4" type3="C1" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1006 <Proper type1="C1" type2="C4" type3="C1" type4="H1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1007 <!-- C4-C4 approx like all rings -->
1008 <Proper type1="C4" type2="C4" type3="C1" type4="H1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1009 <Proper type1="C4" type2="C4" type3="C1" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1010 <Proper type1="C1" type2="C4" type3="C4" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1011 <Proper type1="C1" type2="C4" type3="C4" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1012 <Proper type1="C4" type2="C1" type3="C4" type4="C4" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1013 <Proper type1="C4" type2="C4" type3="C4" type4="C1" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1014 <Proper type1="C4" type2="C4" type3="C4" type4="C4" periodicity1="2" phase1="
       3.14159" k1="12.552"/>
1015 <Proper type1="C3" type2="C2" type3="O1" type4="H2" periodicity1="2" phase1="
       3.14159" k1="9.6232"/>
1016 <Proper type1="O1" type2="C2" type3="O1" type4="H2" periodicity1="2" phase1="
       3.14159" k1="9.6232" periodicity2="1" phase2="0.0" k2="7.9496"/>
1017 <!-- o-DFB from acpype -->
1018 <Proper type1="" type2="SC" type3="SC" type4="" periodicity1="2" phase1="3.14159"
       k1="15.167"/>
1019 <!-- Spiro from acpype -->
1020 <Proper type1="" type2="PC1" type3="PN1" type4="" periodicity1="2" phase1="
       3.14159" k1="2.5104"/>
1021 <Proper type1="" type2="PC1" type3="PC1" type4="" periodicity1="2" phase1="
       3.14159" k1="15.167"/>
1022 <Proper type1="PC1" type2="PC1" type3="PC2" type4="PC3" periodicity1="2" phase1="
       3.14159" k1="27.8236"/>
1023 <Proper type1="PC1" type2="PC1" type3="PC2" type4="PH1" periodicity1="2" phase1="
       3.14159" k1="27.8236"/>
1024 <Proper type1="" type2="PC1" type3="PO2" type4="" periodicity1="2" phase1="
       3.14159" k1="3.7656"/>
1025 <Proper type1="" type2="PC3" type3="PC2" type4="" periodicity1="2" phase1="
       3.14159" k1="27.8236"/>
1026 <Proper type1="" type2="PC3" type3="PC4" type4="" periodicity1="2" phase1="0.0"
       k1="0.0"/>
1027 <Proper type1="" type2="PC4" type3="PO2" type4="" periodicity1="3" phase1="0.0"
       k1="1.602472"/>
1028 <Proper type1="" type2="PC4" type3="PN2" type4="" periodicity1="2" phase1="0.0"
       k1="0.0"/>
1029 <Proper type1="" type2="PC4" type3="PC4" type4="" periodicity1="3" phase1="0.0"
       k1="0.652704"/>
```

```
1030  <Proper type1="" type2="PC1" type3="PN2" type4="" periodicity1="2" phase1="
        3.14159" k1="4.3932"/>
1031  <Proper type1="" type2="PC4" type3="PC1" type4="" periodicity1="2" phase1="0.0"
        k1="0.0"/>
1032  <Proper type1="PH3" type2="PC4" type3="PC4" type4="PC4" periodicity1="3" phase1="
        0.0" k1="0.66944"/>
1033  <Proper type1="" type2="PC5" type3="PC4" type4="" periodicity1="2" phase1="
        3.14159" k1="0.0"/>
1034  <Proper type1="PH3" type2="PC4" type3="PC5" type4="PO1" periodicity1="1" phase1="
        0.0" k1="3.3472" periodicity2="2" phase2="0.0" k2="0.0" periodicity3="3" phase3
        ="3.14159" k3="0.33472"/>
1035  <!-- Spiro open -->
1036  <Proper type1="PO1" type2="PN1" type3="PC6" type4="PC7" periodicity1="2" phase1="
        3.14159" k1="3.138"/>
1037  <Proper type1="PO1" type2="PN1" type3="PC6" type4="PC6" periodicity1="2" phase1="
        3.14159" k1="3.138"/>
1038  <Proper type1="" type2="PC6" type3="PC7" type4="" periodicity1="2" phase1="
        3.14159" k1="16.736"/>
1039  <Proper type1="" type2="PC6" type3="PC6" type4="" periodicity1="2" phase1="
        3.14159" k1="16.736"/>
1040  <Proper type1="" type2="PC7" type3="PC7" type4="" periodicity1="2" phase1="
        3.14159" k1="16.736"/>
1041  <Proper type1="" type2="PC5" type3="PC7" type4="" periodicity1="2" phase1="
        3.14159" k1="12.029"/>
1042  <Proper type1="PC7" type2="PC7" type3="PC2" type4="PC2" periodicity1="2" phase1="
        3.14159" k1="4.184"/>
1043  <Proper type1="" type2="PC2" type3="PC2" type4="" periodicity1="2" phase1="
        3.14159" k1="4.184"/>
1044  <Proper type1="PC5" type2="PC7" type3="PC2" type4="PC2" periodicity1="2" phase1="
        3.14159" k1="4.184"/>
1045  <Proper type1="PC7" type2="PC7" type3="PC2" type4="PH1" periodicity1="2" phase1="
        3.14159" k1="4.184"/>
1046  <Proper type1="PC5" type2="PC7" type3="PC2" type4="PH1" periodicity1="2" phase1="
        3.14159" k1="4.184"/>
1047  <Proper type1="" type2="PC3" type3="PN2" type4="" periodicity1="2" phase1="
        3.14159" k1="2.82420"/>
1048  <!-- Spiro H -->
1049  <Proper type1="" type2="PC5" type3="POH" type4="" periodicity1="2" phase1="
        3.14159" k1="9.6232"/>
1050  <Proper type1="PHO" type2="POH" type3="PC5" type4="PO1" periodicity1="2" phase1="
        3.14159" k1="9.6232" periodicity2="1" phase2="0.0" k2="7.9496"/>
1051  <!-- TBA Br -->
1052    <Proper type1="BC1" type2="BC1" type3="BN1" type4="BC1" periodicity1="3" phase1
        ="0.0" k1="0.6527040000000001"/>
1053    <Proper type1="" type2="BC1" type3="BN1" type4="" periodicity1="3" phase1="
        0.0" k1="0.6527040000000001"/>
1054    <Proper type1="" type2="BC1" type3="BC1" type4="" periodicity1="3" phase1="
        0.0" k1="0.6527040000000001"/>
1055    <Proper type1="BC1" type2="BC1" type3="BC1" type4="BC1" periodicity1="3" phase1
        ="0.0" k1="0.75312" periodicity2="2" phase2="3.141592653589793" k2="1.046"
        periodicity3="1" phase3="3.141592653589793" k3="0.8368000000000001"/>
1056    <Proper type1="BH1" type2="BC1" type3="BC1" type4="BC1" periodicity1="3" phase1
        ="0.0" k1="0.66944"/>
1057    <Proper type1="BH1" type2="BC1" type3="BC1" type4="BH1" periodicity1="3" phase1
        ="0.0" k1="0.6276"/>
1058
1059  <!-- Impropers -->
1060  <!-- IIIIIIIII -->
1061  <Improper type1="C1" type2="C3" type3="O1" type4="O1" periodicity1="2" phase1="
        3.14159" k1="41.840"/>
1062  <Improper type1="C3" type2="C2" type3="C1" type4="C1" periodicity1="2" phase1="
        3.14159" k1="41.840"/>
1063  <Improper type1="C1" type2="C3" type3="C1" type4="H1" periodicity1="2" phase1="
        3.14159" k1="1.548"/>
1064  <!-- C4 like C3 -->
```

```
1065  <Improper type1="C4" type2="C4" type3="C1" type4="C1" periodicity1="2" phase1="
        3.14159" k1="41.840"/>
1066  <Improper type1="C4" type2="C4" type3="C1" type4="C4" periodicity1="2" phase1="
        3.14159" k1="41.840"/>
1067  <Improper type1="C4" type2="C4" type3="C4" type4="C4" periodicity1="2" phase1="
        3.14159" k1="41.840"/>
1068  <Improper type1="C1" type2="C4" type3="C1" type4="H1" periodicity1="2" phase1="
        3.14159" k1="1.548"/>
1069  <!-- o-DFB from acpype -->
1070  <Improper type1="SC" type2="SC" type3="SC" type4="SF" periodicity1="2" phase1="
        3.14159" k1="4.6024"/>
1071  <Improper type1="SC" type2="" type3="" type4="SH" periodicity1="2" phase1="
        3.14159" k1="4.6024"/>
1072  <!-- Spiro from acpype -->
1073  <Improper type1="PC1" type2="PC1" type3="PC1" type4="PN1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1074  <Improper type1="PC1" type2="" type3="" type4="PH1" periodicity1="2" phase1="
        3.14159" k1="4.6024"/>
1075  <Improper type1="PC1" type2="PC1" type3="PC1" type4="PC2" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1076  <Improper type1="PC1" type2="PC1" type3="PC1" type4="PO2" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1077  <Improper type1="PC2" type2="PC3" type3="PC1" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1078  <Improper type1="PC3" type2="PC4" type3="PC2" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1079  <Improper type1="PN2" type2="PC4" type3="PC4" type4="PC1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1080  <Improper type1="PC1" type2="PC4" type3="PC1" type4="PC1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1081  <Improper type1="PC1" type2="PC1" type3="PC1" type4="PN2" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1082  <Improper type1="PC5" type2="" type3="PO1" type4="PO1" periodicity1="2" phase1="
        3.14159" k1="4.6024"/>
1083  <!-- Spiro open -->
1084  <Improper type1="PC6" type2="PC6" type3="PC7" type4="PN1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1085  <Improper type1="PC7" type2="PC6" type3="PC7" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1086  <Improper type1="PC6" type2="PC6" type3="PC7" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1087  <Improper type1="PC7" type2="PC5" type3="PC6" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1088  <Improper type1="PC7" type2="PC5" type3="PC7" type4="PC2" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1089  <Improper type1="PC5" type2="" type3="" type4="PO1" periodicity1="2" phase1="
        3.14159" k1="43.932"/>
1090  <Improper type1="PC2" type2="PC7" type3="PC2" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1091  <Improper type1="PC2" type2="PC3" type3="PC2" type4="PH1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1092  <Improper type1="PC3" type2="PC4" type3="PC2" type4="PN2" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1093  <Improper type1="PN2" type2="PC3" type3="PC4" type4="PC1" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1094  <!-- Spiro H -->
1095  <Improper type1="PC5" type2="PC4" type3="PO1" type4="POH" periodicity1="2" phase1
        ="3.14159" k1="4.6024"/>
1096 </PeriodicTorsionForce>
1097
1098 <NonbondedForce coulomb14scale="0.8" lj14scale="1.0">
1099  <!-- water spce -->
1100  <Atom type="H" charge="0.424" sigma="0.0000" epsilon="0.0000"/>
1101  <Atom type="O" charge="-0.848" sigma="0.3554" epsilon="0.3182"/>
1102 </NonbondedForce>
1103
```

```
1104   <!-- from acpype -->
1105  <NonbondedForce coulomb14scale="0.8" lj14scale="1.0">
1106   <UseAttributeFromResidue name="charge"/>
1107   <Atom type="C2" sigma="0.34310" epsilon="0.4393"/>
1108   <Atom type="C3" sigma="0.34310" epsilon="0.4393"/>
1109   <Atom type="C1" sigma="0.34310" epsilon="0.4393"/>
1110   <Atom type="H1" sigma="0.25710" epsilon="0.1841"/>
1111   <!-- C4 approx C1, O2 like O3, O4 and H2 charge from acepype -->
1112   <Atom type="C4" sigma="0.34310" epsilon="0.4393"/>
1113   <Atom type="H2" sigma="0.25710" epsilon="0.1841"/>
1114   <!-- Double sigma or core collapse -->
1115   <!-- <Atom type="Zr" charge="1.968" sigma="0.27830" epsilon="0.2887"/> -->
1116   <!-- <Atom type="O1" charge="-0.586" sigma="0.31180" epsilon="0.2510"/> -->
1117   <!-- <Atom type="O2" charge="-0.992" sigma="0.31180" epsilon="0.2510"/> -->
1118   <!-- <Atom type="O3" charge="-0.992" sigma="0.31180" epsilon="0.2510"/> -->
1119   <!-- <Atom type="O4" charge="-0.607" sigma="0.31180" epsilon="0.2510"/> -->
1120   <Atom type="Zr" sigma="0.5566" epsilon="0.2887"/>
1121   <Atom type="O1" sigma="0.6236" epsilon="0.2510"/>
1122   <Atom type="O2" sigma="0.6236" epsilon="0.2510"/>
1123   <Atom type="O3" sigma="0.6236" epsilon="0.2510"/>
1124   <Atom type="O4" sigma="0.6236" epsilon="0.2510"/>
1125   <!-- o-DFB -->
1126   <Atom type="SF" sigma="0.311814551349" epsilon="0.25522"/>
1127   <Atom type="SC" sigma="0.339966950842" epsilon="0.35982"/>
1128   <Atom type="SH" sigma="0.259964245953" epsilon="0.06276"/>
1129   <!-- Spiro closed -->
1130   <Atom type="PN1" sigma="0.3249998523775958" epsilon="0.71128"/>
1131   <Atom type="PO1" sigma="0.2959921901149463" epsilon="0.87864"/>
1132   <Atom type="PC1" sigma="0.3399669508423535" epsilon="0.35982"/>
1133   <Atom type="PC2" sigma="0.3399669508423535" epsilon="0.35982"/>
1134   <Atom type="PC3" sigma="0.3399669508423535" epsilon="0.35982"/>
1135   <Atom type="PC4" sigma="0.3399669508423535" epsilon="0.45773"/>
1136   <Atom type="PO2" sigma="0.3000012343465779" epsilon="0.71128"/>
1137   <Atom type="PN2" sigma="0.3249998523775958" epsilon="0.71128"/>
1138   <Atom type="PC5" sigma="0.3399669508423535" epsilon="0.35982"/>
1139   <Atom type="PH1" sigma="0.2599642459533511" epsilon="0.06276"/>
1140   <Atom type="PH2" sigma="0.2471353044121301" epsilon="0.06569"/>
1141   <Atom type="PH3" sigma="0.2649532787749369" epsilon="0.06569"/>
1142   <!-- Spiro open -->
1143   <Atom type="PC6" sigma="0.3399669508423535" epsilon="0.359824"/>
1144   <Atom type="PC7" sigma="0.3399669508423535" epsilon="0.359824"/>
1145   <!-- Spiro H -->
1146   <Atom type="POH" sigma="0.3066473387839048" epsilon="0.8803136"/>
1147   <Atom type="PHO" sigma="1.0" epsilon="0.0"/>
1148   <!-- TBA Br -->
1149   <Atom type="BN1" sigma="0.3249998523775958" epsilon="0.7112800"/>
1150   <Atom type="BC1" sigma="0.3399669508423535" epsilon="0.4577296"/>
1151   <Atom type="BH1" sigma="0.2649532787749369" epsilon="0.0656887"/>
1152   <Atom type="BH2" sigma="0.1959977179908747" epsilon="0.0656888"/>
1153   <Atom type="tip3p_standard-Br-" sigma="0.46469277138200105" epsilon="
         0.24541419360000002"/>
1154  </NonbondedForce>
1155 </ForceField>
```

B | Appendix - Protein Data Bank Files

B.1. NU-1000

B.1.1. Node

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 N1   NOD     1      -0.243   0.581  -0.069  1.00  0.00           Zr
HETATM    2 N2   NOD     1      -0.481  -4.124  -0.086  1.00  0.00           Zr
HETATM    3 N3   NOD     1      -2.710  -1.652  -0.248  1.00  0.00           Zr
HETATM    4 N4   NOD     1       1.987  -1.891   0.091  1.00  0.00           Zr
HETATM    5 N5   NOD     1      -0.192  -1.769  -2.423  1.00  0.00           Zr
HETATM    6 N6   NOD     1      -0.531  -1.774   2.267  1.00  0.00           Zr
HETATM    7 N7   NOD     1       1.060  -3.239  -1.379  1.00  0.00            O
HETATM    8 N8   NOD     1      -1.587  -0.303  -1.570  1.00  0.00            O
HETATM    9 N9   NOD     1       1.187  -0.460  -1.352  1.00  0.00            O
HETATM   10 N10  NOD     1      -1.706  -3.081  -1.563  1.00  0.00            O
HETATM   11 N11  NOD     1      -1.766  -0.322   1.207  1.00  0.00            O
HETATM   12 N12  NOD     1       0.849  -3.223   1.395  1.00  0.00            O
HETATM   13 N13  NOD     1      -1.930  -3.099   1.212  1.00  0.00            O
HETATM   14 N14  NOD     1       0.999  -0.446   1.424  1.00  0.00            O
HETATM   15 N15  NOD     1      -2.338   0.278   1.749  1.00  0.00            H
HETATM   16 N16  NOD     1      -2.251  -3.624  -2.186  1.00  0.00            H
HETATM   17 N17  NOD     1       1.827   0.076  -1.885  1.00  0.00            H
HETATM   18 N18  NOD     1       1.348  -3.819   2.008  1.00  0.00            H
CONECT    1   14   11    8    9
CONECT    2    7   10   13   12
CONECT    3   10   11   13    8
CONECT    4    7   12   14    9
CONECT    5    8    7    9   10
CONECT    6   13   11   12   14
CONECT    7    2    4    5
CONECT    8    1    3    5
CONECT    9    1   17    4    5
CONECT   10    2   16    3    5
CONECT   11    1    6    3   15
CONECT   12    2    6   18    4
CONECT   13    6    2    3
CONECT   14    1    4    6
CONECT   15   11
CONECT   16   10
CONECT   17    9
CONECT   18   12
MASTER        0    0    0    0    0    0    0    0   18    0   18    0
END
```

B.1.2. Linker

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 L1   LNK     1      -2.805   1.034   1.552  1.00  0.00           C
HETATM    2 L2   LNK     1      -1.585   1.596   1.913  1.00  0.00           C
HETATM    3 L3   LNK     1      -1.840  -1.199   1.811  1.00  0.00           C
HETATM    4 L4   LNK     1      -0.463  -7.755   1.705  1.00  0.00           C
HETATM    5 L5   LNK     1      -0.870  -3.440   2.022  1.00  0.00           C
HETATM    6 L6   LNK     1       0.331  -2.809   2.363  1.00  0.00           C
HETATM    7 L7   LNK     1      -0.602  -0.610   2.158  1.00  0.00           C
HETATM    8 L8   LNK     1      -5.254  -2.815   0.874  1.00  0.00           H
HETATM    9 L9   LNK     1      -2.939  -0.373   1.489  1.00  0.00           C
HETATM   10 L10  LNK     1      -5.278  -0.134   0.833  1.00  0.00           C
HETATM   11 L11  LNK     1      -5.082   1.244   0.871  1.00  0.00           C
HETATM   12 L12  LNK     1      -0.490   0.784   2.185  1.00  0.00           C
HETATM   13 L13  LNK     1      -1.464   2.673   1.970  1.00  0.00           H
HETATM   14 L14  LNK     1      -4.176  -0.958   1.139  1.00  0.00           C
HETATM   15 L15  LNK     1      -1.981  -2.604   1.753  1.00  0.00           C
HETATM   16 L16  LNK     1       0.520  -1.427   2.439  1.00  0.00           C
HETATM   17 L17  LNK     1      -4.307  -2.346   1.119  1.00  0.00           C
HETATM   18 L18  LNK     1      -3.907   1.867   1.263  1.00  0.00           C
```

```
HETATM   19 L19  LNK     1       1.924  -0.947   2.744  1.00  0.00           C
HETATM   20 L20  LNK     1      -3.215  -3.158   1.396  1.00  0.00           C
HETATM   21 L21  LNK     1      -1.539  -7.209   2.416  1.00  0.00           C
HETATM   22 L22  LNK     1      -3.353  -4.231   1.325  1.00  0.00           H
HETATM   23 L23  LNK     1       1.183  -3.446   2.588  1.00  0.00           H
HETATM   24 L24  LNK     1       4.324  -1.206   2.373  1.00  0.00           C
HETATM   25 L25  LNK     1      -5.921   1.889   0.613  1.00  0.00           H
HETATM   26 L26  LNK     1       0.451   1.270   2.413  1.00  0.00           H
HETATM   27 L27  LNK     1      -6.701  -0.546   0.607  1.00  0.00           C
HETATM   28 L28  LNK     1       0.963  -4.860   0.646  1.00  0.00           H
HETATM   29 L29  LNK     1      -8.834  -2.155  -1.515  1.00  0.00           H
HETATM   30 L30  LNK     1       2.830  -2.285   1.293  1.00  0.00           H
HETATM   31 L31  LNK     1       0.814  -9.650   0.999  1.00  0.00           O
HETATM   32 L32  LNK     1       1.420   0.452   4.330  1.00  0.00           H
HETATM   33 L33  LNK     1      -4.699   6.102   1.357  1.00  0.00           C
HETATM   34 L34  LNK     1      -6.274   5.543   2.732  1.00  0.00           H
HETATM   35 L35  LNK     1      -2.537  -5.442   3.118  1.00  0.00           H
HETATM   36 L36  LNK     1       1.268  -7.260   0.507  1.00  0.00           H
HETATM   37 L37  LNK     1      -2.240  -7.868   2.923  1.00  0.00           H
HETATM   38 L38  LNK     1       3.007  -1.540   2.067  1.00  0.00           C
HETATM   39 L39  LNK     1      -7.139  -1.418  -0.409  1.00  0.00           C
HETATM   40 L40  LNK     1       2.220  -0.009   3.750  1.00  0.00           C
HETATM   41 L41  LNK     1      -7.361   0.870   2.121  1.00  0.00           H
HETATM   42 L42  LNK     1      -8.502  -1.500  -0.712  1.00  0.00           C
HETATM   43 L43  LNK     1       0.243  -5.501   1.147  1.00  0.00           C
HETATM   44 L44  LNK     1       3.544   0.333   4.039  1.00  0.00           C
HETATM   45 L45  LNK     1      -9.427  -0.688  -0.046  1.00  0.00           C
HETATM   46 L46  LNK     1       4.619  -0.263   3.367  1.00  0.00           C
HETATM   47 L47  LNK     1      -6.427  -1.986  -1.003  1.00  0.00           H
HETATM   48 L48  LNK     1      -7.655   0.180   1.334  1.00  0.00           C
HETATM   49 L49  LNK     1      -8.985   0.134   0.990  1.00  0.00           C
HETATM   50 L50  LNK     1       6.989  -0.537   3.069  1.00  0.00           O
HETATM   51 L51  LNK     1       6.327   0.907   4.582  1.00  0.00           O
HETATM   52 L52  LNK     1      -6.245   7.854   1.839  1.00  0.00           O
HETATM   53 L53  LNK     1      -3.262   4.262   0.649  1.00  0.00           C
HETATM   54 L54  LNK     1      -0.822  -4.944   1.876  1.00  0.00           C
HETATM   55 L55  LNK     1      -5.700   3.187   2.775  1.00  0.00           H
HETATM   56 L56  LNK     1      -4.012   3.359   1.419  1.00  0.00           C
HETATM   57 L57  LNK     1       0.418  -6.880   1.067  1.00  0.00           C
HETATM   58 L58  LNK     1      -3.031   6.297   0.003  1.00  0.00           H
HETATM   59 L59  LNK     1      -5.079   3.859   2.188  1.00  0.00           C
HETATM   60 L60  LNK     1      -2.459   3.908   0.007  1.00  0.00           H
HETATM   61 L61  LNK     1       5.128  -1.699   1.831  1.00  0.00           H
HETATM   62 L62  LNK     1      -5.415   5.208   2.155  1.00  0.00           C
HETATM   63 L63  LNK     1      -1.716  -5.824   2.517  1.00  0.00           C
HETATM   64 L64  LNK     1      -9.663   0.783   1.539  1.00  0.00           H
HETATM   65 L65  LNK     1      -3.600   5.620   0.635  1.00  0.00           C
HETATM   66 L66  LNK     1       3.739   1.064   4.819  1.00  0.00           H
HETATM   67 L67  LNK     1      -5.130   7.542   1.226  1.00  0.00           C
HETATM   68 L68  LNK     1       6.043   0.095   3.724  1.00  0.00           C
HETATM   69 L69  LNK     1     -11.392  -1.367  -1.259  1.00  0.00           O
HETATM   70 L70  LNK     1      -0.252  -9.250   1.651  1.00  0.00           C
HETATM   71 L71  LNK     1      -0.986 -10.050   2.194  1.00  0.00           O
HETATM   72 L72  LNK     1      -4.553   8.353   0.531  1.00  0.00           O
HETATM   73 L73  LNK     1     -11.549   0.235   0.254  1.00  0.00           O
HETATM   74 L74  LNK     1     -10.877  -0.660  -0.421  1.00  0.00           C
CONECT    1    2    9    9   18
CONECT    2    1   12   12   13
CONECT    3   15   15    7    9
CONECT    4   57   21   21   70
CONECT    5   15    6    6   54
CONECT    6   16    5    5   23
CONECT    7   16   16    3   12
CONECT    8   17
CONECT    9    1    1    3   14
CONECT   10   14   14   11   27
CONECT   11   18   18   10   25
CONECT   12    2    2    7   26
CONECT   13    2
CONECT   14    9   10   10   17
CONECT   15    3    3    5   20
CONECT   16    7    7    6   19
CONECT   17    8   14   20   20
CONECT   18   11   11    1   56
CONECT   19   38   40   40   16
CONECT   20   17   17   15   22
```

```
 97 CONECT   21    4    4   37   63
 98 CONECT   22   20
 99 CONECT   23    6
100 CONECT   24   46   38   38   61
101 CONECT   25   11
102 CONECT   26   12
103 CONECT   27   10   39   39   48
104 CONECT   28   43
105 CONECT   29   42
106 CONECT   30   38
107 CONECT   31   70
108 CONECT   32   40
109 CONECT   33   65   65   62   67
110 CONECT   34   62
111 CONECT   35   63
112 CONECT   36   57
113 CONECT   37   21
114 CONECT   38   19   24   24   30
115 CONECT   39   27   27   42   47
116 CONECT   40   19   19   32   44
117 CONECT   41   48
118 CONECT   42   29   39   45   45
119 CONECT   43   57   57   54   28
120 CONECT   44   40   46   46   66
121 CONECT   45   42   42   49   74
122 CONECT   46   24   44   44   68
123 CONECT   47   39
124 CONECT   48   49   49   41   27
125 CONECT   49   48   48   45   64
126 CONECT   50   68
127 CONECT   51   68   68
128 CONECT   52   67
129 CONECT   53   60   56   56   65
130 CONECT   54   43    5   63   63
131 CONECT   55   59
132 CONECT   56   53   53   59   18
133 CONECT   57   43   43    4   36
134 CONECT   58   65
135 CONECT   59   55   56   62   62
136 CONECT   60   53
137 CONECT   61   24
138 CONECT   62   59   59   33   34
139 CONECT   63   21   54   54   35
140 CONECT   64   49
141 CONECT   65   33   33   58   53
142 CONECT   66   44
143 CONECT   67   52   33   72   72
144 CONECT   68   50   51   51   46
145 CONECT   69   74   74
146 CONECT   70   31    4   71   71
147 CONECT   71   70   70
148 CONECT   72   67   67
149 CONECT   73   74
150 CONECT   74   69   69   45   73
151 MASTER        0    0    0    0    0    0    0    0   74    0   74    0
152 END
```

B.2. Spiropyran & Merocyanine

B.2.1. Spiropyran

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 P1   SPI     1      -4.990  -0.734   0.194  1.00  0.00           N
HETATM    2 P2   SPI     1      -3.863  -1.198   0.246  1.00  0.00           O
HETATM    3 P3   SPI     1      -5.210   0.465   0.148  1.00  0.00           O
HETATM    4 P4   SPI     1      -6.144  -1.672   0.137  1.00  0.00           C
HETATM    5 P5   SPI     1      -7.451  -1.183   0.017  1.00  0.00           C
HETATM    6 P6   SPI     1      -5.939  -3.057   0.189  1.00  0.00           C
HETATM    7 P7   SPI     1      -7.018  -3.933   0.103  1.00  0.00           C
HETATM    8 P8   SPI     1      -8.533  -2.060  -0.062  1.00  0.00           C
HETATM    9 P9   SPI     1      -8.312  -3.437  -0.023  1.00  0.00           C
HETATM   10 P10  SPI     1      -9.910  -1.531  -0.223  1.00  0.00           C
HETATM   11 P11  SPI     1     -10.910  -2.402  -0.269  1.00  0.00           C
HETATM   12 P12  SPI     1     -10.753  -3.912  -0.139  1.00  0.00           C
HETATM   13 P13  SPI     1      -9.356  -4.317  -0.220  1.00  0.00           O
HETATM   14 P14  SPI     1     -11.524  -4.554  -1.217  1.00  0.00           N
HETATM   15 P15  SPI     1     -11.408  -4.493   1.150  1.00  0.00           C
HETATM   16 P16  SPI     1     -12.712  -4.971   0.570  1.00  0.00           C
HETATM   17 P17  SPI     1     -12.662  -5.230  -0.849  1.00  0.00           C
HETATM   18 P18  SPI     1     -11.021  -4.702  -2.603  1.00  0.00           C
HETATM   19 P19  SPI     1     -10.076  -3.589  -3.119  1.00  0.00           C
HETATM   20 P20  SPI     1      -9.829  -3.688  -4.610  1.00  0.00           C
HETATM   21 P21  SPI     1     -10.652  -5.721   1.730  1.00  0.00           C
HETATM   22 P22  SPI     1     -11.668  -3.450   2.269  1.00  0.00           C
HETATM   23 P23  SPI     1     -13.651  -5.724   1.367  1.00  0.00           C
HETATM   24 P24  SPI     1     -14.455  -6.697   0.779  1.00  0.00           C
HETATM   25 P25  SPI     1     -14.404  -6.937  -0.593  1.00  0.00           C
HETATM   26 P26  SPI     1     -13.561  -6.167  -1.472  1.00  0.00           C
HETATM   27 P27  SPI     1      -8.826  -2.940  -4.990  1.00  0.00           O
HETATM   28 P28  SPI     1     -10.652  -4.104  -5.403  1.00  0.00           O
HETATM   29 P29  SPI     1      -7.633  -0.113  -0.033  1.00  0.00           H
HETATM   30 P30  SPI     1      -4.936  -3.464   0.279  1.00  0.00           H
HETATM   31 P31  SPI     1      -6.848  -5.008   0.116  1.00  0.00           H
HETATM   32 P32  SPI     1     -10.103  -0.465  -0.305  1.00  0.00           H
HETATM   33 P33  SPI     1     -11.928  -2.038  -0.395  1.00  0.00           H
HETATM   34 P34  SPI     1     -10.516  -5.667  -2.697  1.00  0.00           H
HETATM   35 P35  SPI     1     -11.893  -4.719  -3.262  1.00  0.00           H
HETATM   36 P36  SPI     1     -10.495  -2.603  -2.910  1.00  0.00           H
HETATM   37 P37  SPI     1      -9.111  -3.658  -2.615  1.00  0.00           H
HETATM   38 P38  SPI     1     -10.507  -6.489   0.966  1.00  0.00           H
HETATM   39 P39  SPI     1      -9.672  -5.427   2.113  1.00  0.00           H
HETATM   40 P40  SPI     1     -11.206  -6.169   2.558  1.00  0.00           H
HETATM   41 P41  SPI     1     -10.733  -2.972   2.570  1.00  0.00           H
HETATM   42 P42  SPI     1     -12.364  -2.674   1.941  1.00  0.00           H
HETATM   43 P43  SPI     1     -12.100  -3.925   3.152  1.00  0.00           H
HETATM   44 P44  SPI     1     -13.726  -5.550   2.437  1.00  0.00           H
HETATM   45 P45  SPI     1     -15.132  -7.279   1.399  1.00  0.00           H
HETATM   46 P46  SPI     1     -15.031  -7.707  -1.036  1.00  0.00           H
HETATM   47 P47  SPI     1     -13.595  -6.299  -2.548  1.00  0.00           H
CONECT    1    2    2    3    3
CONECT    1    4
CONECT    2    1    1
CONECT    3    1    1
CONECT    4    1    5    5    6
CONECT    5    4    4    8   29
CONECT    6    4    7    7   30
CONECT    7    6    6    9   31
CONECT    8    5    9    9   10
CONECT    9    8    8    7   13
CONECT   10    8   11   11   32
CONECT   11   10   10   12   33
CONECT   12   11   13   14   15
CONECT   13    9   12
CONECT   14   12   17   18
CONECT   15   12   16   21   22
CONECT   16   15   17   23   23
CONECT   17   16   14   26   26
CONECT   18   14   19   34   35
CONECT   19   18   20   36   37
CONECT   20   19   27   28   28
CONECT   21   15   38   39   40
```

```
CONECT   22   15   41   42   43
CONECT   23   16   16   24   44
CONECT   24   23   25   25   45
CONECT   25   24   24   26   46
CONECT   26   25   17   17   47
CONECT   27   20
CONECT   28   20   20
CONECT   29    5
CONECT   30    6
CONECT   31    7
CONECT   32   10
CONECT   33   11
CONECT   34   18
CONECT   35   18
CONECT   36   19
CONECT   37   19
CONECT   38   21
CONECT   39   21
CONECT   40   21
CONECT   41   22
CONECT   42   22
CONECT   43   22
CONECT   44   23
CONECT   45   24
CONECT   46   25
CONECT   47   26
MASTER        0    0    0    0    0    0    0    0   47    0   47    0
END
```

B.2.2. Merocyanine

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 M1   MER     1      -9.082   2.251   3.480  1.00  0.00           N
HETATM    2 M2   MER     1      -8.291   3.152   3.710  1.00  0.00           O
HETATM    3 M3   MER     1      -9.829   1.795   4.329  1.00  0.00           O
HETATM    4 M4   MER     1      -9.142   1.692   2.102  1.00  0.00           C
HETATM    5 M5   MER     1      -9.947   0.580   1.818  1.00  0.00           C
HETATM    6 M6   MER     1      -8.397   2.270   1.068  1.00  0.00           C
HETATM    7 M7   MER     1      -8.457   1.748  -0.223  1.00  0.00           C
HETATM    8 M8   MER     1     -10.009   0.054   0.525  1.00  0.00           C
HETATM    9 M9   MER     1      -9.261   0.645  -0.497  1.00  0.00           C
HETATM   10 M10  MER     1     -10.869  -1.116   0.258  1.00  0.00           C
HETATM   11 M11  MER     1     -10.487  -2.360  -0.063  1.00  0.00           C
HETATM   12 M12  MER     1     -11.368  -3.500  -0.284  1.00  0.00           C
HETATM   13 M13  MER     1      -9.305   0.150  -1.768  1.00  0.00           O
HETATM   14 M14  MER     1     -11.951  -4.341  -1.210  1.00  0.00           N
HETATM   15 M15  MER     1     -11.704  -4.272   0.976  1.00  0.00           C
HETATM   16 M16  MER     1     -12.862  -5.084   0.472  1.00  0.00           C
HETATM   17 M17  MER     1     -12.908  -5.286  -0.945  1.00  0.00           C
HETATM   18 M18  MER     1     -11.546  -4.359  -2.632  1.00  0.00           C
HETATM   19 M19  MER     1     -10.486  -3.322  -3.070  1.00  0.00           C
HETATM   20 M20  MER     1     -10.138  -3.463  -4.542  1.00  0.00           C
HETATM   21 M21  MER     1     -10.551  -5.215   1.397  1.00  0.00           C
HETATM   22 M22  MER     1     -12.168  -3.384   2.159  1.00  0.00           C
HETATM   23 M23  MER     1     -13.599  -5.975   1.336  1.00  0.00           C
HETATM   24 M24  MER     1     -14.404  -6.968   0.784  1.00  0.00           C
HETATM   25 M25  MER     1     -14.486  -7.140  -0.596  1.00  0.00           C
HETATM   26 M26  MER     1     -13.733  -6.322  -1.514  1.00  0.00           C
HETATM   27 M27  MER     1      -9.043  -2.827  -4.867  1.00  0.00           O
HETATM   28 M28  MER     1     -10.729  -4.182  -5.324  1.00  0.00           O
HETATM   29 M29  MER     1     -10.533   0.113   2.606  1.00  0.00           H
HETATM   30 M30  MER     1      -7.765   3.134   1.258  1.00  0.00           H
HETATM   31 M31  MER     1      -7.874   2.207  -1.017  1.00  0.00           H
HETATM   32 M32  MER     1     -11.937  -0.925   0.361  1.00  0.00           H
HETATM   33 M33  MER     1      -9.422  -2.566  -0.165  1.00  0.00           H
HETATM   34 M34  MER     1     -11.148  -5.354  -2.857  1.00  0.00           H
HETATM   35 M35  MER     1     -12.437  -4.215  -3.249  1.00  0.00           H
HETATM   36 M36  MER     1     -10.856  -2.309  -2.896  1.00  0.00           H
HETATM   37 M37  MER     1      -9.574  -3.453  -2.482  1.00  0.00           H
HETATM   38 M38  MER     1     -10.270  -5.877   0.573  1.00  0.00           H
HETATM   39 M39  MER     1      -9.669  -4.638   1.683  1.00  0.00           H
HETATM   40 M40  MER     1     -10.843  -5.836   2.247  1.00  0.00           H
HETATM   41 M41  MER     1     -11.357  -2.740   2.507  1.00  0.00           H
```

```
HETATM   42 M42  MER      1      -13.008   -2.751    1.863  1.00  0.00           H
HETATM   43 M43  MER      1      -12.481   -3.990    3.011  1.00  0.00           H
HETATM   44 M44  MER      1      -13.561   -5.878    2.417  1.00  0.00           H
HETATM   45 M45  MER      1      -14.978   -7.617    1.440  1.00  0.00           H
HETATM   46 M46  MER      1      -15.122   -7.918   -1.011  1.00  0.00           H
HETATM   47 M47  MER      1      -13.771   -6.492   -2.585  1.00  0.00           H
CONECT    1    2    2    3    3
CONECT    1    4
CONECT    2    1    1
CONECT    3    1    1
CONECT    4    1    5    5    6
CONECT    5    4    4    8   29
CONECT    6    4    7    7   30
CONECT    7    6    6    9   31
CONECT    8    5    9    9   10
CONECT    9    8    8    7   13
CONECT   10    8   11   11   32
CONECT   11   10   10   12   33
CONECT   12   11   14   14   15
CONECT   13    9
CONECT   14   12   12   17   18
CONECT   15   12   16   21   22
CONECT   16   15   17   23   23
CONECT   17   16   14   26   26
CONECT   18   14   19   34   35
CONECT   19   18   20   36   37
CONECT   20   19   27   28   28
CONECT   21   15   38   39   40
CONECT   22   15   41   42   43
CONECT   23   16   16   24   44
CONECT   24   23   25   25   45
CONECT   25   24   24   26   46
CONECT   26   47   25   17   17
CONECT   27   20
CONECT   28   20   20
CONECT   29    5
CONECT   30    6
CONECT   31    7
CONECT   32   10
CONECT   33   11
CONECT   34   18
CONECT   35   18
CONECT   36   19
CONECT   37   19
CONECT   38   21
CONECT   39   21
CONECT   40   21
CONECT   41   22
CONECT   42   22
CONECT   43   22
CONECT   44   23
CONECT   45   24
CONECT   46   25
CONECT   47   26
MASTER        0    0    0    0    0    0    0    0   47    0   47    0
END
```

B.3. Solvent and Ions

B.3.1. o-Difluorobenzene

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 S1   DFB     1       -1.735   1.925   0.000  1.00  0.00           F
HETATM    2 S2   DFB     1       -1.789   0.580  -0.000  1.00  0.00           C
HETATM    3 S3   DFB     1       -3.030  -0.055  -0.000  1.00  0.00           C
HETATM    4 S4   DFB     1       -0.617  -0.177  -0.000  1.00  0.00           C
HETATM    5 S5   DFB     1       -3.099  -1.448  -0.000  1.00  0.00           C
HETATM    6 S6   DFB     1       -1.927  -2.205  -0.000  1.00  0.00           C
HETATM    7 S7   DFB     1       -0.685  -1.570  -0.000  1.00  0.00           C
HETATM    8 S8   DFB     1        0.228  -2.159  -0.000  1.00  0.00           H
HETATM    9 S9   DFB     1       -3.944   0.534  -0.000  1.00  0.00           H
HETATM   10 S10  DFB     1        0.587   0.425  -0.000  1.00  0.00           F
HETATM   11 S11  DFB     1       -4.066  -1.945   0.000  1.00  0.00           H
HETATM   12 S12  DFB     1       -1.981  -3.291  -0.000  1.00  0.00           H
CONECT    1    2
CONECT    2    1    3    4
CONECT    3    2    5    9
CONECT    4    2    7   10
CONECT    5    3    6   11
CONECT    6    5    7   12
CONECT    7    6    4    8
CONECT    8    7
CONECT    9    3
CONECT   10    4
CONECT   11    5
CONECT   12    6
MASTER        0    0    0    0    0    0    0    0   12    0   12    0
END
```

B.3.2. Bromine

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 BR   BRO     1       -1.115   1.111  -0.000  1.00  0.00          Br
MASTER        0    0    0    0    0    0    0    0    1    0    1    0
END
```

B.3.3. Tetrabutylammonium

```
COMPND    UNNAMED
AUTHOR    GENERATED BY OPEN BABEL 3.1.1
HETATM    1 B1   TBA     1       -0.470  -0.118   0.201  1.00  0.00           N
HETATM    2 B2   TBA     1       -1.155   0.645  -0.937  1.00  0.00           C
HETATM    3 B3   TBA     1       -1.517  -0.869   1.034  1.00  0.00           C
HETATM    4 B4   TBA     1        0.254   0.846   1.151  1.00  0.00           C
HETATM    5 B5   TBA     1        0.486  -1.175  -0.370  1.00  0.00           C
HETATM    6 B6   TBA     1       -2.321   1.596  -0.559  1.00  0.00           C
HETATM    7 B7   TBA     1       -4.584   3.604  -2.296  1.00  0.00           H
HETATM    8 B8   TBA     1       -4.911   2.542  -0.919  1.00  0.00           H
HETATM    9 B9   TBA     1       -3.839   3.940  -0.724  1.00  0.00           H
HETATM   10 B10  TBA     1       -0.394   1.211  -1.480  1.00  0.00           H
HETATM   11 B11  TBA     1       -1.538  -0.115  -1.621  1.00  0.00           H
HETATM   12 B12  TBA     1        1.046   2.054   0.581  1.00  0.00           C
HETATM   13 B13  TBA     1        3.407   4.279   1.873  1.00  0.00           H
HETATM   14 B14  TBA     1        3.496   3.402   0.338  1.00  0.00           H
HETATM   15 B15  TBA     1        2.206   4.589   0.608  1.00  0.00           H
HETATM   16 B16  TBA     1       -0.493   1.246   1.843  1.00  0.00           H
HETATM   17 B17  TBA     1        0.926   0.196   1.720  1.00  0.00           H
HETATM   18 B18  TBA     1        1.856  -0.778  -0.965  1.00  0.00           C
HETATM   19 B19  TBA     1        3.565  -3.846  -0.894  1.00  0.00           H
HETATM   20 B20  TBA     1        3.435  -2.630   0.385  1.00  0.00           H
HETATM   21 B21  TBA     1        2.006  -3.552  -0.122  1.00  0.00           H
HETATM   22 B22  TBA     1        0.668  -1.822   0.492  1.00  0.00           H
HETATM   23 B23  TBA     1       -0.067  -1.742  -1.123  1.00  0.00           H
HETATM   24 B24  TBA     1       -2.375  -1.955   0.332  1.00  0.00           C
HETATM   25 B25  TBA     1       -5.436  -1.977   2.039  1.00  0.00           H
```

```
28 HETATM   26 B26  TBA     1      -4.319  -0.611   1.937  1.00  0.00           H
29 HETATM   27 B27  TBA     1      -5.078  -1.250   0.467  1.00  0.00           H
30 HETATM   28 B28  TBA     1      -0.955  -1.345   1.844  1.00  0.00           H
31 HETATM   29 B29  TBA     1      -2.171  -0.127   1.494  1.00  0.00           H
32 HETATM   30 B30  TBA     1      -2.935   2.271  -1.801  1.00  0.00           C
33 HETATM   31 B31  TBA     1      -3.112   1.036  -0.057  1.00  0.00           H
34 HETATM   32 B32  TBA     1      -1.972   2.363   0.136  1.00  0.00           H
35 HETATM   33 B33  TBA     1       1.918   2.706   1.671  1.00  0.00           C
36 HETATM   34 B34  TBA     1       1.686   1.753  -0.246  1.00  0.00           H
37 HETATM   35 B35  TBA     1       0.352   2.800   0.186  1.00  0.00           H
38 HETATM   36 B36  TBA     1       2.613  -1.998  -1.539  1.00  0.00           C
39 HETATM   37 B37  TBA     1       1.713  -0.046  -1.762  1.00  0.00           H
40 HETATM   38 B38  TBA     1       2.480  -0.323  -0.192  1.00  0.00           H
41 HETATM   39 B39  TBA     1      -3.482  -2.514   1.251  1.00  0.00           C
42 HETATM   40 B40  TBA     1      -2.836  -1.560  -0.576  1.00  0.00           H
43 HETATM   41 B41  TBA     1      -1.731  -2.784   0.031  1.00  0.00           H
44 HETATM   42 B42  TBA     1       2.811   3.815   1.084  1.00  0.00           C
45 HETATM   43 B43  TBA     1       1.279   3.127   2.451  1.00  0.00           H
46 HETATM   44 B44  TBA     1       2.554   1.949   2.137  1.00  0.00           H
47 HETATM   45 B45  TBA     1       2.922  -3.070  -0.473  1.00  0.00           C
48 HETATM   46 B46  TBA     1       2.033  -2.446  -2.350  1.00  0.00           H
49 HETATM   47 B47  TBA     1       3.556  -1.652  -1.969  1.00  0.00           H
50 HETATM   48 B48  TBA     1      -4.648  -1.522   1.434  1.00  0.00           C
51 HETATM   49 B49  TBA     1      -3.875  -3.431   0.806  1.00  0.00           H
52 HETATM   50 B50  TBA     1      -3.062  -2.781   2.224  1.00  0.00           H
53 HETATM   51 B51  TBA     1      -4.143   3.144  -1.408  1.00  0.00           C
54 HETATM   52 B52  TBA     1      -2.184   2.888  -2.299  1.00  0.00           H
55 HETATM   53 B53  TBA     1      -3.262   1.507  -2.510  1.00  0.00           H
56 CONECT    1    2    3    4    5
57 CONECT    2    1    6   10   11
58 CONECT    3    1   24   28   29
59 CONECT    4    1   12   16   17
60 CONECT    5    1   18   22   23
61 CONECT    6    2   30   31   32
62 CONECT    7   51
63 CONECT    8   51
64 CONECT    9   51
65 CONECT   10    2
66 CONECT   11    2
67 CONECT   12    4   33   34   35
68 CONECT   13   42
69 CONECT   14   42
70 CONECT   15   42
71 CONECT   16    4
72 CONECT   17    4
73 CONECT   18    5   36   37   38
74 CONECT   19   45
75 CONECT   20   45
76 CONECT   21   45
77 CONECT   22    5
78 CONECT   23    5
79 CONECT   24    3   39   40   41
80 CONECT   25   48
81 CONECT   26   48
82 CONECT   27   48
83 CONECT   28    3
84 CONECT   29    3
85 CONECT   30    6   51   52   53
86 CONECT   31    6
87 CONECT   32    6
88 CONECT   33   12   42   43   44
89 CONECT   34   12
90 CONECT   35   12
91 CONECT   36   18   45   46   47
92 CONECT   37   18
93 CONECT   38   18
94 CONECT   39   24   48   49   50
95 CONECT   40   24
96 CONECT   41   24
97 CONECT   42   13   14   15   33
98 CONECT   43   33
99 CONECT   44   33
100 CONECT   45   19   20   21   36
101 CONECT   46   36
102 CONECT   47   36
103 CONECT   48   25   26   27   39
```

```
104 CONECT   49   39
105 CONECT   50   39
106 CONECT   51    7    8    9   30
107 CONECT   52   30
108 CONECT   53   30
109 MASTER        0    0    0    0    0    0    0    0   53    0   53    0
110 END
```

C | Appendix - Python Scripts

C.1. Modeling

order2pdb.py

```
import argparse
import sys
import os.path
import numpy as np

def parse_args():
    # create command line argument parser object
    parser = argparse.ArgumentParser(description = 'Fixes pdb atom and residue names')

    # define command line arguments
    parser.add_argument('-d', action = 'count',
                        help = '<optional> Debug Mode: extra info, print steps to check stuff',
                        default = 0)
    parser.add_argument('-o', '--outfile', action = 'store',
                        help = 'Specify file name for the output',
                        required = False, type = str, default = '_fixed.pdb')

    parser.add_argument('-i', '--infile', action = 'store',
                        help = 'Specify file name for the intput',
                        required = False, type = str, default = 'molecule.pdb')

    # parse command line
    args = parser.parse_args()

    # return parsed command line arguments
    return args


if __name__ == "__main__":

    #set variables
    args = parse_args()
    debug = args.d
    outfile = args.outfile
    infile = args.infile

    if outfile == '_fixed.pdb':
        mol_name = infile.split('.')[0]
        outfile = mol_name + '_fixed.pdb'

#If line starts with HETATM split -1 = first atom

    line_list = []
    lttr_list = [' ','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']
    skip_count = 0
    atom_nmb = 0
    res_nmb = 0
    lttr_nmb = 1 #should be 1 if more than 9999 residues to start at chain A -> else chain letter is left empty

    with open(infile, 'r') as pdb_file, open(outfile, 'w') as fix_file:
        for line in pdb_file:
            if line.startswith("HETATM"):
                if skip_count == 0:
                    res_nmb += 1
                    if res_nmb == 10000:
                        res_nmb = 1
                        lttr_nmb += 1
                    #line_list = line.split()
                    lineA = line[:6]
                    lineB = line[6:].split()
                    line_list = [lineA]
                    line_list.extend(lineB)

```

```
64                      line_list[1] = int(line_list[1])
65                      line_list[4] = int(line_list[4])
66                      amount = range(5,10)
67                      for i in amount:
68                          line_list[i] = float(line_list[i])
69
70                      first_atom = line_list[-1]
71
72                      if first_atom == 'Zr':
73                          skip_count = 18 #34 with 8 OH across plane
74                          atom_nmb = 1
75                          mode = 'Node'
76                          fix_file.write('HETATM{:5d} N{:<2d}  NOD {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
    ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
    [10]))
77                          skip_count -= 1
78
79                      elif first_atom == 'C':
80                          skip_count = 74
81                          atom_nmb = 1
82                          mode = 'Link'
83                          fix_file.write('HETATM{:5d} L{:<2d}  LNK {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
    ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
    [10]))
84                          skip_count -= 1
85
86                      #elif first_atom == 'O':
87                      #    skip_count = 3
88                      #    atom_nmb = 1
89                      #    mode = 'Water'
90                      #    fix_file.write('HETATM{:5d} WO{}  HOH {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
    ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
    [10]))
91                      #    skip_count -= 1
92
93                      elif first_atom == 'O':
94                          skip_count = 2
95                          atom_nmb = 1
96                          mode = 'Hydroxy1'
97                          fix_file.write('HETATM{:5d} TO1  TOH {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],lttr_list[lttr_nmb],res_nmb,
    line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list[10]))
98                          skip_count -= 1
99
100                     elif first_atom == 'H':
101                         skip_count = 2
102                         atom_nmb = 1
103                         mode = 'Hydroxy2'
104                         fix_file.write('HETATM{:5d} TH1  TOH {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],lttr_list[lttr_nmb],res_nmb,
    line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list[10]))
105                         skip_count -= 1
106
107                     elif first_atom == 'F':
108                         skip_count = 12
109                         atom_nmb = 1
110                         mode = 'Difluorobenzene'
111                         fix_file.write('HETATM{:5d} S{:<2d}  DFB {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
    ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
    [10]))
112                         skip_count -= 1
113
114                     elif first_atom == 'N':
115                         skip_count = 47
116                         atom_nmb = 1
117                         mode = 'Spiropyrane'
118                         fix_file.write('HETATM{:5d} P{:<2d}  SPI {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
    ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
    [10]))
119                         #fix_file.write('HETATM{:5d} M{:<2d}  MER {}{:4d}{:>12.3f}{:>8.3f
    }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
```

```
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
120                         skip_count -= 1
121
122                     else:
123                         print('Error: Residue not found')
124                 else:
125                     #line_list = line.split()
126                     lineA = line[:6]
127                     lineB = line[6:].split()
128                     line_list = [lineA]
129                     line_list.extend(lineB)
130
131                     line_list[1] = int(line_list[1])
132                     line_list[4] = int(line_list[4])
133                     amount = range(5,10)
134                     for i in amount:
135                         line_list[i] = float(line_list[i])
136
137                     skip_count -= 1
138                     atom_nmb += 1
139                     if mode == 'Node':
140                         fix_file.write('HETATM{:5d} N{:<2d}  NOD {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
141
142                     if mode == 'Link':
143                         fix_file.write('HETATM{:5d} L{:<2d}  LNK {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
144                     if mode == 'Water':
145                         wtr_nmb = atom_nmb - 1
146                         fix_file.write('HETATM{:5d} WH{}  HOH {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],wtr_nmb,lttr_list[lttr_nmb],
        res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
147                     if mode == 'Hydroxy1':
148                         fix_file.write('HETATM{:5d} TH1  TOH {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],lttr_list[lttr_nmb],res_nmb,
        line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list[10]))
149                     if mode == 'Hydroxy2':
150                         fix_file.write('HETATM{:5d} TO1  TOH {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],lttr_list[lttr_nmb],res_nmb,
        line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list[10]))
151                     if mode == 'Difluorobenzene':
152                         fix_file.write('HETATM{:5d} S{:<2d}  DFB {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
153                     if mode == 'Spiropyrane':
154                         fix_file.write('HETATM{:5d} P{:<2d}  SPI {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
155                         #fix_file.write('HETATM{:5d} M{:<2d}  MER {}{:4d}{:>12.3f}{:>8.3f
        }{:>8.3f}{:>6.2f}{:>6.2f}{:>12s}  \n'.format(line_list[1],atom_nmb,lttr_list[lttr_nmb
        ],res_nmb,line_list[5],line_list[6],line_list[7],line_list[8],line_list[9],line_list
        [10]))
156
157             else:
158                 fix_file.write(line)
159
160
161 print(res_nmb)
```

packmol.inp

```
tolerance 2.0
seed -5746235


filetype pdb
output nxs135tb10dfb3.pdb

structure ./nu1tXspiro135.pdb
  number 1
  inside box 25. 5. 5. 75. 95. 95.
end structure

structure ./tba.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./tba.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./tba.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./bro.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./bro.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./bro.pdb
  number 1
  inside box 40. 30. 30. 60. 70. 70.
end structure

structure ./tba.pdb
  number 7
  inside box -0. -0. -0. 100. 100. 100.
end structure

structure ./bro.pdb
  number 7
  inside box -0. -0. -0. 100. 100. 100.
end structure

structure ./dfb.pdb
  number 5480
  inside box -0. -0. -0. 120. 120. 120.
end structure
```

C.2. Force Field Conversion

convertff.py

```
import parmed as pmd
molecule = 'dfb'
#FILE NAME molecule_CHARMM.xxx
rtf_file = '{}_CHARMM.rtf'.format(molecule)
prm_file = '{}_CHARMM.prm'.format(molecule)


charmm_params = pmd.charmm.CharmmParameterSet(rtf_file, prm_file)

#structure = pmd.load_file('output.pdb')

omm_params = pmd.openmm.OpenMMParameterSet.from_parameterset(charmm_params)
omm_params.write('charmm_to_openmm.xml', separate_ljforce=True )
```

pdb2xml_atomtypes.py

```
import argparse
import sys
import os.path
import numpy as np

def parse_args():
    # create command line argument parser object
    parser = argparse.ArgumentParser(description = 'Converts pdb atoms to xml atoms with
    atomtypes')

    # define command line arguments
    parser.add_argument('-d', action = 'count',
                        help = '<optional> Debug Mode: extra info, print steps to check
    stuff',
                        default = 0)
    parser.add_argument('-o', '--outfile', action = 'store',
                        help = 'Specify file name for the output',
                        required = False, type = str, default = '_atoms.txt')

    parser.add_argument('-i', '--infile', action = 'store',
                        help = 'Specify file name for the intput',
                        required = False, type = str, default = 'molecule.pdb')

    # parse command line
    args = parser.parse_args()

    # return parsed command line arguments
    return args


if __name__ == "__main__":

    #set variables
    args = parse_args()
    debug = args.d
    outfile = args.outfile
    infile = args.infile

    print('...')

    if outfile == '_atoms.txt':
        mol_name = infile.split('.')[0]
        outfile = mol_name + '_atoms.txt'

    connects=[]
    bonds=[]

    with open(infile, 'r') as pdb_file:
        for line in pdb_file:
            if line.startswith("CONECT"):
                connects = line.split()
                connects.remove('CONECT')
                connects = [eval(i) for i in connects]
```

```
52                  for i in connects:
53                      if i > connects[0]:
54                          out_str = (connects[0],i)
55                          if out_str not in bonds:
56                              bonds.append(out_str)
57      if debug > 0:
58          print('List of bonds found')
59          if debug > 1:
60              for i in bonds: print(i)
61
62      atoms=[]
63      el_dic={}
64      a_types=[]
65
66      with open(infile, 'r') as pdb_file:
67          for line in pdb_file:
68              if line.startswith("ATOM") or line.startswith("HETATM"):
69                  connects = line.split()
70                  atoms.append((int(connects[1]),connects[-1]))
71                  el_dic[int(connects[1])]=connects[-1]
72
73      if debug > 0:
74          print('List of atoms found')
75          if debug > 1:
76              for i in atoms: print(i)
77
78      print('...'+'\n'+'Matching atom types...')
79
80      #ok the fuck am i doing...
81
82      for i in atoms:
83          atom = i[0]
84          element = i[1]
85          bound_2 = []
86          for item in bonds:
87              if atom in item:
88                  if item[0] == atom:
89                      bound_2.append(item[1])
90                  elif item[1] == atom:
91                      bound_2.append(item[0])
92                  else:
93                      print('Error: How did we get here?')
94                      sys.exit()
95
96          bound_to = [el_dic[item] for item in bound_2]
97
98          count_O = 0
99          count_H = 0
100         count_C = 0
101         count_Zr = 0
102
103         for item in bound_to:
104             if item == 'O':
105                 count_O += 1
106             elif item == 'H':
107                 count_H += 1
108             elif item == 'C':
109                 count_C += 1
110             elif item == 'Zr':
111                 count_Zr += 1
112             else:
113                 print('Error: Bound to nothing?')
114                 sys.exit()
115
116         if element == 'O':
117             if count_C > 0:
118                 a_types.append((atom,'O1'))
119             elif count_Zr == 3:
120                 if count_H == 0:
121                     a_types.append((atom,'O2'))
122                 else:
123                     a_types.append((atom,'O3'))
124             elif count_Zr == 1:
125                 a_types.append((atom,'O4')) #approx like O3 in node?
126             else:
127                 print('Error: Unknown Oxygen')
```

```
128         elif element == 'H':
129             if count_C == 1:
130                 a_types.append((atom,'H1'))
131             elif count_O == 1:
132                 a_types.append((atom,'H2'))
133             else:
134                 print('Error: Unknown Hydrogen')
135         elif element == 'C':
136             if count_H == 1:
137                 a_types.append((atom,'C1'))
138             elif count_O == 2:
139                 a_types.append((atom,'C2'))
140             else:
141                 a_types.append((atom,'CX'))
142         elif element == 'Zr':
143             a_types.append((atom,'Zr'))
144         else:
145             print('Element not found'+str(i))
146             sys.exit()
147 
148     if debug > 0:
149         print(a_types)
150 
151     print('Solving carbon atoms...')
152     at_dic = {}
153     at_dic = dict(a_types)
154     a_types_final = []
155     for i in a_types:
156         if i[1] == 'CX':
157             atom = i[0]
158             bound_2 = []
159             for item in bonds:
160                 if atom in item:
161                     if item[0] == atom:
162                         bound_2.append(item[1])
163                     elif item[1] == atom:
164                         bound_2.append(item[0])
165                     else:
166                         print('Error: How did we get here?')
167                         sys.exit()
168 
169             bound_to = [at_dic[item] for item in bound_2]
170 
171             if debug > 0:
172                 print('Atom {} bound to'.format(atom)+str(bound_to))
173 
174             if 'C2' in bound_to:
175                 a_types_final.append((atom,'C3'))
176             else:
177                 a_types_final.append((atom,'C4'))
178         else:
179             a_types_final.append(i)
180 
181     if debug > 0:
182         print('Final atom types:\n' + str(a_types_final))
183 
184 
185     count_O1 = 0
186     count_O2 = 0
187     count_O3 = 0
188     count_O4 = 0
189     count_H1 = 0
190     count_H2 = 0
191     count_C1 = 0
192     count_C2 = 0
193     count_C3 = 0
194     count_C4 = 0
195     count_Zr = 0
196 
197 
198     with open(outfile, "w") as xml_file:
199         for i in a_types_final:
200             xml_file.write('   <Atom name="L{}" type="{}"/>'.format(i[0],i[1])+'\n')
201             if i[1] == 'O1':
202                 count_O1 += 1
203             elif i[1] == 'O2':
```

```
                count_O2 += 1
            elif i[1] == 'O3':
                count_O3 += 1
            elif i[1] == 'O4':
                count_O4 += 1
            elif i[1] == 'H1':
                count_H1 += 1
            elif i[1] == 'H2':
                count_H2 += 1
            elif i[1] == 'C1':
                count_C1 += 1
            elif i[1] == 'C2':
                count_C2 += 1
            elif i[1] == 'C3':
                count_C3 += 1
            elif i[1] == 'C4':
                count_C4 += 1
            elif i[1] == 'Zr':
                count_Zr += 1
            else:
                print('Atom type not found: {}'.format(i[1]))

    print('Atoms and types saved in {}'.format(outfile))
    if debug > 0:
        print('Number O1:{}'.format(count_O1))
        print('Number O2:{}'.format(count_O2))
        print('Number O3:{}'.format(count_O3))
        print('Number O4:{}'.format(count_O4))
        print('Number H1:{}'.format(count_H1))
        print('Number H2:{}'.format(count_H2))
        print('Number C1:{}'.format(count_C1))
        print('Number C2:{}'.format(count_C2))
        print('Number C3:{}'.format(count_C3))
        print('Number C4:{}'.format(count_C4))
        print('Number Zr:{}'.format(count_Zr))

sys.exit()
```

pdb2xml_bonds.py

```
import argparse
import sys
import os.path
import numpy as np

def parse_args():
    # create command line argument parser object
    parser = argparse.ArgumentParser(description = 'Converts pdb connectivities to xml
    bonds')

    # define command line arguments
    parser.add_argument('-d', action = 'count',
                        help = '<optional> Debug Mode: extra info, print steps to check
    stuff',
                        default = 0)
    parser.add_argument('-o', '--outfile', action = 'store',
                        help = 'Specify file name for the output',
                        required = False, type = str, default = '_bonds.txt')

    parser.add_argument('-i', '--infile', action = 'store',
                        help = 'Specify file name for the intput',
                        required = False, type = str, default = 'molecule.pdb')

    # parse command line
    args = parser.parse_args()

    # return parsed command line arguments
    return args


if __name__ == "__main__":

    #set variables
    args = parse_args()
```

```
33      debug = args.d
34      outfile = args.outfile
35      infile = args.infile
36
37      print('...')
38
39      if outfile == '_bonds.txt':
40          mol_name = infile.split('.')[0]
41          outfile = mol_name + '_bonds.txt'
42
43      connects=[]
44      bonds=[]
45
46      with open(infile, 'r') as pdb_file:
47          for line in pdb_file:
48              if line.startswith("CONECT"):
49                  connects = line.split()
50                  connects.remove('CONECT')
51                  connects = [eval(i) for i in connects]
52                  for i in connects:
53                      if i > connects[0]:
54                          out_str = '   <Bond atomName1="B{}" atomName2="B{}"/>'.format(
    connects[0],i)
55                          if out_str not in bonds:
56                              bonds.append(out_str)
57      if debug > 0:
58          print('List of bonds found:')
59          for i in bonds: print(i)
60
61
62      with open(outfile, "w") as xml_file:
63          for i in bonds:
64              xml_file.write(str(i)+'\n')
65
66      print('Bonds saved in {}'.format(outfile))
67
68
69  sys.exit()
```

C.3. Simulation

npt.py

```
# This script was generated by OpenMM-Setup on 2024-07-24.

from openmm import *
from openmm.app import *
from openmm.unit import *
from sys import stdout
import parmed as pmd

#restart file

cnt = int(sys.argv[1])
pcnt = cnt-1

if cnt > 1:
    rst = f'traj/npt_{pcnt}.rst'

# System Configuration

nonbondedMethod = PME
nonbondedCutoff = 1.2*nanometers
ewaldErrorTolerance = 0.0005
constraints = HBonds
rigidWater = True
constraintTolerance = 0.000001
hydrogenMass = 1.5*amu

# Integration Options

dt = 0.002*picoseconds
temperature = 300*kelvin
friction = 1.0/picosecond
pressure = 1.0*atmospheres
barostatInterval = 25

# Input Files

pdb = PDBFile('nxm123tb10dfb.pdb')
#psf = CharmmPsfFile('linker_CHARMM.psf')
#params = CharmmParameterSet('linker_CHARMM.rtf', 'linker_CHARMM.prm')

forcefield = ForceField('SNUFF_2024.xml')

#with open('Pyrene_Acepype_2024.xml') as input:
#    system2 = XmlSerializer.deserialize(input.read())

# Box size

minim = float('inf')
maxim = float('-inf')
for i in range(len(pdb.positions)):
    for j in range(3):
        if pdb.positions[i]._value[j] > maxim:
            maxim = pdb.positions[i]._value[j]
        if pdb.positions[i]._value[j] < minim:
            minim = pdb.positions[i]._value[j]

xtl = maxim - minim
print('System size: {} nm'.format(xtl))
vects = Quantity([[xtl,0.,0.,],[0.,xtl,0.,],[0.,0.,xtl,]] * nanometer)
pdb.topology.setPeriodicBoxVectors(vects)

# Simulation Options

steps = 50000 #500 000 = 1 ns
equilibrationSteps = 1000
platform = Platform.getPlatformByName('CUDA')
platformProperties = {'Precision': 'mixed'}
DCD_steps = 1000 #5000
DataReporter_steps = 1000 #1000
Checkpoint_steps = 10000 #10 000

```

```
72 # Prepare the Simulation
73
74 print('Building system...',flush=True)
75 #topology = psf.topology
76 topology = pdb.topology
77 print('Topology loaded')
78 positions = pdb.positions
79 print('PDB positions found')
80 #system = psf.createSystem(params, nonbondedMethod=nonbondedMethod, nonbondedCutoff=
       nonbondedCutoff,
81 #    constraints=constraints, rigidWater=rigidWater, ewaldErrorTolerance=
       ewaldErrorTolerance, hydrogenMass=hydrogenMass)
82 system = forcefield.createSystem(topology, nonbondedMethod=nonbondedMethod,
       nonbondedCutoff=nonbondedCutoff,
83     constraints=constraints, rigidWater=rigidWater, ewaldErrorTolerance=
       ewaldErrorTolerance, hydrogenMass=hydrogenMass)
84 print('Forcefield system created')
85
86 system.addForce(MonteCarloBarostat(pressure, temperature, barostatInterval))
87 #integrator = LangevinMiddleIntegrator(temperature, friction, dt)
88 integrator = NoseHooverIntegrator(temperature, friction, dt)
89 print('Integrator ready')
90 integrator.setConstraintTolerance(constraintTolerance)
91 print('Constraints set')
92 simulation = Simulation(topology, system, integrator, platform, platformProperties)
93 print('Simulation set up ready')
94 simulation.context.setPositions(positions)
95 print('Positions set')
96
97 if cnt > 1:
98     with open(rst, 'r') as f:
99         simulation.context.setState(XmlSerializer.deserialize(f.read()))
100
101 print('Restart file considered')
102 # Minimize and Equilibrate
103
104 if cnt == 1: #minimize
105     structure = pmd.openmm.load_topology(topology, system)
106     with open('run.psf', 'w') as f:
107         structure.write_psf(f)
108     print('Performing energy minimization...',flush=True)
109     simulation.minimizeEnergy()
110     print('Equilibrating...',flush=True)
111     simulation.context.setVelocitiesToTemperature(temperature)
112     print(simulation.context.getState(getEnergy=True).getPotentialEnergy())
113     print("Saving minimized pdb...",flush=True)
114     integrator.setStepSize(0.0001)
115     simulation.step(equilibrationSteps)
116     print('Equilibration ok',flush=True)
117     simulation.currentStep = 0
118     integrator.setStepSize(dt)
119     positions = simulation.context.getState(getPositions=True).getPositions()
120     PDBFile.writeFile(simulation.topology, positions, open("mini.pdb",'w'))
121
122 dcdReporter = DCDReporter(f'traj/npt_{cnt}.dcd', DCD_steps)
123 dataReporter = StateDataReporter(f'out/npt_{cnt}.out', DataReporter_steps, totalSteps=
       steps,
124     step=True, speed=True, progress=True, potentialEnergy=True, kineticEnergy=True,
       totalEnergy=True, temperature=True, volume=True, density=True, separator='\t')
125 checkpointReporter = CheckpointReporter('checkpoint.chk', Checkpoint_steps)
126
127 # Write XML serialized objects
128
129 #with open("nu1000_save.xml", mode="w") as file:
130 #    file.write(XmlSerializer.serialize(system))
131 #with open("integrator_nu1000.xml", mode="w") as file:
132 #    file.write(XmlSerializer.serialize(integrator))
133
134 # Simulate
135
136 print('Simulating...',flush=True)
137 simulation.reporters.append(dcdReporter)
138 simulation.reporters.append(dataReporter)
139 simulation.reporters.append(checkpointReporter)
140 simulation.step(steps)
141
```

```
#write restart file

state = simulation.context.getState( getPositions=True, getVelocities=True )
with open(f'traj/npt_{cnt}.rst', 'w') as f:
    f.write(XmlSerializer.serialize(state))

#crd = simulation.context.getState(getPositions=True).getPositions()
```

nvt.py

```
# This script was generated by OpenMM-Setup on 2024-07-24.

from openmm import *
from openmm.app import *
from openmm.unit import *
from sys import stdout
import parmed as pmd

#restart file
lastrst = 10 #last npt run for nvt 1
cnt = int(sys.argv[1])
pcnt = cnt-1

if cnt == 1:
    rst = f'traj/npt_{lastrst}.rst'

if cnt > 1:
    rst = f'traj/nvt_{pcnt}.rst'

# System Configuration

nonbondedMethod = PME
nonbondedCutoff = 1.2*nanometers
ewaldErrorTolerance = 0.0005
constraints = HBonds
rigidWater = True
constraintTolerance = 0.000001
hydrogenMass = 1.5*amu

# Integration Options

dt = 0.0001*picoseconds
temperature = 300*kelvin
friction = 1.0/picosecond
pressure = 1.0*atmospheres
barostatInterval = 25

# Input Files

pdb = PDBFile('nxm123tb10dfb.pdb')
#psf = CharmmPsfFile('linker_CHARMM.psf')
#params = CharmmParameterSet('linker_CHARMM.rtf', 'linker_CHARMM.prm')

forcefield = ForceField('SNUFF_2024.xml')

#with open('Pyrene_Acepype_2024.xml') as input:
#    system2 = XmlSerializer.deserialize(input.read())

# Box size

minim = float('inf')
maxim = float('-inf')
for i in range(len(pdb.positions)):
    for j in range(3):
        if pdb.positions[i]._value[j] > maxim:
            maxim = pdb.positions[i]._value[j]
        if pdb.positions[i]._value[j] < minim:
            minim = pdb.positions[i]._value[j]

xtl = maxim - minim
print('System size: {} nm'.format(xtl))
vects = Quantity([[xtl,0.,0.,],[0.,xtl,0.,],[0.,0.,xtl,]] * nanometer)
pdb.topology.setPeriodicBoxVectors(vects)

```

```
# Simulation Options

steps = 25000000 #10 000 000 for 1 ns (dt 0.1 fs)
equilibrationSteps = 1000
platform = Platform.getPlatformByName('CUDA')
platformProperties = {'Precision': 'mixed'}
DCD_steps = 5000 #5000
DataReporter_steps = 5000 #1000
Checkpoint_steps = 10000 #10 000

# Prepare the Simulation

print('Building system...',flush=True)
#topology = psf.topology
topology = pdb.topology
print('Topology loaded')
positions = pdb.positions
print('PDB positions found')
#system = psf.createSystem(params, nonbondedMethod=nonbondedMethod, nonbondedCutoff=nonbondedCutoff,
#    constraints=constraints, rigidWater=rigidWater, ewaldErrorTolerance=ewaldErrorTolerance, hydrogenMass=hydrogenMass)
system = forcefield.createSystem(topology, nonbondedMethod=nonbondedMethod, nonbondedCutoff=nonbondedCutoff,
    constraints=constraints, rigidWater=rigidWater, ewaldErrorTolerance=ewaldErrorTolerance, hydrogenMass=hydrogenMass)
print('Forcefield system created')

#integrator = LangevinMiddleIntegrator(temperature, friction, dt)
integrator = NoseHooverIntegrator(temperature, friction, dt)
print('Integrator ready')

integrator.setConstraintTolerance(constraintTolerance)
print('Constraints set')
simulation = Simulation(topology, system, integrator, platform, platformProperties)
print('Simulation set up ready')
simulation.context.setPositions(positions)
print('Positions set')

with open(rst, 'r') as f:
    simulation.context.setState(XmlSerializer.deserialize(f.read()))

print('Restart file considered')

dcdReporter = DCDReporter(f'traj/nvt_{cnt}.dcd', DCD_steps)
dataReporter = StateDataReporter(f'out/nvt_{cnt}.out', DataReporter_steps, totalSteps=steps,
    step=True, speed=True, progress=True, potentialEnergy=True, kineticEnergy=True, totalEnergy=True, temperature=True, volume=True, density=True, separator='\t')
checkpointReporter = CheckpointReporter('checkpoint.chk', Checkpoint_steps)

# Write XML serialized objects

#with open("nu1000_save.xml", mode="w") as file:
#    file.write(XmlSerializer.serialize(system))
#with open("integrator_nu1000.xml", mode="w") as file:
#    file.write(XmlSerializer.serialize(integrator))

# Simulate

print('Simulating...',flush=True)
simulation.reporters.append(dcdReporter)
simulation.reporters.append(dataReporter)
simulation.reporters.append(checkpointReporter)
simulation.step(steps)

#write restart file

state = simulation.context.getState( getPositions=True, getVelocities=True )
with open(f'traj/nvt_{cnt}.rst', 'w') as f:
    f.write(XmlSerializer.serialize(state))

#crd = simulation.context.getState(getPositions=True).getPositions()
```

run.sh

```
#!/bin/bash

#SBATCH -p gpu
#SBATCH --gres=gpu
#SBATCH -x n0021

source /home/marion/miniconda3/bin/activate openmm

inp_file="nvt4DE_nXm123tb10dfb.py"
job_name="NUTm1"
last=40

python3 $inp_file $1

ret_val=$?

if [ $ret_val -eq 0 -a $1 -lt $last ] ; then
    let next=$1+1
    sbatch -J ${job_name}_$next -o out/nvt_$next.out run_nvt4.sh $next
elif [ ! $ret_val -eq 0 ] ; then
    echo "Error in run $1 .." >> out/error.log
fi
```

C.4. Trajectory Analysis

analysis.py

File structure for DE mode -> ./analysis.py , ./DE/MDMD_fit.* ./traj/*.dcd ./run.psf

```
import numpy as np
import MDAnalysis
from newanalysis.diffusion import msdCOM, unfoldTraj, msdMJ, msdMJdecomp, msdMJcrossterms
from newanalysis.correl import correlateParallel
from newanalysis.functions import atomsPerResidue, residueFirstAtom # "apr", "rfa", needed for faster calculation of centerOfMassByResidue, dipoleMomentByResidue
from newanalysis.functions import centerOfMassByResidue, dipoleMomentByResidue
import os, sys
import time
from datetime import timedelta
from pathlib import Path
import re
import argparse
from scipy.signal import savgol_filter
####################################dielectric only
import subprocess
import matplotlib.pyplot as plt
import matplotlib.cm
import json



############
# argparse
############
parser = argparse.ArgumentParser()
parser.add_argument("-s", "--short", help="non verbose option, no printing of current frame calculation", action="store_true")
parser.add_argument('-i', '--infrared', help='calculate IR spectra', action='store_true', default=False)
parser.add_argument('-d', '--dielectric', help='calculate dielectric spectra', action='store_true', default=False)
parser.add_argument('-t', '--traj_anal', help='skip loading trajectory if already done', action='store_false', default=True)
args = parser.parse_args()


#####################################################################
#####################################################################
# Trajectory
#####################################################################
#####################################################################
firstfile=31
lastfile=40
skip=1

#####################################################################RES PARA
tasks = ["all", "mof", "mIs", 'dfb']
residues = ["NOD", "LNK", "TOH","MER", "DFB", "BRO", "TBA"]
mof_res = ["NOD", "LNK", "TOH"]
spi_res = ["MER"]
molecules = ["NOD", "LNK", "TOH","MER", "DFB"] #residues without ions for mdmd
############################################################ PATH
base = "/site/raid6/johnny/mof/sim10_nu1t-spiroII/ions/TBABr_10/m123/"
#############################################################
psf_path = base + "run.psf"
dcd_path = base + "traj/nvt_"
ir_path = "./IR"
de_path = './DE'
mdmdfolder = "/mdmd"
msdmjfolder = "/msdmj"
msdfolder = "/msd"
#########################################################SIM PARA
T_sim = 300 #K
boxl = 101.36 #Angstrom
#####################################################################SAVGOL PARA
savgol_size = 12
norm2 = 1
#####################################################################DIEL FITSIZE
maxtime = 5000 #1000
```

```
66
67 if args.dielectric:
68     mdmdfolder = de_path + mdmdfolder
69     msdmjfolder = de_path + msdmjfolder
70     msdfolder = de_path + msdfolder
71     Path(de_path).mkdir(parents=True, exist_ok=True)
72 elif args.infrared:
73     mdmdfolder = ir_path + mdmdfolder
74     msdmjfolder = ir_path + msdmjfolder
75     msdfolder = ir_path + msdfolder
76     Path(ir_path).mkdir(parents=True, exist_ok=True)
77 else:
78     print('No mode selected. Saving files in working directory.')
79
80 ##############################################################################
81 ##############################################################################
82 ##############################################################################
83 ######################## D E F I N I T I O N S ################################
84 ##############################################################################
85 ##############################################################################
86 ##############################################################################
87
88 def msd_com(com_array):
89     """
90     Calculates mean squared displacement using the Fast Correlation Algorithm in the
       package tidynamics
91     Dependency: tidynamics
92     input: numpy array with shape (timesteps, particles, xyz)
93     returns: numpy array with shape (timestep, msd)
94     """
95     import tidynamics
96
97     com_array = com_array.astype(np.float64)
98     time, n_particles, ndim = com_array.shape
99     msds_by_particle = np.zeros((time, n_particles))
100    for particle in range(n_particles):
101        msds_by_particle[:, particle] = tidynamics.msd(com_array[:, particle, :])
102    msd = msds_by_particle.mean(axis=1)
103    return msd
104
105
106 def msd_mj(*args: tuple[np.array, int]):
107     """
108     Calculates mj using the Fast Correlation Algorithm in the package tidynamics
109     Dependency: tidynamics
110     input: any number of tuples: (numpy array with shape (timesteps, n_particles, ndim),
       int charge or numpy array of charges)
111     numpy array can either be 1D with n_particle charge values or 2D with n_timesteps,
       n_particle number of charges
112     input arrays must all have same timesteps and dimension, number of particles may vary
       between two separate arrays
113     Example call of function: msdmj = msd_mj((com_cat, 1), (com_an, -1)) or msd_mj((
       com_cat, np.array([[1,1,1,][1,1,1]])) where charges are for three particles with two
       timesteps
114     int charges and array charges can be mixed for different com array inputs
115
116     if only one input tuple is given, just the autocorelation of it will be calculated, eg
       . sigma++ for ie im1h or sigma-- for oac (analog to msdMJcrossterms) not working for
       cossterms like simga+-
117     """
118     import tidynamics
119
120     time, particles, ndim = args[0][0].shape
121     mj = np.zeros((time, ndim), dtype=np.float64)
122     for msd, charge in args:
123         if isinstance(charge, int):
124             mj_tmp = msd.sum(axis=1) * charge
125         elif isinstance(charge, (np.ndarray, np.generic)):
126             if charge.ndim > 2:
127                 print(
128                     f"Error in msd_mj function! Charge array has more dimensions ({charge.
       ndim}) than allowed."
129                 )
130                 print(
131                     "Only 1D (charges for positions, same at each timestep) or 2D (charges
        for each timestep and position) numpy arrays are allowed"
```

```
                )
                print("Exiting...")
                sys.exit()
            if (
                charge.shape[0] == time and charge.shape[1] == msd.shape[1]
            ):  # time, particles
                # print("Using custom charges for each timestep and particle")
                mj_tmp = (msd * charge[:, :, np.newaxis]).sum(axis=1)
            elif charge.shape[0] == msd.shape[1]:  # particles
                # print("Using custom charges for each particle")
                mj_tmp = (msd * charge[np.newaxis, :, np.newaxis]).sum(axis=1)
        else:
            print("Error in msd_mj function!!!")
            print(
                "charge must be suplied either as one (1) int or as numpy array with 1 or 2 dimensions"
            )
            print("Exiting...")
            sys.exit()
        mj += mj_tmp
    msd_mj = tidynamics.msd(mj)
    # msd_mj = tidynamics.correlation(args[0][0].sum(axis=1)*args[0][1], args[1][0].sum(axis=1)*args[1][1])

    return msd_mj


# Note: to use msd_mj with charge array do sth like:
# charges = [[charge]*n_particles]*time
# charges = np.asarray(charges)
# msdmj = msd_mj((com_cat. charges), (com_an, -1))


################ Laplace transform ####################################################
# zero padding does not change
def laplacetransform(time,corfun):
    print('>\tLaplace transform')
    xlen = len(time)
    #x = time
    #y = corfun
    x = np.zeros(2*xlen-1)
    y = np.zeros(2*xlen-1)
    x[0:xlen]  = -time[::-1]
    x[xlen-1:] = time
    y[xlen-1:] = corfun

    # simple Fourier transform
    dt = x[1]-x[0]
    freq = 1./dt
    shifty = np.fft.fftshift(y)
    fourier = np.fft.fft(shifty)/freq

    N = int(len(fourier)/2)+1

    frequencies = np.linspace(0,freq/2,N,endpoint=True)

    return frequencies, fourier


############## Apodization = Smoothing ################################################
def apodization(alpha,x,y):
    # J. Non.-Cryst. Solids (1992), 140, 350
    # Phys. Rev. Lett. (1996), 77, 4023
    print('>\tApodization of correlation function with alpha = %10.5f'%alpha)

    xlen = len(x)
    ysmooth = np.zeros(xlen)

    for i in range(xlen):
        # apodization
        ysmooth[i] = y[i] * np.exp(-alpha*x[i]*x[i])
    return ysmooth


##### maxperiod, e.g. first 100 ps of the trajectory corresponds to 0.2 cm^-1 IR resolution ####
```

```
205 def maxperiod(x,y,xmax):
206     print('>\tMaximum simulation period = %8.1f ps'%xmax)
207     new_x = []
208     new_y = []
209     xlen = len(x)
210 
211     for i in range(xlen):
212         new_x.append(x)
213         new_y.append(y)
214         if x[i] >xmax:
215             break
216     return new_x, new_y
217 
218 
219 ###################### Prefactor ############################################################
220 def quantum_correction(frequencies,spectrum):
221     print("\tApplying quantum correction")
222 
223     # beta = 1. / (1.3806 10^-23 J/K * 300 K )
224     # hbar = 1.0545718*10^-34 J/s
225     # factor = beta * hbar ( 10^12 ps/s)
226     beta_hbar = 0.0254617
227     for i in range(len(frequencies)):
228         qc = beta_hbar * frequencies[i] / (1.0 - np.exp(-beta_hbar*frequencies[i]))
229         spectrum[i] *= qc
230     return spectrum
231 
232 
233 ################ Baseline correction ####################################################
234 def baseline(fourier):
235     print('>\tBaseline correction')
236     min = np.min(spectrum)
237     return spectrum -min
238 
239 
240 def find_first_index(array, threshold):
241     mask = array > threshold
242     indices = np.where(mask)[0]
243     if len(indices) > 0:
244         return indices[0]
245     return -1  # If no value is larger than the threshold
246 
247 
248 
249 ###########################################################################################
250 ################## MAIN PROGRAM ###########################################################
251 ###########################################################################################
252 
253 if args.traj_anal:
254     # check manually for right paths t/ psf and dcd files!
255     print("Checking for psf & dcd")
256     dcd = ["%s%d.dcd" % (dcd_path,i) for i in range(firstfile,lastfile+1)]
257     print("PSF file:", psf_path)
258     print("DCD base:", dcd_path)
259 
260     print(f"Using dcd files from {firstfile} to {lastfile} with skip {skip}")
261     print(f"Using psf {psf_path}")
262 
263     u=MDAnalysis.Universe(psf_path,dcd)
264     boxl = np.float64(round(u.coord.dimensions[0],4))
265     dt=round(u.trajectory.dt,4)
266 
267     #n = u.trajectory.numframes//skip  # used integer division (//) instead of (/) -> so
    dtype is int, otherwise problems with line 42 com_cat=np.zeros(...)
268     n = int(u.trajectory.n_frames/skip)
269     if u.trajectory.n_frames%skip != 0:
270         n+=1
271     print(f"Boxlength is {boxl}")
272     ########################################################################
273     ########################################################################
274     # molecular species
275     ########################################################################
276     ########################################################################
277     sel = {}
278     n_res = {}
279     masses = {}
```

```
    charges = {}
    apr = {}
    rfa = {}
    coms = {}
    md = {}
    mu={}
    md_total = np.zeros((n,3),dtype=np.float64)
    md_mof = np.zeros((n,3),dtype=np.float64)
    md_spi = np.zeros((n,3),dtype=np.float64)

    for residue in residues:
        sel[residue] = u.select_atoms(f"resname {residue}")
        n_res[residue] = sel[residue].n_residues
        masses[residue] = sel[residue].masses
        charges[residue] = sel[residue].charges
        apr[residue] = atomsPerResidue(sel[residue])
        rfa[residue] = residueFirstAtom(sel[residue])
        coms[residue] = np.zeros((n,n_res[residue],3), dtype=np.float64)
        md[residue] = np.zeros((n,3),dtype=np.float64)
        mu[residue] = np.zeros((n,n_res[residue]*3), dtype=np.float64)
        print(f"Number of {residue}: {n_res[residue]}")


    ######################################################################
    ######################################################################
    # Running through trajectory
    ######################################################################
    ######################################################################
    ctr=0

    start=time.time()
    print("")

    for ts in u.trajectory[::skip]:
        if not args.short:
            print("\033[1AFrame %d of %d" % (ts.frame,len(u.trajectory)), "\tElapsed time: %.2f hours" % ((time.time()-start)/3600))#"    Est. time left: %s" % (str(timedelta(seconds=(time.time()-start)/ts.frame * (u.trajectory.numframes-ts.frame)))[:7]))

        for residue in residues:
            coms[residue][ctr] = sel[residue].center_of_mass(compound="residues")
            coor = np.ascontiguousarray(sel[residue].positions, dtype='double') #array shape needed for C
            #com  = centerOfMassByResidue(sel[residue], coor=coor, masses=masses[residue], apr=apr[residue], rfa=rfa[residue])
            md[residue][ctr] += np.sum(dipoleMomentByResidue(sel[residue], coor=coor, charges=charges[residue],masses=masses[residue],com=coms[residue][ctr], apr=apr[residue], rfa=rfa[residue],axis=0), axis=0)
            mu[residue][ctr] = dipoleMomentByResidue(sel[residue], coor=coor,charges=charges[residue],masses=masses[residue],com=coms[residue][ctr], apr=apr[residue], rfa=rfa[residue],axis=0).flatten()
            md_total[ctr]     += md[residue][ctr]

        for residue in mof_res:
            md_mof[ctr] += md[residue][ctr]
        for residue in spi_res:
            md_spi[ctr] += md[residue][ctr]
        #imd_mof[ctr] = md["NOD"][ctr]+md["LNK"][ctr]+md["MER"][ctr]+md["TOH"][ctr] #maybe ????????
        ctr+=1

    print(f"Time elapsed: {time.time()-start}")


    ######################################################################
    ######################################################################
    # Post-processing
    ######################################################################
    ######################################################################

    print("unfolding coordinates ..")
    for residue in residues:
        unfoldTraj(coms[residue],boxl)

    time_axis = np.arange(0, n*skip*dt, skip*dt) # eigentlich ab dt bis lastvalue included

```

```
347     print("calculating <MD(0)MD(t)>")
348     Path(mdmdfolder).mkdir(parents=True, exist_ok=True)
349 
350     # total
351     md_total = np.ascontiguousarray(md_total.T)
352     md0mdt = np.zeros(n)
353     correlateParallel(md_total,md_total,md0mdt,ltc=1)
354     np.savetxt(f"{mdmdfolder}/md0mdt_all.dat", np.c_[time_axis, md0mdt])
355 
356     # mof
357     md_mof = np.ascontiguousarray(md_mof.T)
358     md0mdt = np.zeros(n)
359     correlateParallel(md_mof,md_mof,md0mdt,ltc=1)
360     np.savetxt(f"{mdmdfolder}/md0mdt_mof.dat", np.c_[time_axis, md0mdt])
361     #np.savetxt(f"{mdmdfolder}/mu_mof_ts.dat", np.c_[time_axis,md_mof]) #ERROR array size?
362 
363     # spi
364     md_spi = np.ascontiguousarray(md_spi.T)
365     md0mdt = np.zeros(n)
366     correlateParallel(md_spi,md_spi,md0mdt,ltc=1)
367     np.savetxt(f"{mdmdfolder}/md0mdt_mIs.dat", np.c_[time_axis, md0mdt])
368 
369     print("calculating msd ..")
370     Path(msdfolder).mkdir(parents=True, exist_ok=True)
371 
372     #cross corr
373     for residue in residues:
374         msd = msd_com(coms[residue])
375         np.savetxt(f"{msdfolder}/msd_{residue.lower()}_{firstfile}-{lastfile}_{skip}.dat",
        np.c_[time_axis, msd])
376 
377         md[residue] = np.ascontiguousarray(md[residue].T)
378         md0mdt = np.zeros(n)
379         correlateParallel(md[residue],md_total,md0mdt,ltc=1)
380         np.savetxt(f"{mdmdfolder}/md0mdt_{residue.lower()}.dat", np.c_[time_axis, md0mdt])
381 
382     ############ MSD MJ
    ########################################################################
383 
384     print("calculating msdMJ ..")
385     Path(msdmjfolder).mkdir(parents=True, exist_ok=True)
386 
387     msdmj = msd_mj((coms["TBA"], 1), (coms["BRO"], -1))
388 
389     timeseries_msdMJ_cat = msdMJdecomp(coms["TBA"], coms["BRO"], coms["TBA"],  1)
390     timeseries_msdMJ_ani = msdMJdecomp(coms["TBA"], coms["BRO"], coms["BRO"],  -1)
391     timeseries_msdMJ_catcat = msdMJcrossterms(coms["TBA"], coms["TBA"], 1,  1)
392     timeseries_msdMJ_aniani = msdMJcrossterms(coms["BRO"], coms["BRO"], -1, -1)
393     timeseries_msdMJ_catani = msdMJcrossterms(coms["TBA"], coms["BRO"],  1, -1)
394 
395     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_total.dat",        np.
    c_[time_axis, msdmj])
396     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_total_cation.dat", np.
    c_[time_axis, timeseries_msdMJ_cat])
397     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_total_anion.dat",  np.
    c_[time_axis, timeseries_msdMJ_ani])
398     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_cation_cation.dat", np.
    c_[time_axis, timeseries_msdMJ_catcat])
399     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_anion_anion.dat",   np.
    c_[time_axis, timeseries_msdMJ_aniani])
400     np.savetxt(f"{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_cation_anion.dat",  np.
    c_[time_axis, timeseries_msdMJ_catani])
401 else:
402     print('Skipping trajectory analysis')
403 
404 
405 ##############################################################################################
406 ##############################################################################################
407 ################ DIELECTRIC CALCULATIONS
408 ##############################################################################################
409 ##############################################################################################
410 if args.dielectric:
411     print('Beginning calculation of dielectric spectra...')
412 
413     fit = True
414     plot = False
```

```
    show_plots = False

    fitfunctions = {}
    maxtimes = []

    for molecule in molecules:
        fname = f"{mdmdfolder}/md0mdt_{molecule.lower()}.dat"
        orig_data = np.loadtxt(fname)
        fitfunctions[molecule] = [orig_data]
        #maxtime = float(os.popen(f"grep -m 1 -B 1 '-' {fname} | head -n1 ").read().split()[0])
        maxtimes.append(maxtime)
        fitname = f"{mdmdfolder}/md0mdt_{molecule.lower()}_fit_{maxtime}.dat"
        #print(f"{maxtime=}")
        if fit:
            newfile = f"{mdmdfolder}/MDMD_fit_{molecule.lower()}_{maxtime}.json"
            #copy input json
            os.system(f"cp {de_path}/MDMD_fit.json {newfile}")
            # replace input file name
            replace_string = f's~"data":.*~"data":      "{fname}",~'
            os.system(f"sed -i '{replace_string}' {newfile}")
            # replace fit filename
            replace_string = f's~"residuals":.*~"residuals": "{fitname}",~'
            os.system(f"sed -i '{replace_string}' {newfile}")
            # replace maxtime
            replace_string = f's~"maxtime":.*~"maxtime": {maxtime},~'
            os.system(f"sed -i '{replace_string}' {newfile}")
            output = subprocess.check_output(f"python {de_path}/MDMD_fit.py {newfile}", shell=True, text=True)
            #output = subprocess.check_output(f"python MDMD_fit.py {newfile}", shell=True, text=True)
        fit_data = np.loadtxt(fitname)
        fitfunctions[molecule].append(fit_data)
    print('MDMD_fit done!')

    msdMJ_infile = f'{de_path}/msdMJ_inp.json'
    msdMJ_is = ['total','total_cation','total_anion','cation_cation','anion_anion','cation_anion']

    for msdMJ_i in msdMJ_is:
        newfile = f'{msdmjfolder}/msdMJ_{msdMJ_i}_{maxtime}.json'
        fitname = f'{msdmjfolder}/msdMJ_fit_{msdMJ_i}_{maxtime}.dat'
        fname = f'{msdmjfolder}/msdMJ_{firstfile}-{lastfile}_{skip}_{msdMJ_i}.dat'
        os.system(f'cp {msdMJ_infile} {newfile}')
        # replace input file name
        replace_string = f's~"data":.*~"data":      "{fname}",~'
        os.system(f"sed -i '{replace_string}' {newfile}")
        # replace fit filename
        replace_string = f's~"residuals":.*~"residuals": "{fitname}",~'
        os.system(f"sed -i '{replace_string}' {newfile}")
        # replace maxtime
        replace_string = f's~"maxtime":.*~"maxtime": {maxtime},~'
        os.system(f"sed -i '{replace_string}' {newfile}")
        # replace maxtime
        replace_string = f's~"boxl":.*~"boxl": {format(boxl,".2f")},~'
        os.system(f"sed -i '{replace_string}' {newfile}")
        #run fit
        os.system(f"python {de_path}/msdMJ_fit.py {newfile}")
    print('msdMJ_fit done!')

    with open(f'{mdmdfolder}/maxtimes.dat', 'w') as f:
        for maxtime in maxtimes:
            f.write(f"{str(maxtime)}\n")
        print('Maxtimes written!')
    if plot:
        #savedir = "./plots"
        savedir = "."
        Path(savedir).mkdir(parents=True, exist_ok=True)

        # one endtime, different fits
        for molecule in molecules:
            plt.plot(fitfunctions[molecule][0][:,0],fitfunctions[molecule][0][:,1], "r", label=f"{molecule}, orig", linewidth=1.0)
            plt.axis([0,10000, -50, 100])
            plt.plot(fitfunctions[molecule][0][0:len(fitfunctions[molecule][1][:,2]),0], fitfunctions[molecule][1][:,2], label=f"{molecule}, fit", linewidth=0.5)
```

```
485             plt.xlim(left=-50)
486             plt.ylabel("md0mdt")
487             plt.xlabel("time ps")
488             plt.legend()
489 
490             #a = plt.axes([0.55,0.15,0.35,0.35])
491             #plt.yticks([])
492             #plt.plot(fitfunctions[molecule][0][0:500,0],fitfunctions[molecule
    ][0][0:500,1], "r", label=f"{molecule},orig", linewidth=1.0)
493             #plt.plot(fitfunctions[molecule][0][0:500,0],fitfunctions[molecule
    ][1][0:500,2], label=f"{molecule}, fit", linewidth=0.5)
494             fig = plt.gcf()
495             if show_plots:
496                 plt.show()
497             fig.savefig(f"{savedir}/plot_{molecule}.pdf")
498             plt.close()
499 else:
500     print('DE-Mode: FALSE')
501 
502 ##############################################################################################
503 ##############################################################################################
504 ################ INFRARED   CALCULATIONS
505 ##############################################################################################
506 ##############################################################################################
507 if args.infrared:
508     print('Beginning calculation of infrared spectra...')
509 
510     ############ SECOND GRADIENT
    ################################################################
511     print("calculating gradient...")
512 
513     #file = sys.argv[1]
514     mdmd_files = []
515     for task in tasks:
516         mdmd_files.append(f"md0mdt_{task}.dat")
517 
518     for mdmd_file in mdmd_files:
519         t, C = np.loadtxt(f'{mdmdfolder}/{mdmd_file}', unpack =True)
520         timescale = t[1]-t[0]
521         J = -np.gradient(np.gradient(C)/timescale)/timescale
522         np.savetxt(f'{mdmdfolder}/gradient_{mdmd_file}', np.c_[t,J])
523     print("DONE")
524 
525     ############ IR CALCULATIONS
    ################################################################
526     print("~"*120)
527     print("Laplace transform of < dmd/dt(0) * dmd/dt(t) >")
528     print("~"*120)
529 
530     for task in tasks:
531         # correlation function
532         mudot_mudot = np.loadtxt(f'{mdmdfolder}/gradient_md0mdt_{task}.dat')
533         #mudot_mudot = np.loadtxt("./msdmj/gradient_msdMJ_1-10_1_total.dat")
534         time    = mudot_mudot[:,0]
535         corfun = mudot_mudot[:,1]
536 
537         # apodization = gaussian smoothing
538         alpha = 0.05 #0.05
539         corfun = apodization(alpha,time,corfun)
540 
541 
542         # Laplace transform
543         frequencies, fourier = laplacetransform(time,corfun)
544 
545 
546         THz2cm = 33.356
547 
548         minfreq = find_first_index(frequencies,0./THz2cm)
549         maxfreq = find_first_index(frequencies,3800./THz2cm)
550         frequencies = frequencies[minfreq:maxfreq]
551         spectrum = np.real(fourier[minfreq:maxfreq])
552         #spectrum = spectrum/frequencies brauchen wir nicht da spaeter mit omega/c
    multipliziert wird
553         max_spectrum = np.max(spectrum)
554         #spectrum /= max_spectrum
555 
```

```
556
557         # Baseline correction
558         #spectrum = baseline(spectrum)
559
560
561         # Quantum corrections
562         #spectrum = quantum_correction(frequencies,spectrum)
563
564
565         #output = sys.argv[1].split(".")[0]+".IR"
566         output = f"{ir_path}/{task}.IR"
567         print("\n\tWriting %s"%output)
568         f1 = open(output,'w')
569         for j in range(len(spectrum)):
570             current_frequency = frequencies[j]*THz2cm ####why ERROR ??? ---> if dt & dcd
    step too large, freq available proportional to 1/dt
571             f1.write("%8.1f %12.8f\n" %(current_frequency,spectrum[j]))
572         f1.close()
573
574     ############ CALC PREFACTOR cm-1 absorbtion
    ###############################################
    '''
575     prefactor
576         1          4 pi
577     -------- * ---------
578     4 pi e_0    3 V k T
579     '''
580     k = 8.61733326 #10^(-5) eV/K
581     e_0 = 8.85     #10^(-12) As/Vm
582     elementary_charge = 1.6022 #10^(-19) As
583     c = 0.0299792 # speed of light in cm / ps
584     V = boxl**3 #in angstrom^3
585     beta = 1/(k*T_sim)
586     prefactor =  elementary_charge * 1.E8 / (3 * V * k * T_sim * e_0)
587     print(prefactor/c)
588
589     input_files = []
590     output_files = []
591     for task in tasks:
592         input_files.append(f"{ir_path}/{task}.IR")
593         output_files.append(f"{ir_path}/{task}.thz")
594
595
596     n_files = len(input_files)
597
598     for f in range(n_files):
599         omega, spectrum = np.loadtxt(input_files[f], unpack = True)
600         spectrum *= prefactor/c #nicht omega/c weil omega eigentlich schon multipliziert
    wurde im IR script
601
602         np.savetxt(output_files[f], np.c_[omega,spectrum])
603
604     ############ SAVGOL FILTER
    ################################################################
605     try:
606         mysize = savgol_size #= int(sys.argv[2])
607         # mysize = window length, must be positive odd int
608     except:
609         mysize = None
610
611     for task in tasks:
612         data  = np.loadtxt(f"{ir_path}/{task}.IR") #np.loadtxt(sys.argv[1])
613
614         #========================================
615         # Smoothing
616         #========================================
617         if mysize:
618             #4 = polyorder must be less than window legth
619             smoothed = savgol_filter(data[:,1],mysize,4)
620         else:
621             smoothed = savgol_filter(data[:,1],3,4)
622
623         output = f"{ir_path}/savgol_{task}.IR"
624         f1 = open(output,'w')
625         for i in range(len(smoothed)):
626             f1.write("%8.1f %12.8f\n" %(data[i,0], smoothed[i])) #f1.write
627
```

```
628         f1.close()
629
630     ########### Norm and flip
        ##################################################################
631     xval, yval = np.loadtxt(f"{ir_path}/savgol_all.IR", unpack = True)
632     ymax = max(yval)
633     factor = norm2/ymax
634
635     for task in tasks:
636         xval, yval = np.loadtxt(f"{ir_path}/savgol_{task}.IR", unpack = True)
637         yval = yval * factor * -1 + 1
638         np.savetxt(f"{ir_path}/norm_savgol_{task}.IR", np.c_[xval,yval])
639
640
641 else:
642     print('IR-Mode: FALSE')
643
644 sys.exit()
```

MDMD_fit.py

```
1  from lmfit import Minimizer, Parameters
2  from lmfit.printfuncs import report_fit
3  import numpy as np
4  import sys
5  import json
6  import copy
7
8  print('~'*120)
9  print('MDMD_fit.py'),
10 print('~'*120)
11
12 ##############################################################################
13 class InputClass:
14 ##############################################################################
15     def __init__(self):
16         self.infile  = ''
17         self.outfile = ''
18         self.maxtime = 0
19
20         self.fit_model = 'lestsq'
21         #self.fit_model = 'ampgo'
22         self.initialexp = []
23         self.exp = []
24         self.number_of_exp = 0
25
26     def info(self):
27         print('< Json Input:')
28         print('\tdata          = ',self.infile)
29         print('\tresiduals     = ',self.outfile)
30         print('\tmaximal time = ',self.maxtime)
31         print('\n\texp:')
32         for i in self.exp:
33             i.info()
34         print('\tFit model = ',self.fit_model)
35
36     def fromJson(self,data):
37         if "data" in data:
38             self.infile = data["data"]
39         if "residuals" in data:
40             self.outfile = data["residuals"]
41         if "maxtime" in data:
42             self.maxtime = data["maxtime"]
43         if "exp" in data:
44             for i in data["exp"]:
45                 tmp = ExpClass()
46                 tmp.fromJson(i)
47                 self.exp.append(tmp)
48             self.number_of_exp = len(self.exp)
49             self.initialexp = copy.deepcopy(self.exp)
50         if "fit_model" in data:
51             self.fit_model = data["fit_model"]
52
53     def toJson(self,JsonFile):
```

```
54          print('>\t Writing Json File ',JsonFile)
55          f = open(JsonFile,'w')
56          f.write('{\n')
57          f.write('\t"data":      "%s", \n'%self.infile)
58          f.write('\t"residuals": "%s",\n'%self.outfile)
59          f.write('\t"maxtime":   %s,\n\n'%self.maxtime)
60
61          if len(self.initialexp) >0:
62              f.write('\t"initialexp":\n')
63              f.write('\t[\n')
64              for ctr,i in enumerate(self.initialexp):
65                  if i.a_vary:
66                      f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'true'))
67                  else:
68                      f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'false'))
69                  if i.tau_vary:
70                      f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'true'))
71                  else:
72                      f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'false'))
73                  if not ctr == self.number_of_exp-1:
74                      f.write(',')
75                  f.write('\n')
76              f.write('\t],\n\n')
77          f.write('\t"exp":\n')
78          f.write('\t[\n')
79
80          for ctr,i in enumerate(self.exp):
81              if i.a_vary:
82                  f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'true'))
83              else:
84                  f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'false'))
85              if i.tau_vary:
86                  f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'true'))
87              else:
88                  f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'false'))
89              if not ctr == self.number_of_exp-1:
90                  f.write(',')
91              f.write('\n')
92          f.write('\t]\n')
93          f.write('}\n')
94          f.close()
95
96      def ExampleInput(self):
97          JsonFile = "fit_example.json"
98          self.infile = "md0mdt.dat"
99          self.outfile = "md0mdt_fit.dat"
100         self.maxtime = 5000
101
102         # example tri exponential fit
103         tmp = ExpClass()
104         tmp.a         = 20.0
105         tmp.a_vary    = True
106         tmp.tau       = 0.2
107         tmp_tau_vary = True
108         self.exp.append(tmp)
109
110         tmp = ExpClass()
111         tmp.a         = 50.0
112         tmp.a_vary    = True
113         tmp.tau       = 50.0
114         tmp_tau_vary = True
115         self.exp.append(tmp)
116
117         tmp = ExpClass()
118         tmp.a         = 200.0
119         tmp.a_vary    = True
120         tmp.tau       = 200.0
121         tmp_tau_vary = True
122         self.exp.append(tmp)
123         self.number_of_exp = len(self.exp)
124         self.toJson(JsonFile)
125
126 ##########################################################################
127 class ExpClass:
128 ##########################################################################
129     def __init__(self):
```

```
130         self.a      = 0.0
131         self.a_vary = True
132 
133         self.tau      = 1.0
134         self.tau_vary = True
135 
136     def info(self):
137         if self.a_vary:
138             print('\t\ta = %10.5f        '%self.a, end='')
139         else:
140             print('\t\ta = %10.5f (fixed)'%self.a, end='')
141 
142         if self.tau_vary:
143             print('\ttau = %10.5f        '%self.tau)
144         else:
145             print('\ttau = %10.5f (fixed)'%self.tau)
146 
147     def fromJson(self,data):
148         self.a        = data["a"]["value"]
149         self.a_vary   = data["a"]["vary"]
150         self.tau      = data["tau"]["value"]
151         self.tau_vary = data["tau"]["vary"]
152 
153     def fromFit(self,i,result):
154         self.a   = abs(result.params['a'+str(i+1)].value)
155         self.tau = abs(result.params['tau'+str(i+1)].value)
156 
157 ################################################################################
158 class DataClass:
159 ################################################################################
160     def __init__(self):
161         self.time = []
162         self.f    = []
163         self.derivative = []
164 
165     def fromFile(self,filename,maxtime):
166         try:
167             infile = open(filename,'r')
168         except FileNotFoundError:
169             print('\n! Error!')
170             print('\t data file %s not found!'%filename)
171             sys.exit()
172 
173         tmp_time = []
174         tmp_f = []
175         for line in infile:
176             line_element = line.split()
177             current_time = float(line_element[0])
178             if current_time > maxtime:
179                 break
180             if len(line)>1:
181                 tmp_time.append(current_time)
182                 tmp_f.append(float(line_element[1]))
183         self.time = np.asarray(tmp_time)
184         self.f    = np.asarray(tmp_f)
185 
186     def compute_derivative(self):
187         ########################################################################
188         # "Numerische Methoden" J. Douglas Faiers / R.L. Burden, p. 168
189         ########################################################################
190         print('\n> Computing derivative of the time series ...')
191         h = self.time[1] - self.time[0]
192         tmp_derivative = []
193 
194         # end point calculation of the first two values of d/dt self.f
195         tmp = -2.0833333*self.f[0]+4.0*self.f[1]-3.0*self.f[2]+1.3333333*self.f[3]-0.25*
    self.f[4]
196         tmp_derivative.append(tmp/h)
197 
198         tmp = -0.25*self.f[0]-0.8333333*self.f[1]+1.5*self.f[2]-0.5*self.f[3]+0.08333333*
    self.f[4]
199         tmp_derivative.append(tmp/h)
200 
201         # mid point calculation of d/dt self.f
202         lenf = len(self.f)
203         for i in range(lenf-5):
```

```
204             tmp = 0.0833333*self.f[i]-0.666666*self.f[i+1]+0.666666*self.f[i+3]-0.0833333*
    self.f[i+4]
205             tmp_derivative.append(tmp/h)
206
207         # end point calculation of the last values of d/dt self.f
208         tmp = -0.0833333*self.f[i]+0.5*self.f[i+1]-1.5*self.f[i+2]+0.8333333*self.f[i
    +3]+0.25*self.f[i+4]
209         tmp_derivative.append(tmp/h)
210
211         tmp = 0.25*self.f[i]-1.333333*self.f[i+1]+3.0*self.f[i+2]-4.0*self.f[i
    +3]+2.08333333*self.f[i+4]
212         tmp_derivative.append(tmp/h)
213         tmp_derivative.append(0.0)
214         self.derivative = np.asarray(tmp_derivative)
215
216 ##############################################################################
217 # Fit model
218 ##############################################################################
219 def residual(parfit, t, fdata=None, ddata = None):
220     model1 = 0.0
221     model2 = 0.0
222     for i in range(input.number_of_exp):
223         tmp_a   = parfit['a'+str(i+1)].value
224         tmp_tau = parfit['tau'+str(i+1)].value
225
226         # correlation function
227         model1 += abs(tmp_a) * np.exp(-t/abs(tmp_tau))
228
229         # its derivative
230         model2 -= abs(tmp_a) / abs(tmp_tau) * np.exp(-t/abs(tmp_tau))
231
232     if fdata is None:
233         resid1 = model1
234     else:
235         resid1 = model1 - fdata
236
237     if ddata is None:
238         resid2 = model2
239     else:
240         resid2 = model2 - ddata
241     return np.concatenate(( resid1, resid2 ))
242
243
244
245 ##############################################################################
246 # Main program
247 ##############################################################################
248
249 # Generating input object
250 input = InputClass()
251 try:
252     JsonFile = sys.argv[1]
253     with open(JsonFile) as infile:
254         data = json.load(infile)
255     input.fromJson(data)
256     input.info()
257 except IndexError:
258     print('\n! Error!')
259     print('!\t Json input file is missing: python3 mdmd_fit.py ___.json. ')
260     print('!\t Writing example input fit_example.json')
261     input.ExampleInput()
262     sys.exit()
263 except FileNotFoundError:
264     print('\n! Error!')
265     print('!\t Json input file %s not found!'%JsonFile)
266     sys.exit()
267
268 # Generating data object
269 data = DataClass()
270 data.fromFile(input.infile,input.maxtime)
271 data.compute_derivative()
272
273 # Generating fit model
274 print('\n> Fitting ...')
275
276 print('\t Setting up fit model ...')
```

```
277 parfit = Parameters()
278 for i,exp in enumerate(input.exp):
279     parfit.add('a'+str(i+1), value = exp.a, vary = exp.a_vary, min = 0.05*data.f[0], max =
        data.f[0])
280
281     if i>0:
282         min_tau = parfit['tau'+str(i)]*2.0
283     else:
284         min_tau = 0.1
285
286     max_tau = 0.5*float((i+1)/input.number_of_exp)*data.time[-1]
287     parfit.add('tau'+str(i+1), value = exp.tau, vary = exp.tau_vary, min = min_tau, max =
        max_tau)
288
289
290 print('\t Fitting data, minimizing residuals ...\n')
291 myfit  = Minimizer(residual, parfit,fcn_args=(data.time,), fcn_kws={'fdata': data.f, '
        ddata':data.derivative}, scale_covar=True)
292 result = myfit.minimize(input.fit_model)
293 fit    = residual(result.params, data.time)
294 resids = residual(result.params, data.time, data.f, data.derivative)
295 report_fit(result)
296 print('\n')
297
298 # update exp section in Json
299 for i,exp in enumerate(input.exp):
300     exp.fromFit(i,result)
301 input.toJson(JsonFile)
302
303 print('\n>\t Writing residual file = ',input.outfile)
304 f = open(input.outfile,'w')
305 fitlen = int(len(fit)/2)
306 for i in range(fitlen):
307     f.write("%10.5f "%data.time[i])
308     f.write("%10.5f "%data.f[i])
309     f.write("%10.5f "%fit[i])
310     f.write("%10.5f "%resids[i])
311     f.write("%10.5f "%data.derivative[i])
312     f.write("%10.5f "%fit[i+fitlen])
313     f.write("%10.5f "%resids[i+fitlen])
314     f.write("\n")
```

MDMD_fit.json

```
1  {
2    "data":      "md0mdt_im1h_avg.dat",
3    "residuals": "md0mdt_im1h_fit_2688.75.dat",
4    "maxtime":   2688.75,
5
6    "initialexp":
7    [
8      { "a": {"value":   14.15406, "vary": true}, "tau": {"value":    0.17804, "vary": true}
        },
9      { "a": {"value":    3.21456, "vary": true}, "tau": {"value":    6.44233, "vary": true}
        },
10     { "a": {"value":   28.78433, "vary": true}, "tau": {"value": 1021.28872, "vary": true}
        }
11   ],
12
13   "exp":
14   [
15     { "a": {"value":    1.21455, "vary": true}, "tau": {"value":    1.44160, "vary": true}
        },
16     { "a": {"value":    3.21455, "vary": true}, "tau": {"value":    6.44160, "vary": true}
        },
17     { "a": {"value":   28.78434, "vary": true}, "tau": {"value": 1021.27942, "vary": true}
        }
18   ]
19 }
```

msdMJ_fit.py

```
from lmfit import Minimizer, Parameters
from lmfit.printfuncs import report_fit
import numpy as np
import sys
import json
import copy
from decimal import Decimal

print('~'*120)
print('msdMJ_fit.py'),
print('~'*120)

################################################################################
class InputClass:
################################################################################
    def __init__(self):
        self.infile  = ''
        self.outfile = ''
        self.charmmfile = ''
        self.boxl = ''
        self.maxtime = 0

        self.fit_model = 'leastsq'
        #self.fit_model = 'ampgo'
        self.initialexp = []
        self.exp = []
        self.number_of_exp = 0

        self.initialline = LineClass()
        self.line = LineClass()

    def info(self):
        print('< Json Input:')
        print('\tdata          = ',self.infile)
        print('\tresiduals     = ',self.outfile)
        print('\tmaximal time = ',self.maxtime)
        print('\tcharmmfile    = ',self.charmmfile)
        print('\tboxlength    = ',self.boxl)
        print('\n\texp:')
        for i in self.exp:
            i.info()
        print('\n\tline:')
        self.line.info()

        print('\n\tFit model = ',self.fit_model)

    def fromJson(self,data):
        if "data" in data:
            self.infile = data["data"]
        if "maxtime" in data:
            self.maxtime = data["maxtime"]
        if "residuals" in data:
            self.outfile = data["residuals"]
            self.outfile = self.outfile[:-4] + "_" + str(self.maxtime) + ".dat"
        if "exp" in data:
            for i in data["exp"]:
                tmp = ExpClass()
                tmp.fromJson(i)
                self.exp.append(tmp)
            self.number_of_exp = len(self.exp)
            self.initialexp = copy.deepcopy(self.exp)
        if "fit_model" in data:
            self.fit_model = data["fit_model"]
        if "line" in data:
            # taking only the last slope and overwrite former declarations
            for i in data["line"]:
                self.line = LineClass()
                self.line.fromJson(i)
            self.initialline = copy.deepcopy(self.line)
        ##### optional ######
        if "charmmfile" in data:
            self.charmmfile = data["charmmfile"]
        if "boxl" in data: #not needed if self.charmmfile
            self.boxl = data["boxl"]
```

```
75
76      def toJson(self,JsonFile):
77          print('>\t Writing Json File ',JsonFile)
78          f = open(JsonFile,'w')
79          f.write('{\n')
80          f.write('\t"data":       "%s", \n'%self.infile)
81          f.write('\t"residuals": "%s",\n'%(self.outfile.replace('_'+str(self.maxtime), ''))
        )
82          f.write('\t"charmmfile": "%s",\n'%self.charmmfile)
83          f.write('\t"boxl": "%s",\n'%self.boxl)
84          f.write('\t"maxtime":   %s,\n\n'%self.maxtime)
85
86          if len(self.initialexp) >0:
87              f.write('\t"initialexp":\n')
88              f.write('\t[\n')
89              for ctr,i in enumerate(self.initialexp):
90                  if i.a_vary:
91                      f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'true'))
92                  else:
93                      f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'false'))
94                  if i.tau_vary:
95                      f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'true'))
96                  else:
97                      f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'false'))
98                  if not ctr == self.number_of_exp-1:
99                      f.write(',')
100                 f.write('\n')
101             f.write('\t],\n\n')
102         f.write('\t"exp":\n')
103         f.write('\t[\n')
104
105         for ctr,i in enumerate(self.exp):
106             if i.a_vary:
107                 f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'true'))
108             else:
109                 f.write('\t\t{ "a": {"value": %10.5f, "vary": %s}, '%(i.a,'false'))
110             if i.tau_vary:
111                 f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'true'))
112             else:
113                 f.write('"tau": {"value": %10.5f, "vary": %s} }' %(i.tau,'false'))
114             if not ctr == self.number_of_exp-1:
115                 f.write(',')
116             f.write('\n')
117         f.write('\t],\n')
118
119         if self.initialline.slope>0.0:
120             f.write('\n\t"initialline":\n')
121             f.write('\t[\n')
122             if self.initialline.slope_vary:
123                 f.write('\t\t{ "slope": {"value": %.8f, "vary": %s} }\n'%(self.initialline
        .slope,'true'))
124             else:
125                 f.write('\t\t{ "slope": {"value": %.8f, "vary": %s} }\n'%(self.initialline
        .slope,'false'))
126             f.write('\t],\n')
127         f.write('\n\t"line":\n')
128         f.write('\t[\n')
129         if self.line.slope_vary:
130             f.write('\t\t{ "slope": {"value": %.8f, "vary": %s} }\n'%(self.line.slope,'
        true'))
131         else:
132             f.write('\t\t{ "slope": {"value": %.8f, "vary": %s} }\n'%(self.line.slope,'
        false'))
133         f.write('\t]\n')
134         f.write('}\n')
135         f.close()
136
137     def ExampleInput(self):
138         JsonFile = "msdMJ.json"
139         self.infile  = "msdMJ.dat"
140         self.outfile = "msdMJ_fit.dat"
141         self.maxtime = 500
142
143         # example tri exponential fit
144         tmp = ExpClass()
145         tmp.a         = 20.0
```

```
146         tmp.a_vary    = True
147         tmp.tau       = 0.2
148         tmp_tau_vary = True
149         self.exp.append(tmp)
150 
151         tmp = ExpClass()
152         tmp.a         = 50.0
153         tmp.a_vary    = True
154         tmp.tau       = 50.0
155         tmp_tau_vary = True
156         self.exp.append(tmp)
157 
158         tmp = ExpClass()
159         tmp.a         = 200.0
160         tmp.a_vary    = True
161         tmp.tau       = 200.0
162         tmp_tau_vary = True
163         self.exp.append(tmp)
164         self.number_of_exp = len(self.exp)
165 
166         self.line.slope      = 6.0
167         self.line.slope_vary = True
168 
169         self.toJson(JsonFile)
170 
171 ##############################################################################
172 class ExpClass:
173 ##############################################################################
174     def __init__(self):
175         self.a      = 0.0
176         self.a_vary = True
177 
178         self.tau      = 1.0
179         self.tau_vary = True
180 
181     def info(self):
182         if self.a_vary:
183             print('\t\ta = %10.5f         '%self.a, end='')
184         else:
185             print('\t\ta = %10.5f (fixed)'%self.a, end='')
186 
187         if self.tau_vary:
188             print('\ttau = %10.5f         '%self.tau)
189         else:
190             print('\ttau = %10.5f (fixed)'%self.tau)
191 
192     def fromJson(self,data):
193         self.a        = data["a"]["value"]
194         self.a_vary   = data["a"]["vary"]
195         self.tau      = data["tau"]["value"]
196         self.tau_vary = data["tau"]["vary"]
197 
198     def fromFit(self,i,result):
199         self.a   = abs(result.params['a'+str(i+1)].value)
200         self.tau = abs(result.params['tau'+str(i+1)].value)
201 
202 ##############################################################################
203 class LineClass:
204 ##############################################################################
205     def __init__(self):
206         self.slope = 0.0
207         self.slope_vary = True
208 
209     def info(self):
210         if self.slope_vary:
211             print('\t\tslope = %.8f         '%self.slope)
212         else:
213             print('\t\tslope = %.8f (fixed)'%self.slope)
214 
215     def fromJson(self,data):
216         self.slope      = data["slope"]["value"]
217         self.slope_vary = data["slope"]["vary"]
218 
219     def fromFit(self,result):
220         self.slope = abs(result.params['slope'].value)
221 
```

```
222 ################################################################################
223 class DataClass:
224 ################################################################################
225     def __init__(self):
226         self.time = []
227         self.f    = []
228         self.derivative = []
229 
230     def fromFile(self,filename,maxtime):
231         try:
232             infile = open(filename,'r')
233         except FileNotFoundError:
234             print('\n! Error!')
235             print('\t data file %s not found!'%filename)
236             sys.exit()
237 
238         tmp_time = []
239         tmp_f = []
240         for line in infile:
241             line_element = line.split()
242             current_time = float(line_element[0])
243             if current_time > maxtime:
244                 break
245             if len(line)>1:
246                 tmp_time.append(current_time)
247                 tmp_f.append(float(line_element[1]))
248         self.time = np.asarray(tmp_time)
249         self.f    = np.asarray(tmp_f)
250 
251     def compute_derivative(self):
252         ######################################################################
253         # "Numerische Methoden" J. Douglas Faiers / R.L. Burden, p. 168
254         ######################################################################
255         print('\n> Computing derivative of the time series ...')
256         h = self.time[1] - self.time[0]
257         tmp_derivative = []
258 
259         # end point calculation of the first two values of d/dt self.f
260         tmp = -2.0833333*self.f[0]+4.0*self.f[1]-3.0*self.f[2]+1.3333333*self.f[3]-0.25*
    self.f[4]
261         tmp_derivative.append(tmp/h)
262 
263         tmp = -0.25*self.f[0]-0.8333333*self.f[1]+1.5*self.f[2]-0.5*self.f[3]+0.08333333*
    self.f[4]
264         tmp_derivative.append(tmp/h)
265 
266         # mid point calculation of d/dt self.f
267         lenf = len(self.f)
268         for i in range(lenf-5):
269             tmp = 0.0833333*self.f[i]-0.666666*self.f[i+1]+0.666666*self.f[i+3]-0.0833333*
    self.f[i+4]
270             tmp_derivative.append(tmp/h)
271 
272         # end point calculation of the last values of d/dt self.f
273         tmp = -0.0833333*self.f[i]+0.5*self.f[i+1]-1.5*self.f[i+2]+0.8333333*self.f[i
    +3]+0.25*self.f[i+4]
274         tmp_derivative.append(tmp/h)
275 
276         tmp = 0.25*self.f[i]-1.333333*self.f[i+1]+3.0*self.f[i+2]-4.0*self.f[i
    +3]+2.08333333*self.f[i+4]
277         tmp_derivative.append(tmp/h)
278         tmp_derivative.append(0.0)
279         self.derivative = np.asarray(tmp_derivative)
280 
281 ################################################################################
282 # Fit model
283 ################################################################################
284 def residual(parfit, t, fdata=None, ddata = None):
285     model1 = 0.0
286     model2 = 0.0
287     for i in range(input.number_of_exp):
288         tmp_a   = parfit['a'+str(i+1)].value
289         tmp_tau = parfit['tau'+str(i+1)].value
290 
291         # correlation function
292         model1 += abs(tmp_a) * (1.0 - np.exp(-t/abs(tmp_tau)))
```

```
293
294         # its derivative
295         model2 += abs(tmp_a) / abs(tmp_tau) * np.exp(-t/abs(tmp_tau))
296
297     tmp_line = parfit['slope'].value
298     model1 += abs(tmp_line) * t
299     model2 += abs(tmp_line)
300
301     if fdata is None:
302         resid1 = model1
303     else:
304         resid1 = model1 - fdata
305
306     if ddata is None:
307         resid2 = model2
308     else:
309         resid2 = model2 - ddata
310     return np.concatenate(( resid1, resid2 ))
311
312 ##############################################################################
313 # Main program
314 ##############################################################################
315
316 # Generating input object
317 input = InputClass()
318 try:
319     JsonFile = sys.argv[1]
320     with open(JsonFile) as infile:
321         data = json.load(infile)
322     input.fromJson(data)
323     input.info()
324 except IndexError:
325     print('\n! Error!')
326     print('!\t Json input file is missing: python3 mdmd_fit.py ___.json. ')
327     print('!\t Writing example input msdMJ.json')
328     input.ExampleInput()
329     sys.exit()
330 except FileNotFoundError:
331     print('\n! Error!')
332     print('!\t Json input file %s not found!'%JsonFile)
333     sys.exit()
334
335 # Generating data object
336 data = DataClass()
337 data.fromFile(input.infile,input.maxtime)
338 data.compute_derivative()
339
340 # Generating fit model
341 print('\n> Fitting ...')
342
343 print('\t Setting up fit model ...')
344 parfit = Parameters()
345 parfit.add('slope', value = input.line.slope,   vary = input.line.slope_vary)
346 a0 = min(data.f[-1]-input.line.slope*data.time[-1],0.01*data.f[-1])
347 for i,exp in enumerate(input.exp):
348     parfit.add('a'+str(i+1),   value = exp.a, vary = exp.a_vary, min = a0, max = 0.9*data.
    f[-1])
349
350     if i>0:
351         min_tau = parfit['tau'+str(i)]*2.0
352     else:
353         min_tau = 0.1
354
355     max_tau = 0.3*float((i+1)/input.number_of_exp)*data.time[-1]
356     parfit.add('tau'+str(i+1), value = exp.tau, vary = exp.tau_vary, min = min_tau, max =
    max_tau)
357
358
359 print('\t Fitting data, minimizing residuals ...\n')
360 myfit  = Minimizer(residual, parfit,fcn_args=(data.time,), fcn_kws={'fdata': data.f, '
    ddata':data.derivative}, scale_covar=True)
361 result = myfit.minimize(input.fit_model)
362 fit    = residual(result.params, data.time)
363 resids = residual(result.params, data.time, data.f, data.derivative)
364 report_fit(result)
365 print('\n')
```

```
366
367 # update exp section in Json
368 for i,exp in enumerate(input.exp):
369     exp.fromFit(i,result)
370 input.line.fromFit(result)
371 input.toJson(JsonFile)
372
373 print('\n>\t Writing residual file = ',input.outfile)
374 f = open(input.outfile,'w')
375 fitlen = int(len(fit)/2)
376 for i in range(fitlen):
377     f.write("%10.5f "%data.time[i])
378     f.write("%10.5f "%data.f[i])
379     f.write("%10.5f "%fit[i])
380     f.write("%10.5f "%resids[i])
381     f.write("%10.5f "%data.derivative[i])
382     f.write("%10.5f "%fit[i+fitlen])
383     f.write("%10.5f "%resids[i+fitlen])
384     f.write("\n")
385
386 slope = input.line.slope
387 temp = 300
388 boxl = float(input.boxl)
389 print(f"\n>\tTemperature is set to {temp} K and box length (= {boxl} Angstrom) is taken
        from input file.")
390 conductivity = ( 1.602176634 * slope ) / (6 * 8.617332478 * 10**-8 * temp * boxl**3 )
391 cond2 =(1.602176634**2*slope)/(6*1.380649*boxl**3*temp*10**-7)
392 print(f"\n>\tConductivity is: {Decimal(conductivity):.10e} [S/m]")
393 print(f"\n>\tConductivity might also be: {Decimal(cond2):.10e} [S/m]")
394
395 J_split = JsonFile.split('/')[:-1] # path/path/file.json
396 con_path = ''
397 for i in J_split:
398     con_path = con_path + i + '/'
399 f1 = open(f'{con_path}conductivity.dat', "a")
400 f1.write(f"0.25\t{Decimal(conductivity):.10e}")
401 f1.write(f"[S/m]\t{JsonFile}\n")
402 f1.close()
403
404
405 #try:
406 #    with open(input.charmmfile, "r") as f:
407 #        lines = f.readlines()
408 #        for line in lines:
409 #            l = line.split()
410 #            if len(l) >= 3 and l[0] == "set" and l[1] == "XTL":
411 #                boxl = float(l[3])
412 #            if len(l) >= 7 and l[1] == "@?TEMP":
413 #                temp = float(l[7])
414 #    print(f"\n>\tExtracted temperature (= {temp} K) and box length (= {boxl} Angstrom)
        from file {input.charmmfile}.")
415 #except:
416 #    temp = 300
417 #    boxl = float(input.boxl)
418 #    print(f"\n>\tTemperature is set to {temp} K and box length (= {boxl} Angstrom) is
        taken from input file.")
419 #conductivity = ( 1.602176634 * slope ) / (6 * 8.617332478 * 10**-8 * temp * boxl**3 )
420 #print(f"\n>\tConductivity is: {Decimal(conductivity):.10e} [S/m]")
421 #    conductivity = ( 1.602176634 * slope ) / (6 * 8.617332478 * 10**-8 * temp * boxl**3 )
422 #    print(f"\n>\tConductivity is: {Decimal(conductivity):.10e} [S/m]")
423 #except FileNotFoundError:
424 #    print("No Charmm input file supplied or not found.")
425 #    print("No Conductivity was calculated.")
426 #    print("It can be calcualted with:")
427 #    print("conductivity = ( 1.602176634 * slope ) / (6 * 8.617332478 * 10**-8 * temp *
        boxl**3 )")
428 #    print("temp in K, boxl in Angstrom, conductivity is in S/m")
```

msdMJ_inp.json

```
{
  "data":      "msdmj/msdMJ_31-40_1_total.dat",
  "residuals": "msdmj/msdMJ_fit_31-40_1_total.dat",
  "charmmfile": "",
  "boxl": "0",
  "maxtime":   200,

  "initialexp":
  [
    { "a": {"value":   30.01885, "vary": true}, "tau": {"value":   16.80114, "vary": true} }
  ],

  "exp":
  [
    { "a": {"value":   30.01891, "vary": true}, "tau": {"value":   16.80125, "vary": true} }
  ],

  "initialline":
  [
    { "slope": {"value": 1.04337436, "vary": true} }
  ],

  "line":
  [
    { "slope": {"value": 1.04337397, "vary": true} }
  ]
}
```

write_gdc_inp_file.py

```
import json

number = "1"
outfile = f"GDC_inp_test_{number}.json"
molecules = ["nod", "lnk", "toh", "mer", "dfb"] #ions ignored
boxl= 101.52 #Angstrom
#molecules = ["all"]
maxtimes = []
with open("../mdmd/maxtimes.dat", "r") as f:
    for line in f.readlines():
        maxtimes.append(line.strip())

out = {}
out["correlations"] = {}
#plt.plot(freq_kcl025_e_tip3*1000, im_kcl01_e_tip3, label=r"$\varepsilon$ tip3", color=
    colordict[4])

#md0mdt
out["correlations"]["mdmd"] = []
for pos, residue in enumerate(molecules):
    #read necessary info from fit file
    path = f"../mdmd/MDMD_fit_{residue}_{maxtimes[pos]}.json"
    with open(path) as infile:
        data = json.load(infile)
#plt.plot(freq_kcl025_e_tip3*1000, im_kcl025_e_tip3, label=r"$\varepsilon$ tip3", color=
    colordict[4])
        maxtime = float(data["maxtime"])
        fits = []
        for exp in data["exp"]:
            fits.append({"id": "exp", "a": float(exp["a"]["value"]),
                         "tau": float(exp["tau"]["value"]),
                         "a0": 0.0})

    # build output json file
    molecule = {}
    molecule["key"] = str(residue)
    molecule["file"] = []
    molecule["file"].append({"in": f"../mdmd/md0mdt_{residue}.dat",
                             "out": f"mdmd_{residue}_{number}.dat",
                             "residuals": False,
```

```
                                "bspline": 10,
                                "maxtime": maxtime, "expdamping": 0.2 })
    molecule["fitfunction"] = fits

    out["correlations"]["mdmd"].append(molecule)

################################################################################
#delta mj2
######################### FOR CHARGED MOLECULES ################################
#out["correlations"]["deltamj2"] = []
#path = "../msdmj/msdMJ_inp.json"
#with open(path) as infile:
#    #plt.plot(freq_kcl025_e_tip3*1000, im_kcl025_e_tip3, label=r"$\varepsilon$ tip3",
    color=colordict[4])
#    data = json.load(infile)
#    boxl = float(data["boxl"])
#    maxtime = float(data["maxtime"])
#    fits = []
#    for exp in data["exp"]:
#        fits.append({"id": "exp", "a": float(exp["a"]["value"]),
#                     "tau": float(exp["tau"]["value"]),
#                     "a0": 0.0})
#
#molecule = {}
#molecule["key"] = "all"
#molecule["file"] = []
#molecule["file"].append({"in": f"../msdmj/msdMJ_31-40_1_total.dat",
#                         "out": f"mdmj_all_{number}.dat",
#                         "residuals": True,
#                         "bspline": 10,
#                         "maxtime": maxtime, "expdamping": 0.2 })
#molecule["fitfunction"] = fits
#
#out["correlations"]["deltamj2"].append(molecule)
################################################################################
#spectrum
out["spectrum"] = {}
out["spectrum"]["prefactor"] = [{"temperature": 300, "boxlength":boxl, "boxtype": "cubic"
    }]
out["spectrum"]["frequency"] = [{"type": "nue", "unit": "THz", "Min": 0.00001, "Max":
    50.0, "logscale": True}]
epsilon = []
for residue in molecules:
    epsilon.append({ "key": residue,
           "out": f"epsilon_{residue}_{number}.dat",
           "mdmd":[ f"{residue}" ],
           "smoothing": True })
epsilon.append({"key": "all",
              "out": f"epsilon_all_{number}.dat",
              "mdmd": molecules,
              "smoothing": True })
out["spectrum"]["epsilon"] = epsilon

##### THETA MJ
#out["spectrum"]["theta0"] = [
#        { "key": "all",
#          "out": f"theta_{number}.dat",
#          "deltamj2":[ "all" ],
#          "smoothing": True} ]
#####

out["spectrum"]["gendicon"] = [
         { "key": "all",
           "out": f"gd_all_{number}.dat",
           "smoothing": True} ]


with open(outfile, "w") as outf:
    json.dump(out, outf, indent=4)
```

poreVSbulk.py

```
import MDAnalysis
from newanalysis.functions import atomsPerResidue, residueFirstAtom
from newanalysis.functions import centerOfMassByResidue, dipoleMomentByResidue
from newanalysis.diffusion import msdCOM, unfoldTraj, msdMJ, msdMJdecomp, msdMJcrossterms
import MDAnalysis.transformations as trans
import numpy as np
import os, sys
import time
from datetime import timedelta
from pathlib import Path
import re
from scipy import stats

################################################################################
# Trajectory
################################################################################
firstfile=31
lastfile=40
skip=1
#################################################################
base = "/site/raid6/johnny/mof/sim10_nu1t-spiroII/ions/TBABr_10/m123/rep2/"
psf_path = base + "run.psf"
dcd_path = base + "traj/nvt_"
savefolder = './PVB'
msdfolder = "./PVB/msd"
#################################################################
residues = ["NOD", "LNK", "TOH","MER", "DFB", "BRO", "TBA"]
mof_res = ["NOD", "LNK", "TOH"]
spi_res = ["MER"]
################################################################# lin reg.
lin_max = 600

################################################################################
################## DEFINITIONS #################################################
################################################################################
def msd_com(com_array):
    """
    Calculates mean squared displacement using the Fast Correlation Algorithm in the
    package tidynamics
    Dependency: tidynamics
    input: numpy array with shape (timesteps, particles, xyz)
    returns: numpy array with shape (timestep, msd)
    """
    import tidynamics

    com_array = com_array.astype(np.float64)
    time, n_particles, ndim = com_array.shape
    msds_by_particle = np.zeros((time, n_particles))
    for particle in range(n_particles):
        msds_by_particle[:, particle] = tidynamics.msd(com_array[:, particle, :])
    msd = msds_by_particle.mean(axis=1)
    return msd

################################################################################
################## MAIN PROGRAM ################################################
################################################################################
print(f"Using dcd files from {firstfile} to {lastfile} with skip {skip}")
Path(savefolder).mkdir(parents=True, exist_ok=True)
Path(msdfolder).mkdir(parents=True, exist_ok=True)

Din = {}
Dout = {}
Dham = {}
for residue in residues:
    Din[residue] = []
    Dout[residue] = []
    Dham[residue] = []

for i in range(firstfile,lastfile+1):
    print("Checking for psf & dcd")
    dcd = f'{dcd_path}{i}.dcd'
    print("PSF file:", psf_path)
    print("DCD file:", dcd)

```

```
74      u=MDAnalysis.Universe(psf_path,dcd)
75      boxl = np.float64(round(u.coord.dimensions[0],4))
76      dt=round(u.trajectory.dt,4)
77
78      n = int(u.trajectory.n_frames/skip)
79      if u.trajectory.n_frames%skip != 0:
80          n+=1
81      print(f"Boxlength is {boxl}")
82
83      ###############################################################################
84      # Running through trajectory
85      ###############################################################################
86      ctr=0
87      start=time.time()
88
89      pore=[] #put resid here
90      not_pore=[]
91      hamletA3S1=[] #[...] in/out the pore
92
93      #umof = MDAnalysis.Merge(u.select_atoms("resname NOD and resname LNK and resname TOH")
        )
94
95      for ts in u.trajectory[::skip]:
96          print("\033[1AFrame %d of %d" % (ts.frame,len(u.trajectory)), "\tElapsed time: %.2
        f hours" % ((time.time()-start)/3600))
97      ##############
98          sel_mof = u.select_atoms("resname NOD or resname LNK or resname TOH")
99          all_atoms = u.select_atoms("all")
100         #center = sel_mof.center_of_geometry(wrap = True)
101         #dim = ts.triclinic_dimensions
102         #box_center = np.sum(dim, axis=0) / 2
103         #u.atoms.translate(box_center - center)
104         in_atoms = u.select_atoms("byres (sphzone 15 group sel_mof)", sel_mof=sel_mof)
105         out_atoms = all_atoms - in_atoms
106
107         for r_ID in in_atoms.resids:
108             if r_ID not in pore:
109                 pore.append(r_ID)
110             else:
111                 pass
112
113         for r_ID in out_atoms.resids:
114             if r_ID not in not_pore:
115                 not_pore.append(r_ID)
116             else:
117                 pass
118         ctr+=1
119     print(f"Time elapsed: {time.time()-start}")
120
121     for r_ID in pore:
122         if r_ID in not_pore:
123             hamletA3S1.append(r_ID)
124         else:
125             pass
126     for r_ID in hamletA3S1:
127         if r_ID in pore and r_ID in not_pore:
128             pore.remove(r_ID)
129             not_pore.remove(r_ID)
130         else:
131             print('How')
132             print('Lol what')
133             sys.exit()
134
135     ##############
136     all_in = u.select_atoms(None)
137     all_out = u.select_atoms(None)
138     all_hamlet = u.select_atoms(None)
139
140     for r_ID in pore:
141         all_in += u.select_atoms(f'resid {r_ID}')
142     for r_ID in not_pore:
143         all_out += u.select_atoms(f'resid {r_ID}')
144     for r_ID in hamletA3S1:
145         all_hamlet += u.select_atoms(f'resid {r_ID}')
146
147     #with MDAnalysis.Writer("dcd/IN.dcd", all_in.n_atoms) as W:
```

```
148     #    for ts in u.trajectory:
149     #        W.write(all_in)
150     #with MDAnalysis.Writer("dcd/OUT.dcd", all_out.n_atoms) as W:
151     #    for ts in u.trajectory:
152     #        W.write(all_out)
153     #with MDAnalysis.Writer("dcd/HAM.dcd", all_hamlet.n_atoms) as W:
154     #    for ts in u.trajectory:
155     #        W.write(all_hamlet)
156     #with MDAnalysis.Writer(f"pdb/IN.pdb", all_in.n_atoms) as W:
157     #    W.write(all_in)
158     #with MDAnalysis.Writer(f"pdb/OUT.pdb", all_out.n_atoms) as W:
159     #    W.write(all_out)
160     #with MDAnalysis.Writer(f"pdb/HAM.pdb", all_hamlet.n_atoms) as W:
161     #    W.write(all_hamlet)
162
163     ###############################################################################
164     # molecular species
165     ###############################################################################
166     in_sel = {}
167     in_nres = {}
168     in_coms = {}
169     out_sel = {}
170     out_nres = {}
171     out_coms = {}
172     ham_sel = {}
173     ham_nres = {}
174     ham_coms = {}
175
176     for residue in residues:
177         in_sel[residue] = all_in.select_atoms(f"resname {residue}")
178         in_nres[residue] = in_sel[residue].n_residues
179         in_coms[residue] = np.zeros((n,in_nres[residue],3), dtype=np.float64)
180         out_sel[residue] = all_out.select_atoms(f"resname {residue}")
181         out_nres[residue] = out_sel[residue].n_residues
182         out_coms[residue] = np.zeros((n,out_nres[residue],3), dtype=np.float64)
183         ham_sel[residue] = all_hamlet.select_atoms(f"resname {residue}")
184         ham_nres[residue] = ham_sel[residue].n_residues
185         ham_coms[residue] = np.zeros((n,ham_nres[residue],3), dtype=np.float64)
186         print(f"Number of {residue} inside pore:   {in_nres[residue]}")
187         print(f"Number of {residue} outside pore:  {out_nres[residue]}")
188         print(f"Number of {residue} visiting pore: {ham_nres[residue]}")
189
190     ctr=0
191     for ts in u.trajectory[::skip]:
192         print("\033[1AFrame %d of %d" % (ts.frame,len(u.trajectory)), "\tElapsed time: %.2
    f hours" % ((time.time()-start)/3600))
193         for residue in residues:
194             if in_nres[residue] == 0:
195                 pass
196             else:
197                 in_coms[residue][ctr] = in_sel[residue].center_of_mass(compound="residues"
    )
198             if out_nres[residue] == 0:
199                 pass
200             else:
201                 out_coms[residue][ctr] = out_sel[residue].center_of_mass(compound="
    residues")
202             if ham_nres[residue] == 0:
203                 pass
204             else:
205                 ham_coms[residue][ctr] = ham_sel[residue].center_of_mass(compound="
    residues")
206         ctr+=1
207
208     print("calculating msd ..")
209     time_axis = np.arange(0, n*skip*dt, skip*dt)
210     for residue in residues:
211         in_msd = msd_com(in_coms[residue])
212         out_msd = msd_com(out_coms[residue])
213         ham_msd = msd_com(ham_coms[residue])
214         np.savetxt(f"{msdfolder}/msd_IN_{residue.lower()}_{i}.dat", np.c_[time_axis,
    in_msd])
215         msd = np.loadtxt(f'{msdfolder}/msd_IN_{residue.lower()}_{i}.dat')
216         res = stats.linregress(msd[:int(lin_max),0], msd[:int(lin_max),1])
217         Din[residue].append(1/6 * res.slope)
```

```
218         np.savetxt(f"{msdfolder}/msd_OUT_{residue.lower()}_{i}.dat", np.c_[time_axis,
        out_msd])
219         msd = np.loadtxt(f'{msdfolder}/msd_OUT_{residue.lower()}_{i}.dat')
220         res = stats.linregress(msd[:int(lin_max),0], msd[:int(lin_max),1])
221         Dout[residue].append(1/6 * res.slope)
222         np.savetxt(f"{msdfolder}/msd_HAM_{residue.lower()}_{i}.dat", np.c_[time_axis,
        ham_msd])
223         msd = np.loadtxt(f'{msdfolder}/msd_HAM_{residue.lower()}_{i}.dat')
224         res = stats.linregress(msd[:int(lin_max),0], msd[:int(lin_max),1])
225         Dham[residue].append(1/6 * res.slope)
226
227 print(Din)
228 print(Dout)
229 print(Dham)
230 outs = [(f'{savefolder}/D_data_in.dat',Din), (f'{savefolder}/D_data_out.dat',Dout), (f'{
        savefolder}/D_data_ham.dat',Dham)]
231 for out in outs:
232     with open(out[0], 'w') as file:
233         for i in out[1]:
234             L = out[1][i]
235             file.write(f'{i}    {L}\n')
236 outs = [(f'{savefolder}/D_in.dat',Din), (f'{savefolder}/D_out.dat',Dout), (f'{savefolder}/
        D_ham.dat',Dham)]
237 for out in outs:
238     with open(out[0], 'w') as file:
239         for i in out[1]:
240             L = out[1][i]
241             file.write(f'{i}    {sum(L)/float(len(L))}\n')
242
243 #sys.exit()
```

D | Appendix - Replica Results

D.1. Infrared Spectra

D.1.1. Bulk DFB

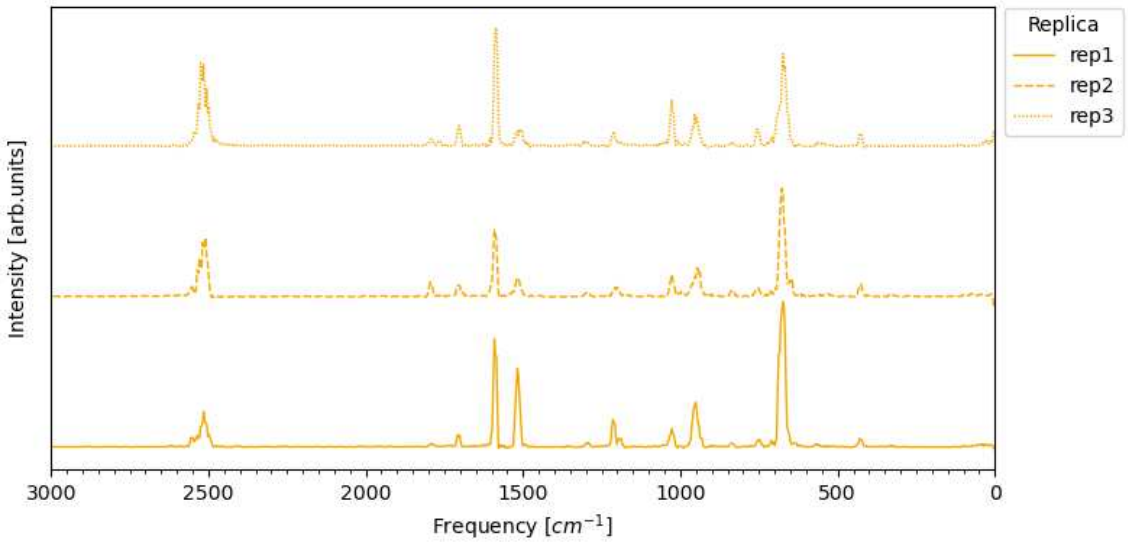


Figure D.1.: IR Spectrum of bulk DFB.

D.1.2. Unmodified NU-1000

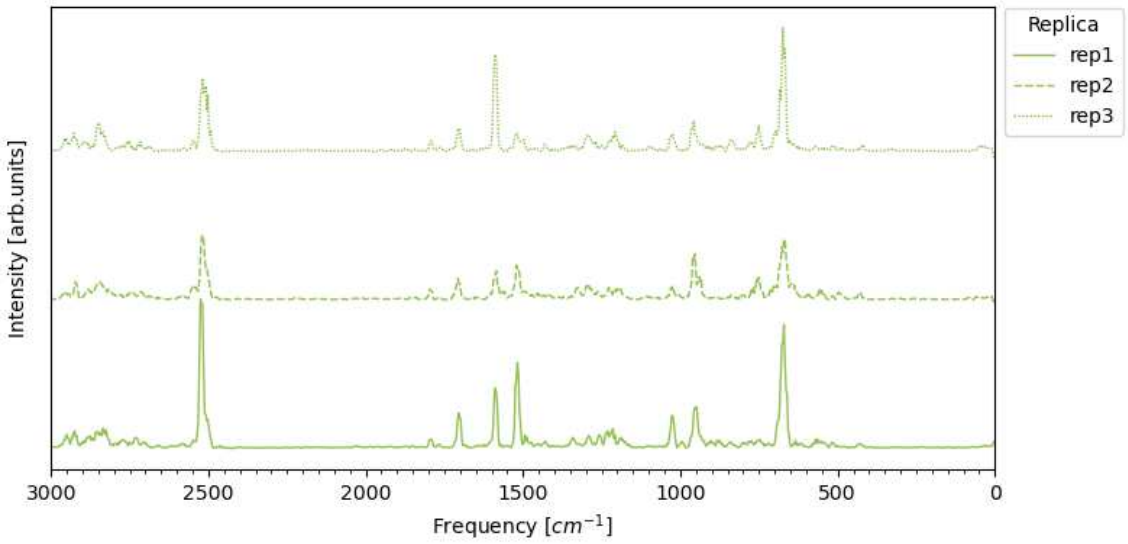


Figure D.2.: IR Spectrum of unmodified NU-1000 in DFB.

D.1.3. Spiropyran functionalized NU-1000

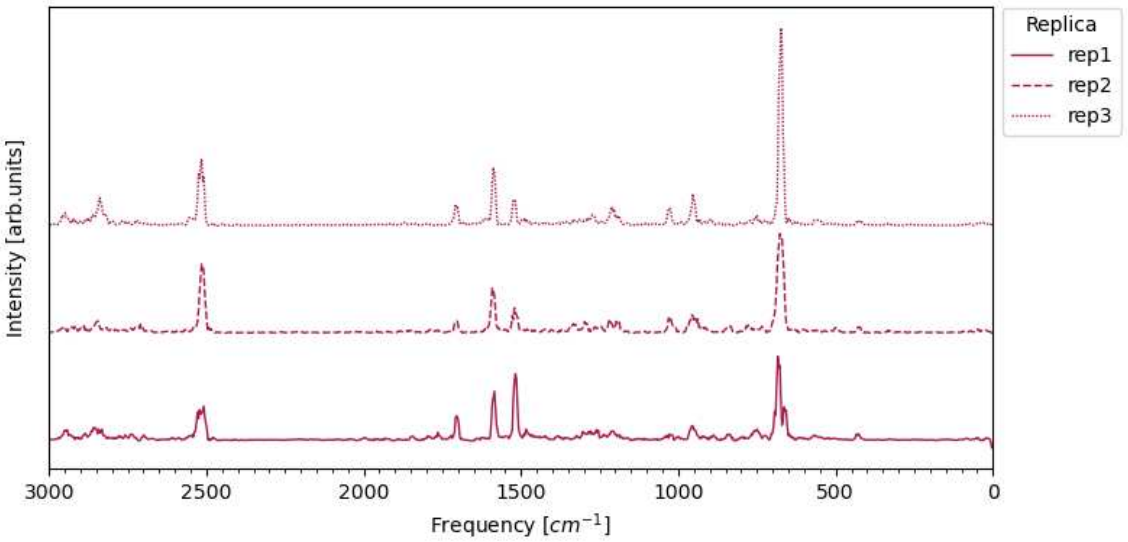


Figure D.3.: IR Spectra of s123.

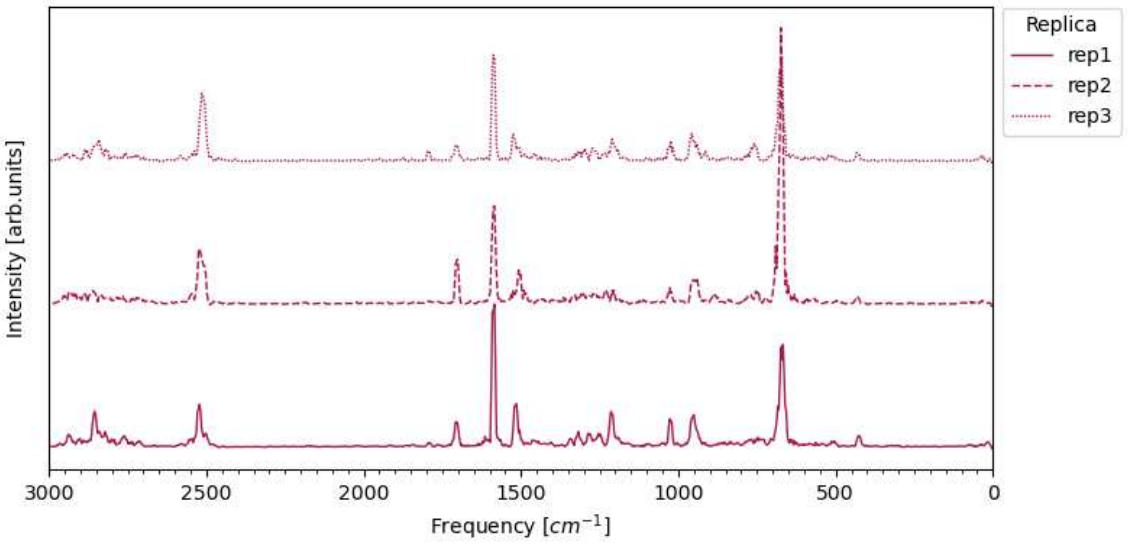


Figure D.4.: IR Spectra of s124.

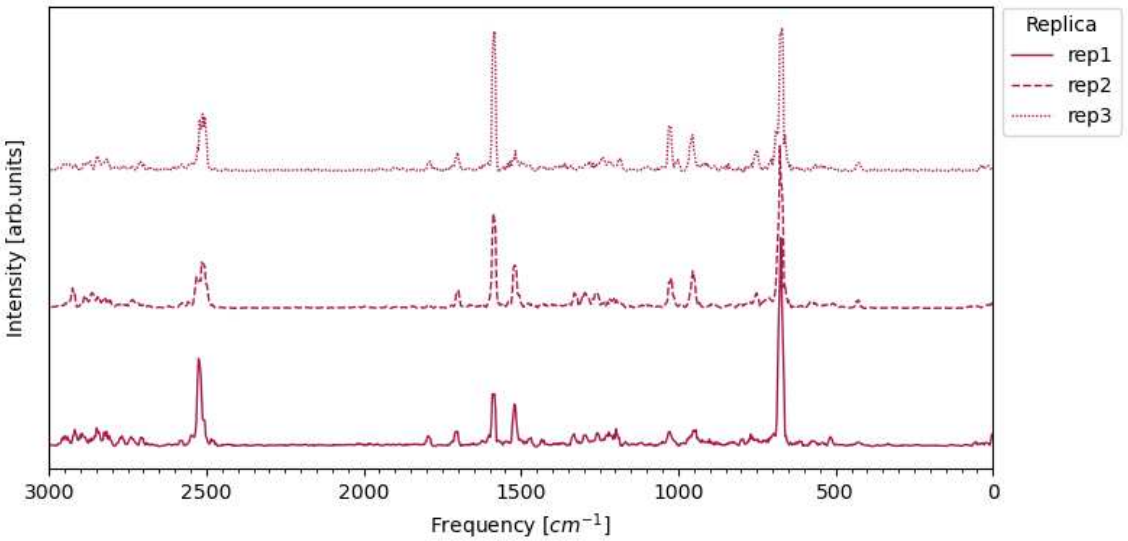


Figure D.5.: IR Spectra of s135.

D.1.4. Merocyanine functionalized NU-1000

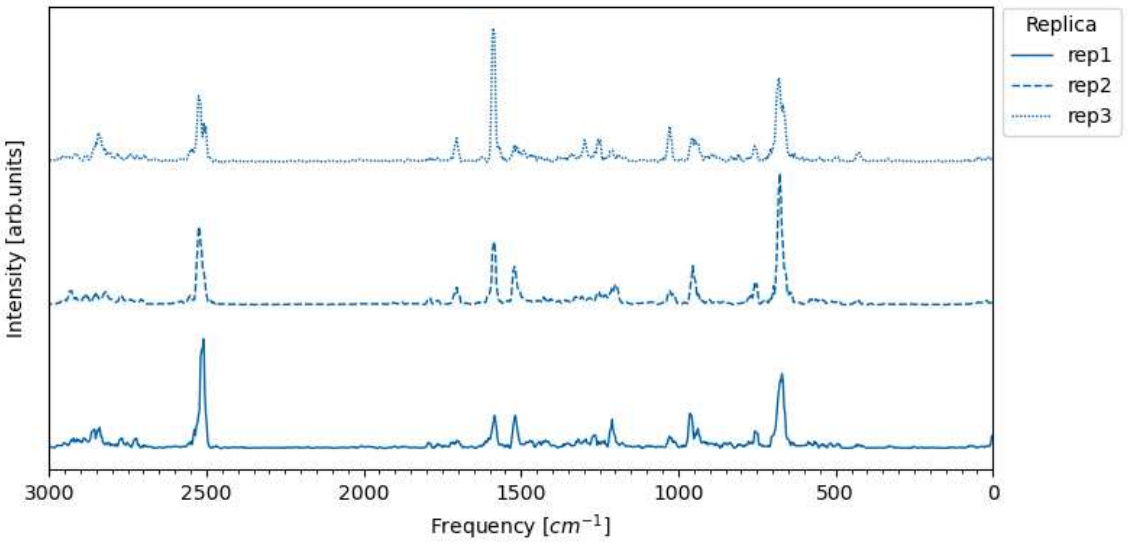


Figure D.6.: IR Spectra of m123.

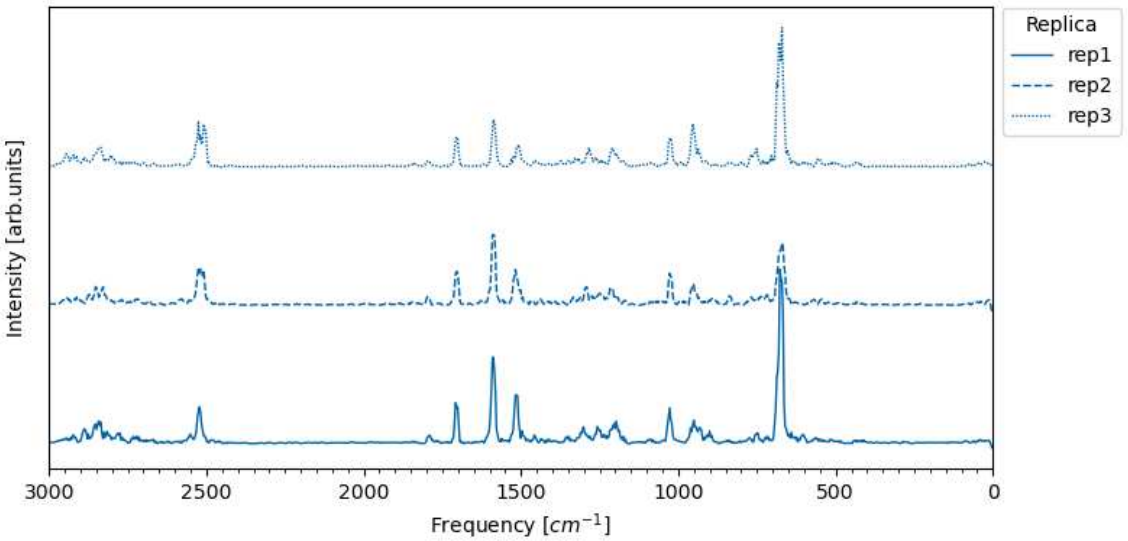


Figure D.7.: IR Spectra of m124.

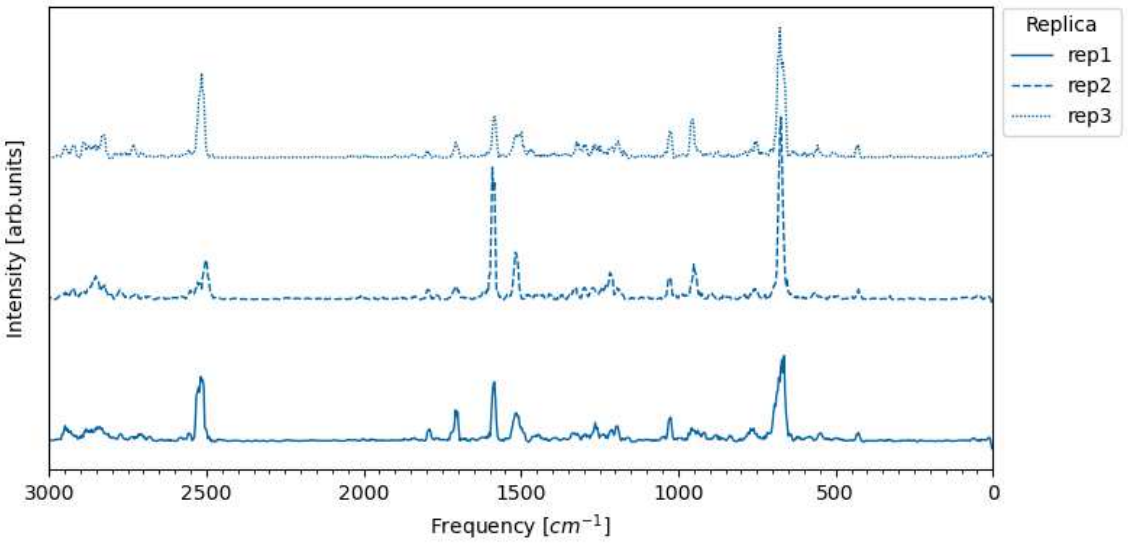
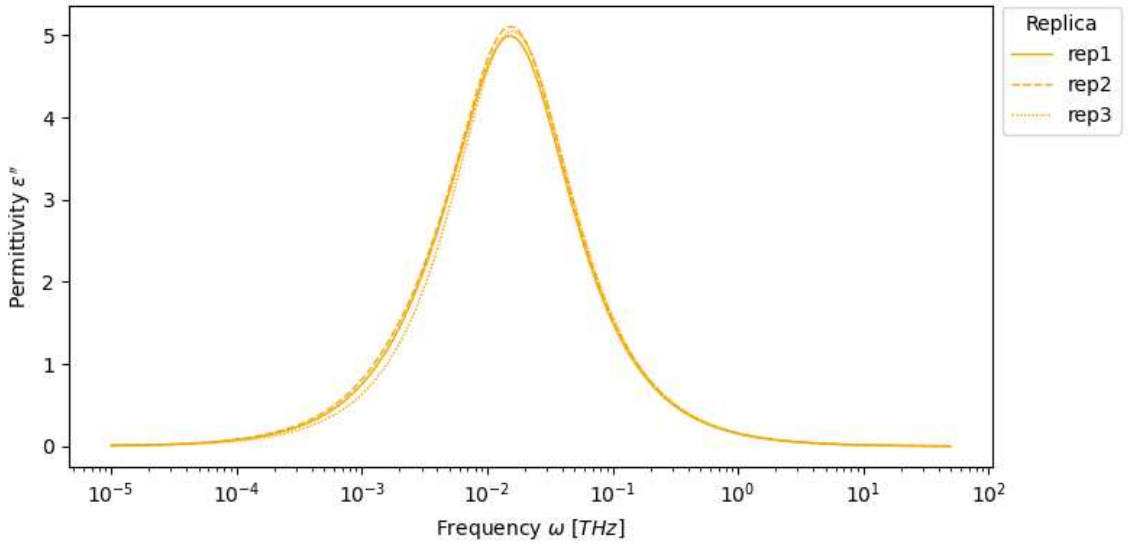


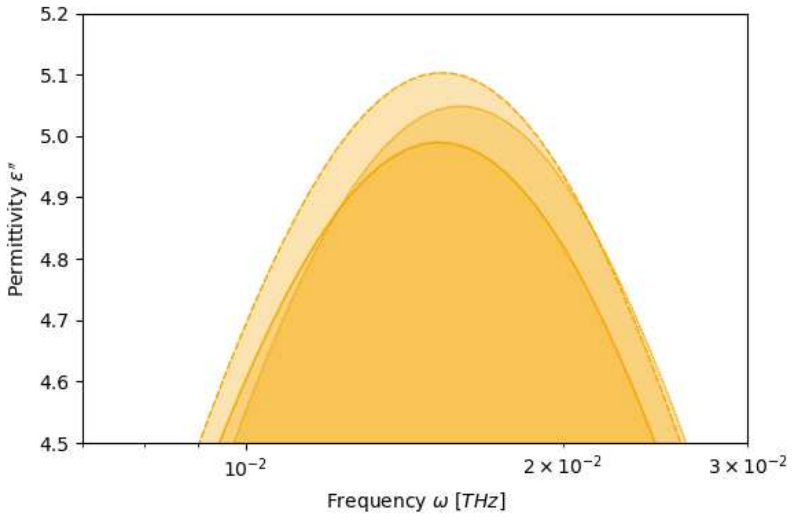
Figure D.8.: IR Spectra of m135.

D.2. Dielectric Spectra

D.2.1. Bulk DFB



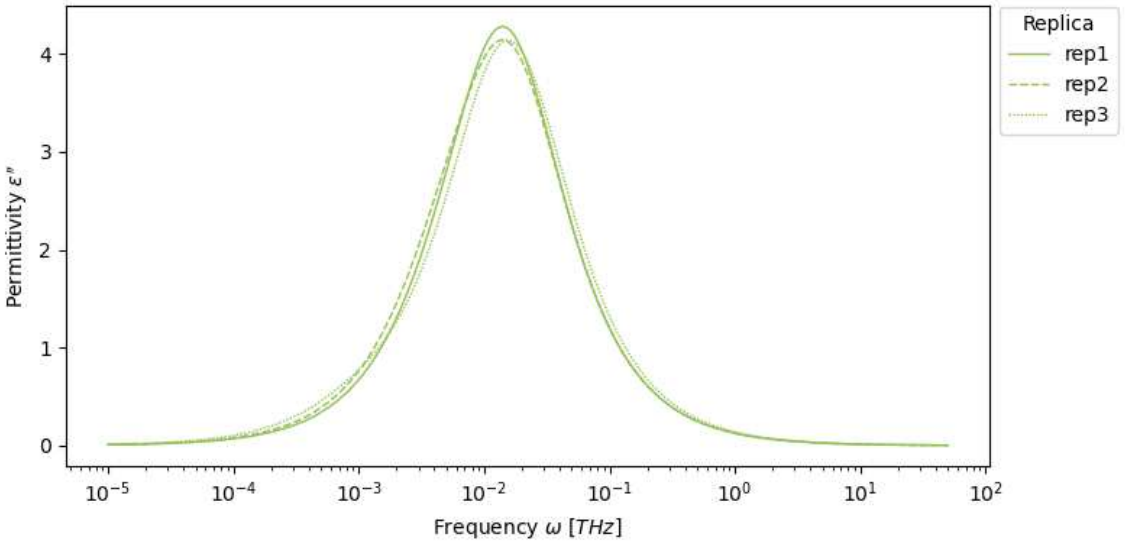
(a) Full dielectric spectra.



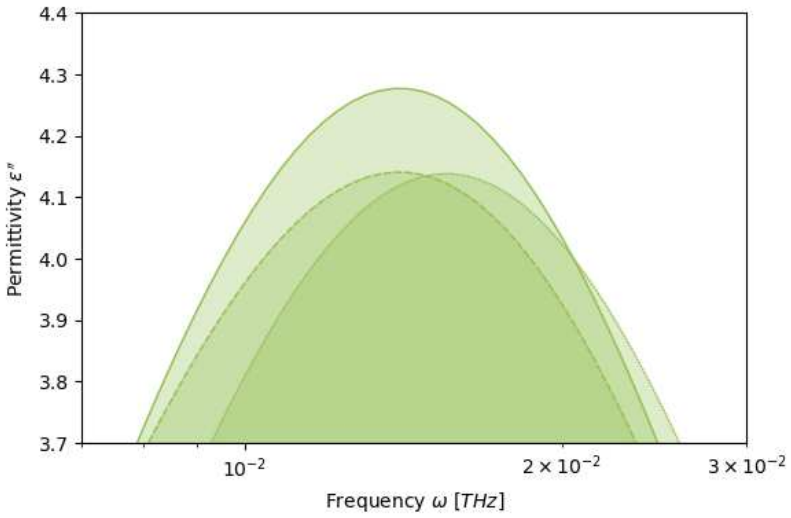
(b) Closer view of the peak maxima.

Figure D.9.: The imaginary part of the dielectric permittivity ϵ'' of all bulk DFB simulation replicas.

D.2.2. NU-1000



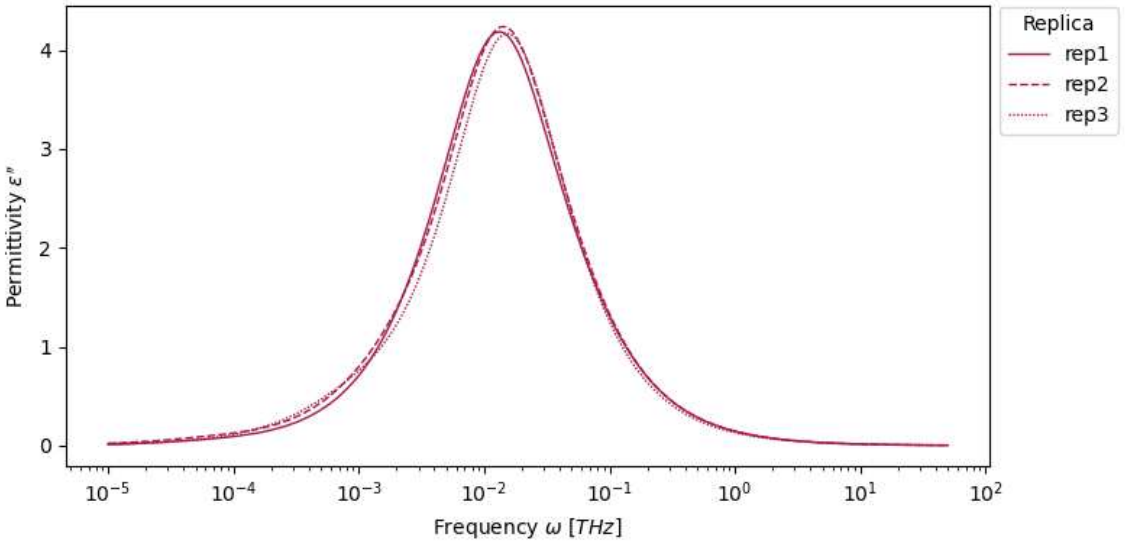
(a) Full dielectric spectra.



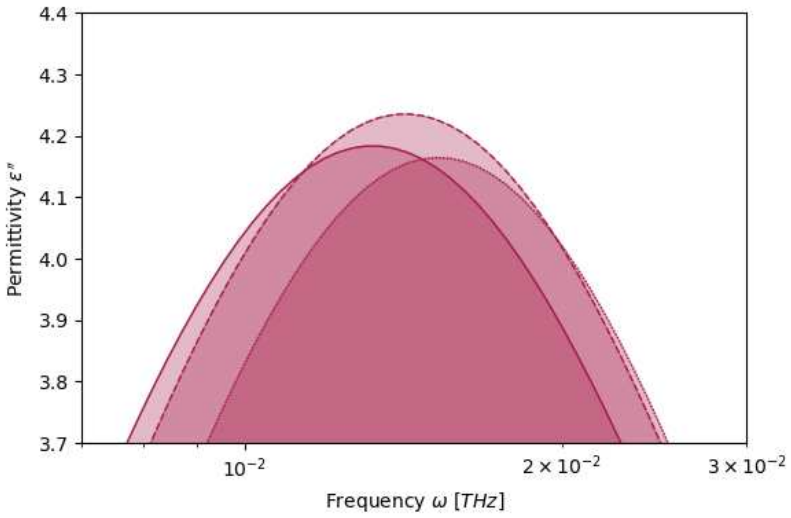
(b) Closer view of the peak maxima.

Figure D.10.: The imaginary part of the dielectric permittivity ϵ'' of all non-functionalized NU-1000 simulation replicas.

D.2.3. Spiropyran

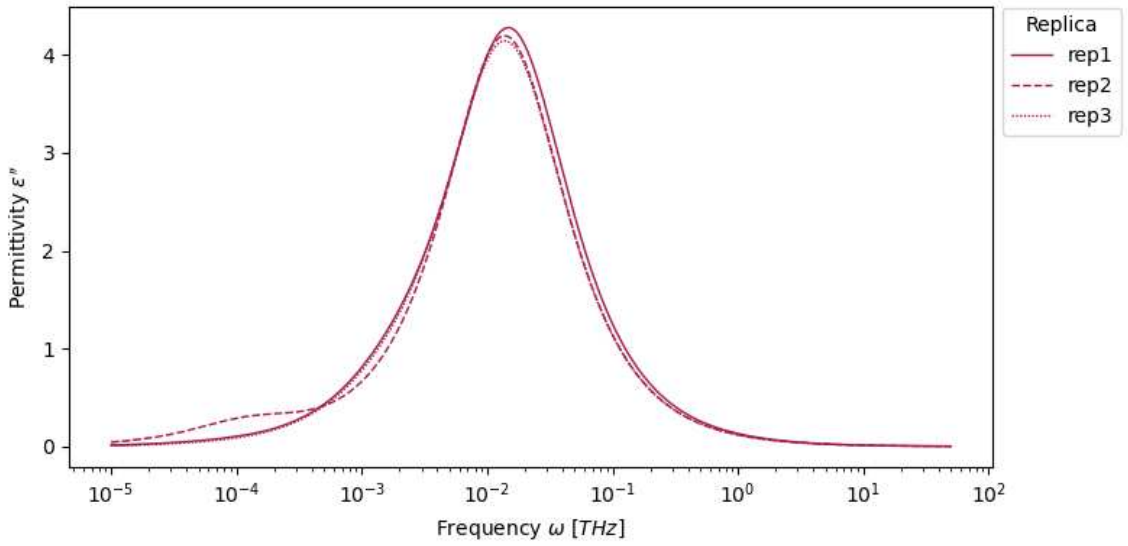


(a) Full dielectric spectra.

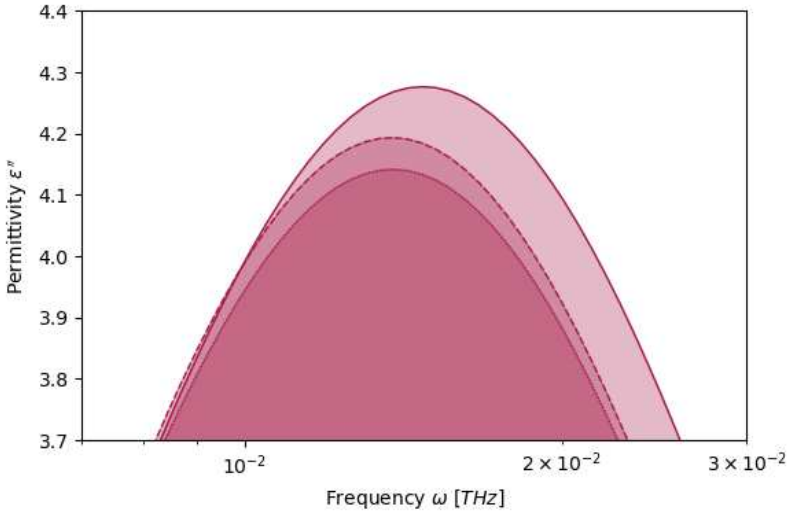


(b) Closer view of the peak maxima.

Figure D.11.: The imaginary part of the dielectric permittivity ϵ'' of all spiropyran-functionalized NU-1000 simulation replicas in 123 configuration.

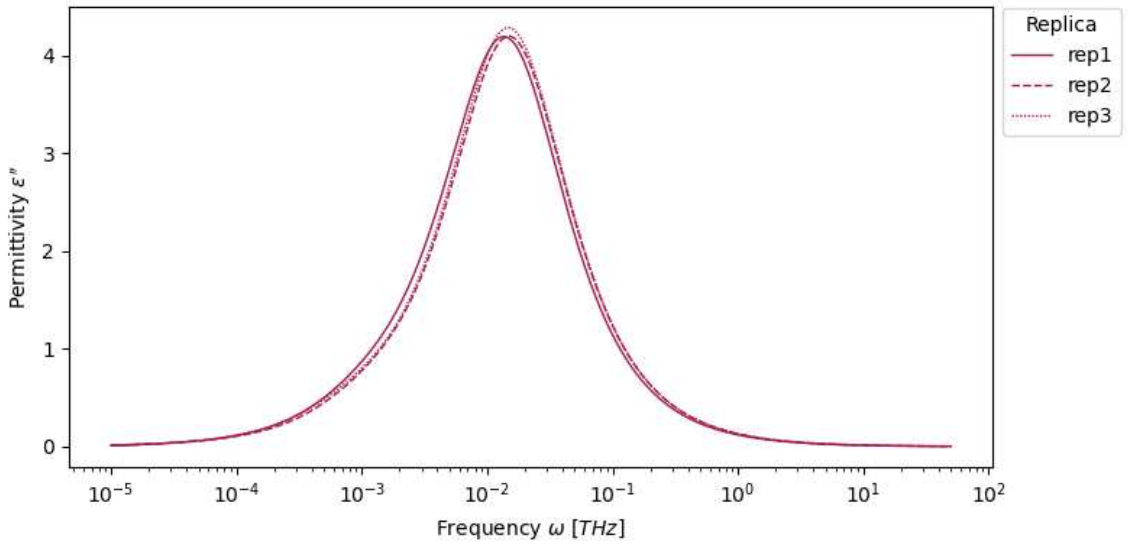


(a) Full dielectric spectra.

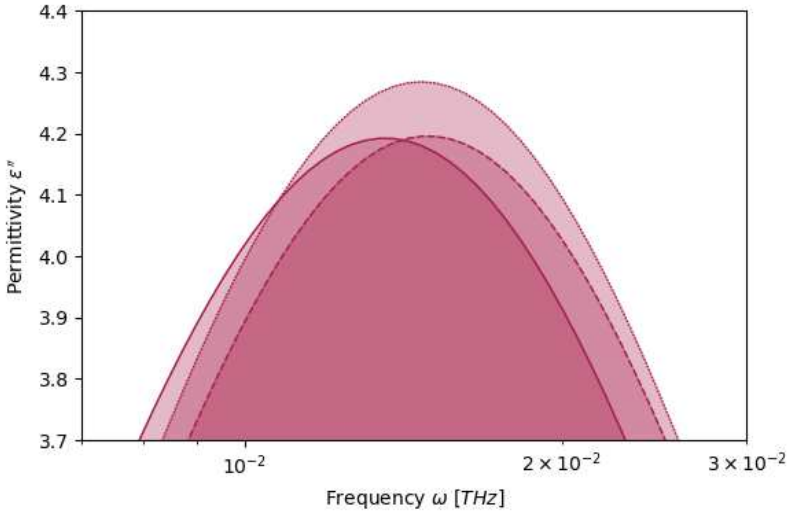


(b) Closer view of the peak maxima.

Figure D.12.: The imaginary part of the dielectric permittivity ϵ'' of all spiropyran-functionalized NU-1000 simulation replicas in 124 configuration.



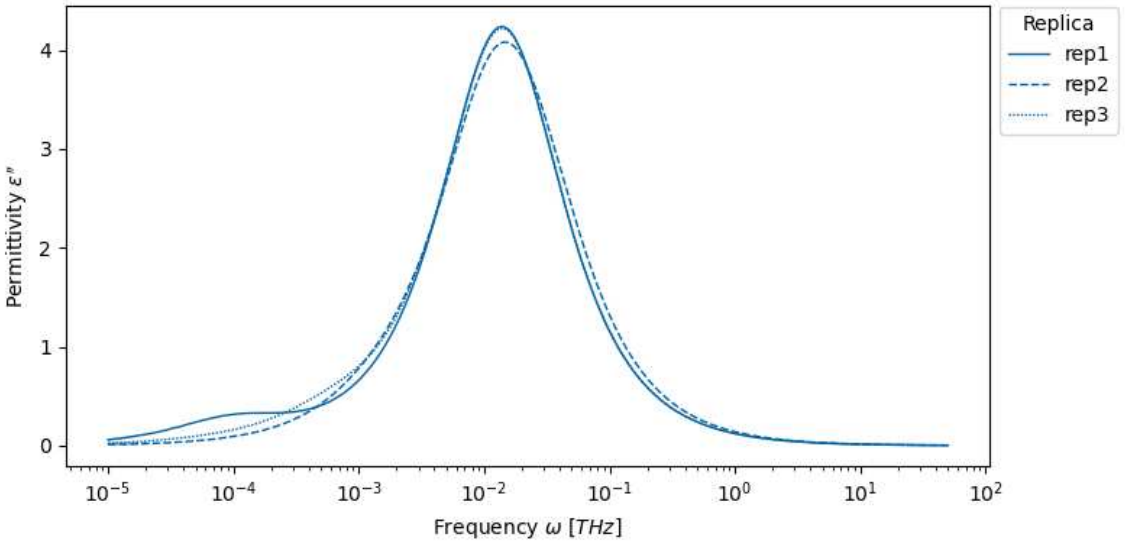
(a) Full dielectric spectra.



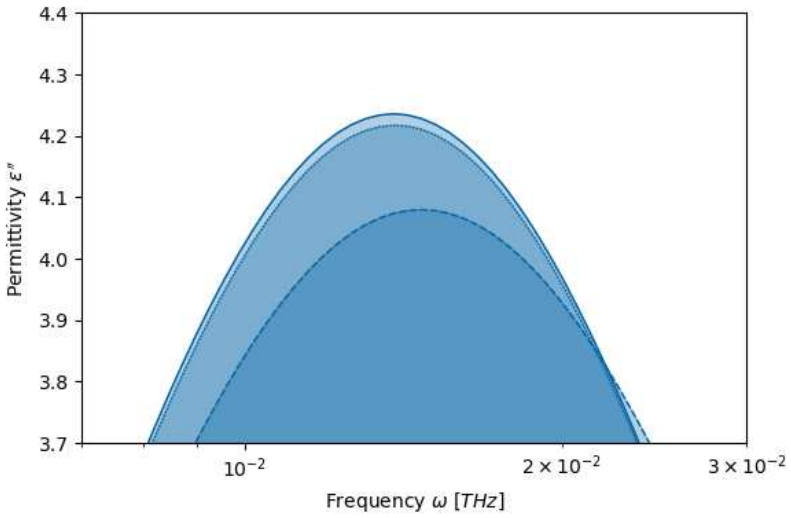
(b) Closer view of the peak maxima.

Figure D.13.: The imaginary part of the dielectric permittivity ϵ'' of all spiropyran-functionalized NU-1000 simulation replicas in 135 configuration.

D.2.4. Merocyanine

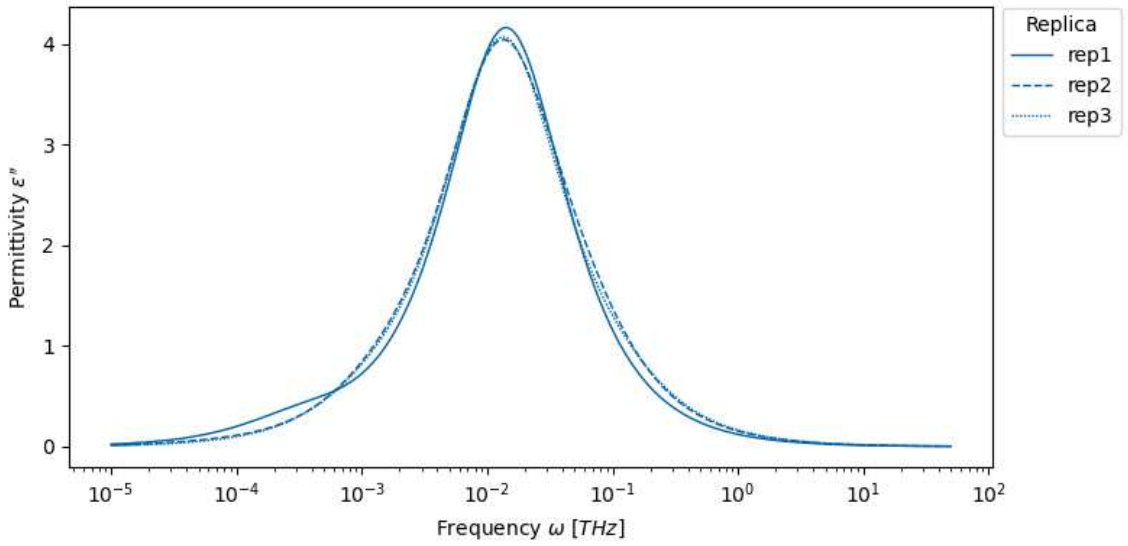


(a) Full dielectric spectra.

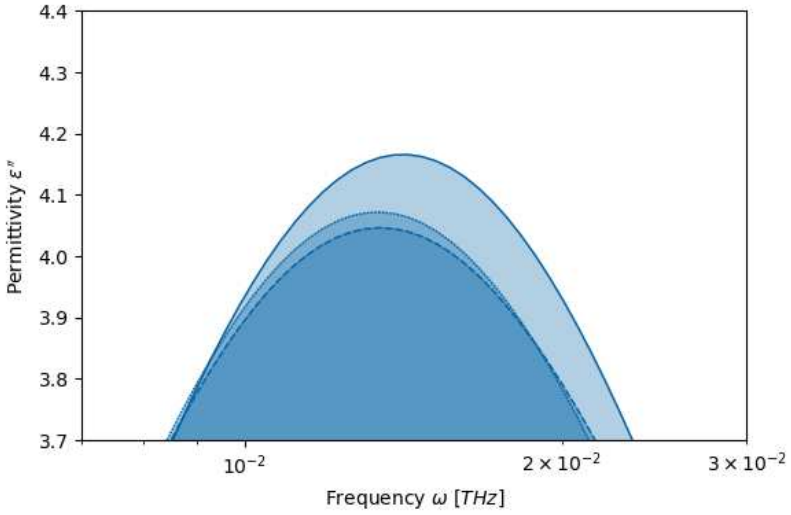


(b) Closer view of the peak maxima.

Figure D.14.: The imaginary part of the dielectric permittivity ϵ'' of all merocyanine-functionalized NU-1000 simulation replicas in 123 configuration.

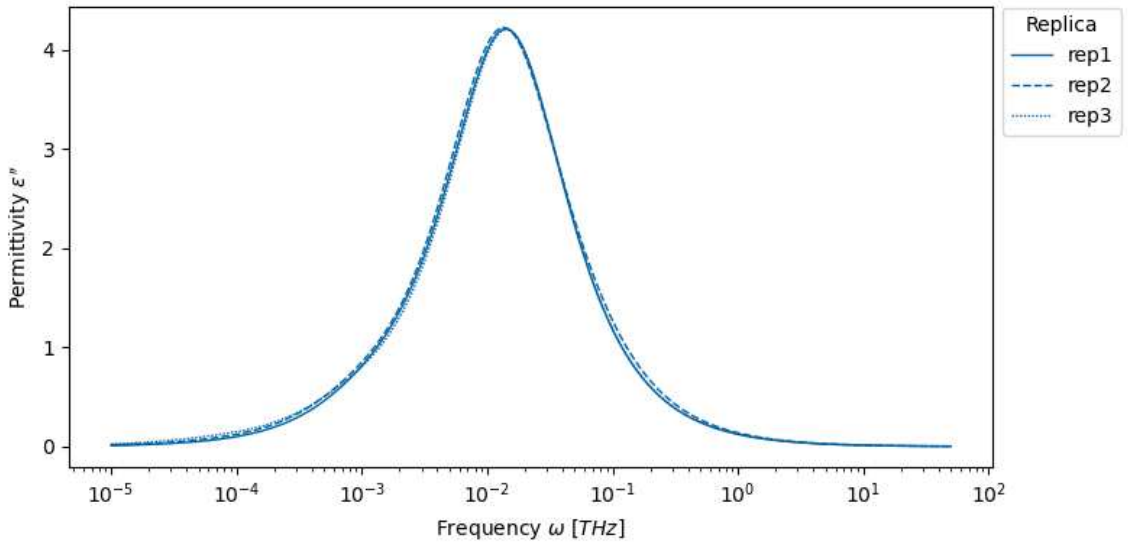


(a) Full dielectric spectra.

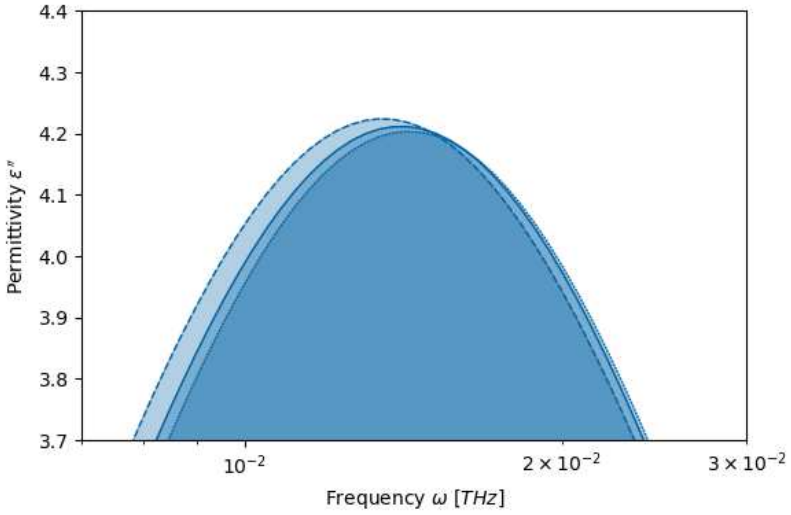


(b) Closer view of the peak maxima.

Figure D.15.: The imaginary part of the dielectric permittivity ϵ'' of all merocyanine-functionalized NU-1000 simulation replicas in 124 configuration.



(a) Full dielectric spectra.



(b) Closer view of the peak maxima.

Figure D.16.: The imaginary part of the dielectric permittivity ϵ'' of all merocyanine-functionalized NU-1000 simulation replicas in 135 configuration.