Uppsala University

Department of Physics and Astronomy

Master's thesis 30 ECTS

Polarizability and Orientation Dynamics of Small Proteins

Ebba Koerfer

Supervisors: Oscar Grånäs and Carl Caleman
Subject reader: Nicusor Timneanu
Examiner: Andreas Korn

July 6, 2022

Abstract

Proteins often carry an intrinsic electric dipole moment, which can interact with external electric fields and cause protein motion. Previous research has found that the orientation of small proteins in gas phase can be controlled in a static electric field. This effect is hoped to benefit applications such as single-particle imaging, and possibly other techniques involving proteins in electric fields. With the purpose of improving our understanding and modeling of protein orientation, this project investigated the scarcely explored quantum mechanical aspects of the process, namely the polarizability. Ground-state electronic structure simulations of three small model proteins, ubiquitin, Trp-cage and lysozyme, under the influence of electric fields were performed in vacuum. The electric dipole moments of the proteins were extracted from simulations with an applied electric field of strength 1 V/nm for varying angles, with respect to a body fixed reference frame. A Python program was written to analyze and visualize the results. The results point to a connection between the polarizability and the structure of the proteins, as well as size. Next a 3D rigid rotor model was developed using Mathematica in order to study the orientation dynamics classically in a simplified and time efficient way, with the possibility of including the previous quantum results. A comparison between a simulation of ubiquitin with and without polarizability concluded that the polarizability seems to have a damping effect on the orientation dynamics, at least for the initial conditions tested in this study. Further research is necessary to validate the model and perform statistical analysis of many simulations with varying initial conditions.

Sammanfattning

Proteiner bär ofta på ett inneboende elektriskt dipolmoment, som vid interaktion med externa elektriska fält och orsakar rörelse hos proteinerna. Tidigare studier har funnit att orienteringen av små proteiner i gasfas kan kontrolleras i ett statiskt elektriskt fält. Den effekten kan förhoppningsvis vara en fördel i tillämpningar såsom single-particle imaging, och eventuellt andra tekniker som innefattar proteiner i elektriska fält. I syftet att förbättra vår förståelse och modellering av protein-orientering, har detta projekt undersökt de föga utforskade kvantmekaniska aspekterna av processen, nämligen polariserbarheten. Kvant-baserade simuleringar av grundtillståndet av tre små proteiner, ubiquitin, Trp-cage och lysozym, under påverkan av elektriska fält utfördes i vakuum. Proteinernas elektriska dipolmoment extraherades från simuleringar med ett elektriskt fält med styrkan 1 V/nm för olika vinklar, med avseende på ett kroppsfixerat koordinatsystem. Ett Python-program skrevs för att analysera och visualisera resultaten. Resultaten tyder på att polariserbarheten beror på strukturen och storleken av proteinerna. Därefter utformades en stel-rotor-modell med hjälp av Mathematica för att studera prienteringen klassiskt på ett förenklat och tidseffektivt sätt, med möjligheten att inkludera de tidigare kvantmekaniska resultaten. En jämförelse mellan en simulering av ubiquitin med och utan polariserbarhet konstaterade att polariserbarheten verkar ha en dämpande effekt på orienteringen, åtminstone för begynnelsevillkoren som testades i denna studie. Vidare forskning krävs för att styrka modellen och utföra statistisk analys av många simuleringar med varierande begynnelsevillkor.

Contents

Abbreviations

XFEL	X-ray free-electron laser
SPI	Single-particle imaging
MD	Molecular dynamics
PDB	Protein data bank
DFT	Density functional theory
XC	Exchange-correlation
LDA	Local density approximation
GGA	Generalized gradient approximation
PBE	Perdew-Burke-Ernzerhof
SIESTA	Spanish Initiative for Electronic Simulations with Thousands of Atoms
KS	Kohn-Sham
SZ	Single-zeta
DZ	Double-zeta
TZ	Triple-zeta
SCF	Self-consistent field
BRF	Body reference frame
FRF	Fixed reference frame
TDDFT	Time-dependent density functional theory

1 Introduction

Proteins are macromolecules found in all living organisms and they play a vital role in all sorts of chemical reactions and construction of tissue. The shape of a protein is correlated to its function, it is therefore valuable to determine its spacial configuration. Information about the amino acid sequence of a protein is often insufficient to determine the 3D structure [1]. X-ray crystallography has long been the primary way to collect the missing structural information, which combined with prior knowledge can determine the full configuration. While successful this method is limited to molecules that can be crystallized [2].

An alternative imaging method, that would work for noncrystalline biomolecules, was suggested two decades ago by Neutze et al. [3]. In their new method, called single-particle imaging (SPI), isolated particles in vacuum are irradiated with ultrashort and ultraintense X-ray pulses. Such conditions can be found at X-ray free-electron laser (XFEL) facilities [2]. The pulses need to be short enough to allow detection of the diffraction pattern before the sample is ruined by radiation damage. Each pattern must come from different but identical particles, because the samples are destroyed in the process. SPI has been developed significantly over the years. While the resolution needs to be improved further, the method has become much more sophisticated [4]. Certain classes of proteins, such as membrane proteins, are difficult to crystallize and so SPI is a suitable imaging technique [5].

Unfortunately there are several challenges with SPI [5, 6]. One issue is that the weakly scattering biomolecules lead to low signal-to-noise ratios, and background noise coming from various external factors poses challenges in itself. Additionally, the particles are assumed to be identical but this is almost never true at high resolution, this is called the sample heterogeneity challenge. Radiation damage is another concern, since the X-ray pulses are not infinitely short in real experiments the samples can undergo dynamic changes [6]. Another issue is that the diffraction patterns come from random and unknown orientations of the sample molecules, making it difficult to retrieve a complete and consistent 3D data set [5, 7]. This ”orientation challenge” will be the main focus of this project.

Orienting the proteins prior to the irradiation could improve the assembly of diffraction patterns in SPI, as it provides more information. Such preorientation, using an external electric field, was suggested by Marklund et al. [5]. Proteins often carry an intrinsic electric dipole moment, due to uneven charge distribution, which interacts with the external field. A torque is generated and tends to align the dipole in the field, causing an oscillatory motion that can be damped by internal absorption of energy in vibrations. In their study, the possibility of orienting four small proteins in vacuum using static electric fields was investigated using molecular dynamics (MD) simulations. Their results suggest that for a certain window of time and electric field strength, the molecules were oriented without disrupting their structure. This information about the orientation of the proteins, despite being incomplete, improved the assembly of SPI diffraction patterns [5]. Furthermore, such additional knowledge can allow faster convergence of orientation-recovery algorithms, and make it less sensitive to missing data. Consequently, less samples and beam time is required for the SPI experiments and more accurate models can be obtained [2].

Protein orientation in electric fields has, so far, mainly been investigated using classical MD simulations. Previously the research was also limited to static fields or step functions, as in

Marklund et al. [5], and extended to time-dependent fields by Sinelnikova et al. [2]. In such classical simulations the effect of the external electric field on the electronic structure of the proteins is not considered explicitly. Due to the polarizability of the proteins the field can induce an additional contribution to the electric dipole moment. A varying dipole might influence the oscillatory motion of the protein, and therefore the orientation process. Information about the polarizability of a protein, and how it affects its motion in a field, could improve the current models of protein orientation. In turn the theoretical models can provide valuable insight for applications of the technique, such as improving the assembly of diffraction patterns in SPI.

The purpose of this project is to investigate how an applied static electric field changes the polarization, and thus the electric dipole moment, of three small proteins in vacuum. This information will then be implemented in a classical rotor model to study what effect the polarizability has on the orientation dynamics. Static quantum mechanical simulations of the proteins ubiquitin, Trp-cage and lysozyme, all carrying an intrinsic dipole, will be performed in order to extract the induced dipole in varying external electric fields. For a chosen field strength different orientations of the applied field, with respect to a body fixed reference frame of the protein, will be considered. Results from these simulations will be analyzed using Python, to visualize the results in heatmaps. Subsequently the orientation dynamics will be simulated classically by developing a 3D rigid rotor model in Mathematica, solving Euler's equations for the system. For a chosen protein, and same initial conditions, this model will be used to compare the time evolution with and without including polarizability. These results aim to answer the following problem statements:

1. How does the electric dipole moment of the proteins ubiquitin, Trp-cage and lysozyme depend on the strength and orientation of the applied electric field? Namely, what is the polarizability of these small proteins?
2. How does the polarizability affect the motion of the protein in a static electric field? What, if any, impact does it have on the orientation dynamics?

2 Background

2.1 Model proteins

2.1.1 Ubiquitin

Ubiquitin is a small regulatory protein present in nearly all eukaryotic organisms, in other words it can be found *ubiquitously*. The strucutre of ubiquitin was determined several decades ago, and was notably refined at 1.8 Å resolution in the 1980s [8]. Ubiquitin is commonly used in modelling due to its small size and established structure, which has several advantages such as allowing the use of computationally heavy simulations. In previous research on protein orientation in an electric field, ubiquitin was one of the model proteins in the molecular dynamics simulations [2, 5]. It is therefore relevant to investigate the same protein in this study, such that the results can be compared and perhaps combined in the future. Figure 1 shows a ribbon diagram of ubiquitin (Protein Data Bank (PDB): 1UBQ). The unit cell lengths are a = 50.84 Å, b = 42.77 Å and c = 28.95 Å, and once hydrogen atoms are added to the structure it has 1238 number of atoms.

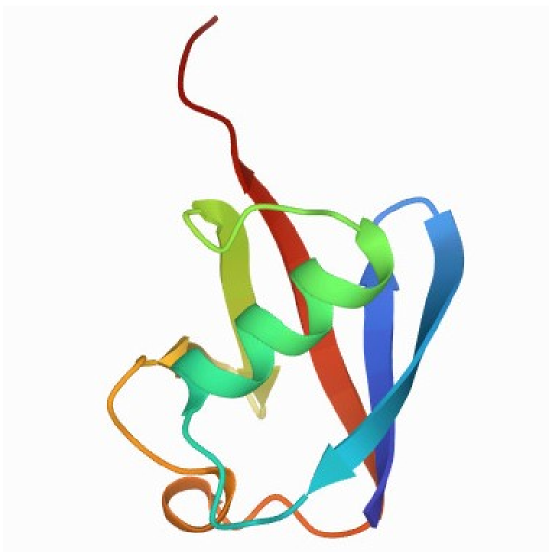


Figure 1: Visualization of the 3D crystal structure of the small protein ubiquitin (PDB ID: 1UBQ) refined at 1.8 Å resolution, in a ribbon diagram. Image from the RCSB Protein Data Bank, corresponding to the original structure publication [8].

2.1.2 Trp-cage

Tryptophan cage (Trp-cage) is a very small protein-like construct, often called miniprotein. Different variants of the Trp-cage miniprotein provide valuable models for both experimental studies and computational simulations [9]. Trp-cage, like ubiquitin, was a successful model in previous protein orientation studies [2, 5] and will be considered here as well. Specifically the Trp-cage miniprotein with PDB ID: 1L2Y, see figure 2. Once hydrogen atoms are added to the structure it has 304 number of atoms.

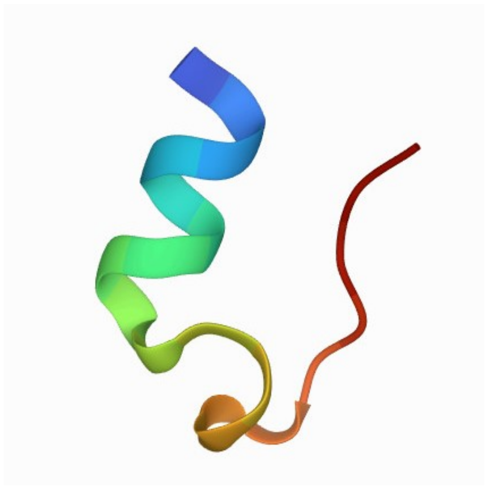


Figure 2: Visualization of the 3D crystal structure of the miniprotein Trp-cage (PDB ID: 1L2Y), construct TC5b, in a ribbon diagram. Image from the RCSB Protein Data Bank, corresponding to the original structure publication [9].

2.1.3 Lysozyme

Lysozyme is a protein that catalyzes biological reactions, an *enzyme*, connected to the immune system of animals. More specifically, it is involved in the process of killing or slowing down the growth of microorganisms. It is a larger protein than both Trp-cage and ubiquitin, but it is still small enough to be suitable for MD simulations [5]. One specific type of lysozyme is the *hen egg-white lysozyme*. The name comes from its abundance in egg-white, and one of its forms is called the *orthorhombic* form. This particular lysozyme variant has PDB ID: 1AKI, see figure 3. The unit cell lengths are a = 59.062 Å, b = 68.451 Å and c = 30.517 Å, and once hydrogen atoms are added to the structure it has 1960 number of atoms.

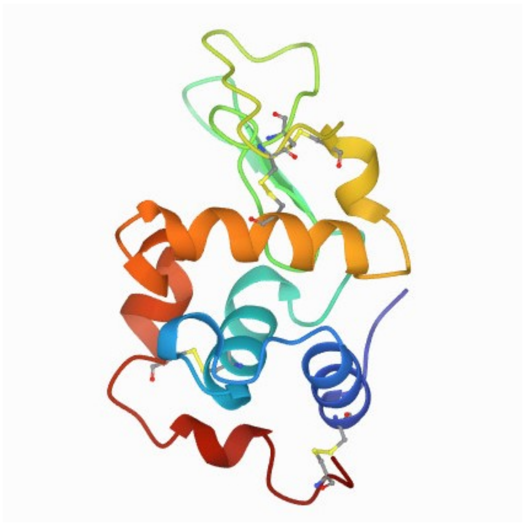


Figure 3: Visualization of the 3D crystal structure of the orthorhombic form of hen egg-white lysozyme (PDB ID: 1AKI) at 1.5 Å resolution, in a ribbon diagram. Image from the RCSB Protein Data Bank, corresponding to the original structure publication [10].

2.2 Dipole orientation

2.2.1 Electric dipole moment

In general, two opposite charges of equal magnitude, separated by a distance r, generate an electric dipole moment $\vec{\mu}$. In physics the convention is to define the dipole moment to be pointing from the negative charge to the positive charge, and a common unit is Debye (1 Debye $\approx 3.34 \cdot 10^{-30}$ Cm) [11]. As a reference, a proton and electron separated by 100 pm corresponds to a dipole of magnitude 4.80 Debye. The electric dipole moment of a molecule is given by the sum of all i contributions:

$$\vec{\mu} = \sum_i q_i \vec{r}_i \tag{1}$$

where q_i is the magnitude of the i^{th} charge-pair and $\vec{r}_i$ is positional vector pointing from the negative to the positive charge [11]. For small molecules it is rather straightforward to calculate its dipole moment, by summing over all bond dipoles - determined by the difference in electronegativity of the two atoms. If all the bond dipoles cancel, which is generally the case for spatially symmetric molecules, the net dipole moment is zero. However, if the structure is not symmetric, like a water molecule, the contributions from each bond do *not* cancel and the molecule carries a net dipole moment [12]. This net dipole of a molecule in vacuum and absence of an external electric field, will be referred to as its *intrinsic* dipole. While explicit dipole calculations can be done by hand for small molecules, it is increasingly difficult for larger, more complex molecules such as proteins. Although the same principle still holds, an uneven charge distribution leads to a non-zero intrinsic dipole moment. This is often the case for proteins [5].

Electric dipole moments interact with external electric fields. In a static electric field a torque is generated, which tends to align the dipole in the field. The torque is given by the following expression,

$$\vec{\tau} = \vec{\mu} \times \vec{E} \tag{2}$$

where $\vec{\mu}$ is the electric dipole moment and $\vec{E}$ is the external electric field [11]. The magnitude of the torque is given by $|\vec{\tau}| = |\vec{\mu}||\vec{E}| \sin\alpha$, where α is the angle between the dipole and the field.

2.2.2 Rigid body dynamics

In classical mechanics the motion of a rigid body is described by Euler's equations, a vectorial differential equation defined in a rotating body fixed reference frame. This section reviews the topic briefly, based on the derivations in [13]. In an inertial reference Newton's second law gives a relation between the time derivative of the angular momentum $\vec{L}$ and the applied torques $\vec{\tau}$,

$$\frac{d\vec{L}}{dt} = \vec{\tau} \tag{3}$$

where by definition $\vec{L} = \mathbb{I}\vec{\omega}$ for a rigid body with inertia tensor $\mathbb{I}$ and angular velocity $\vec{\omega}$, here calculated in the inertial reference frame. Applying the chain rule for the time derivative, equation (3) becomes

$$\frac{d\vec{L}}{dt} = \frac{d(\mathbb{I}\vec{\omega})}{dt} = \dot{\mathbb{I}}\vec{\omega} + \mathbb{I}\dot{\vec{\omega}} = \vec{\tau} \tag{4}$$

where $\dot{\mathbb{I}}$ and $\dot{\vec{\omega}}$ represent time derivatives of the inertia tensor and the angular velocity respectively. This equation exposes a problem with calculating the motion in the fixed reference frame. Since the body is rotating in the lab frame, the time derivative of the inertia tensor is nonzero and often complicates the solution. Therefore a body fixed reference frame is chosen, namely one with a basis defined by the principal axes of the rigid body. This reference frame is not inertial, since it rotates with the body. Therefore equation 3 does not hold in this case, and Coriolis theorem must be applied to obtain the correct expression in the rotating reference frame

$$\frac{d\vec{L}}{dt} + \vec{\omega} \times \vec{L} = \vec{\tau}. \tag{5}$$

The main advantage is that $\dot{\mathbb{I}} = 0$ in this reference frame, thus as the definition of angular momentum of a rigid body is applied the expression becomes

$$\frac{d\vec{L}}{dt} + \vec{\omega} \times \vec{L} = \cancel{\dot{\mathbb{I}}\vec{\omega}} + \mathbb{I}\dot{\vec{\omega}} + \vec{\omega} \times (\mathbb{I}\vec{\omega}) = \vec{\tau} \tag{6}$$

which provides the general form of Euler's equation, in vectorial form

$$\mathbb{I}\dot{\vec{\omega}} + \vec{\omega} \times (\mathbb{I}\vec{\omega}) = \vec{\tau} \tag{7}$$

and in components of the body reference frame basis (defined by the three principal axes suffixed with 1,2,3):

$$\begin{aligned} I_1\dot{\omega}_1 + (I_3 - I_2)\omega_2\omega_3 &= \tau_1 \\ I_2\dot{\omega}_2 + (I_1 - I_3)\omega_3\omega_1 &= \tau_2 \\ I_3\dot{\omega}_3 + (I_2 - I_1)\omega_1\omega_2 &= \tau_3 \end{aligned} \tag{8}$$

If the rotations of the body reference frame, with respect to a lab frame, is characterized by the Euler angles (ϕ, θ, ψ) in for instance a z, x', z" sequence, the angular velocities about the principal axes can be calculated as

$$\begin{aligned} \omega_1(t) &= \dot{\phi}(t)\sin(\theta(t))\sin(\psi(t)) + \dot{\theta}(t)\cos(\psi(t)) \\ \omega_2(t) &= \dot{\phi}(t)\sin(\theta(t))\cos(\psi(t)) - \dot{\theta}(t)\sin(\psi(t)) \\ \omega_3(t) &= \dot{\phi}(t)\cos(\theta(t)) + \dot{\psi}(t) \end{aligned} \tag{9}$$

and a general rotation of a vector $\vec{v}$ in the lab frame to the rotating body frame is given by the following rotation matrix in terms of the Euler angles.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \sin\theta\sin\psi \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \sin\theta\cos\psi \\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{10}$$

2.3 Polarizability

Polarizability, or sometimes called electric polarizability, is essentially a material's *ability* to become *polarized* in the presence of an electric field. Electrical insulators with this tendency are referred to as *dielectrics* in electromagnetism. When a dielectic material is placed in an external electric field the charge distribution is distorted, due to the displacement of charges within the atoms or molecules. Positive charges, nuclei, tend to be pushed in the direction of the field while the negative electrons are forced in the opposite direction. If the field is extremely strong the atoms can be ionized, but for moderate field strengths an equilibrium is reached. The field tends to separate the nuclei and electrons while their mutual attractive force still holds them together. Consequently the atom or molecule now has an induced electric dipole moment, which depends on the applied electric field. For example, a neutral atom placed in an electric field $\vec{E}$ will, generally, obtain an induced dipole moment $\vec{\mu}$ approximately proportional to the field, unless the field is too strong [14]. This is given by the relation

$$\vec{\mu} = \alpha \vec{E}. \tag{11}$$

Here α is the *atomic polarizability*, and the induced dipole is parallel to the electric field. It is, however, not as simple for most molecules, or any kind of anisotropic system. Such structures can be more polarizable in some directions and less in others, and the induced dipole is not necessarily pointing along the applied field. Molecules are therefore better described by the *polarizability tensor* $\boldsymbol{\alpha}$, also called the *molecular polarizability*, a second-rank tensor of the form

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{xx} & \alpha_{xy} & \alpha_{xz} \\ \alpha_{yx} & \alpha_{yy} & \alpha_{yz} \\ \alpha_{zx} & \alpha_{zy} & \alpha_{zz} \end{bmatrix} \tag{12}$$

where the components α_{ij} depend on the choice of axes x, y and z [14]. For instance, a large value of α_{zy} corresponds to strong polarization in the z-direction when a field is applied in the y-direction. The polarizability of a molecule is thus a function of the field, and the general expression of its dipole can be written as a power series in the electric field. For a static electric field this becomes

$$\vec{\mu} = \vec{\mu}_0 + \boldsymbol{\gamma}^{(1)} \cdot \vec{E} + \frac{1}{2}\boldsymbol{\gamma}^{(2)} : \vec{E}\vec{E} + \frac{1}{6}\boldsymbol{\gamma}^{(3)} \vdots \vec{E}\vec{E}\vec{E} + ... \tag{13}$$

where $\vec{\mu}_0$ is the intrinsic dipole and $\boldsymbol{\gamma}^{(n)}$ are rank $n+1$ tensors with units Cm(Vm^{-1})$^{-n}$[15]. The linear term involves a second-rank tensor $\boldsymbol{\gamma}^{(1)}$, which corresponds to the previously described *molecular polarizability* $\boldsymbol{\alpha}$. For the higher-order terms, $n > 1$, these tensors are often called *hyperpolarizabilities*. Here the operations between the polarizabilities and the field is denoted by a tensor dot product, double dot product, triple dot product and so on for higher order terms. Using a common notation for the lowest order tensors the dipole is given by the following expansion [15].

$$\vec{\mu} = \vec{\mu}_0 + \boldsymbol{\alpha} \cdot \vec{E} + \boldsymbol{\beta} : \vec{E}\vec{E} + \boldsymbol{\gamma} \vdots \vec{E}\vec{E}\vec{E} + ... \tag{14}$$

For sufficiently low field strengths the linear, first order term should dominate the contribution of the induced dipole to the total dipole. In that case the dipole can be approximated as $\vec{\mu} = \vec{\mu}_0 + \boldsymbol{\alpha} \cdot \vec{E}$, where the dipoles and the field can be represented by 3D vectors and the molecular polarizability as a matrix. In this work the polarizability of the proteins will be parameterized, and the linearity for the considered field strengths will be investigated.

2.4 Density functional theory and SIESTA

Systems with many atoms, such as proteins, could in principle be described explicitly by all interactions between the nuclei and electrons. In addition to this there could be external interactions as well, such as with an electric field. These many-body problems have large Schrödinger equations with too many coupled degrees of freedom to be solved in practice. In the 1960s a new approach to this problem was established by Hohenberg, Kohn and Sham [16, 17], called Density functional theory (DFT). It is a method that considers the electron density rather than separate electrons, enabling various calculations of complex many-body systems. DFT is today one of the most useful and powerful methods for calculating the electronic structure, and other properties, of many-body systems in condensed matter. It is also used more and more for studies of molecules and other finite systems [18].

To simplify the complicated equations of an interacting many-body system, DFT often starts from the Born-Oppenheimer approximation. In situations where the nuclei are significantly heavier and slower than the electrons, the nuclei can be considered at rest within the time-scale required for the electrons to reach their ground state configuration. This allows the nuclei and electron degrees of freedom to be decoupled from each other in the Schrödinger equation, which is less computationally demanding. Another fundamental aspect of DFT is the Hohenberg-Kohn theorems. The first one states that the external potential, and hence the total energy, in a system of many interacting particles is a unique functional of the electron density. The second theorem adds that the ground state energy can be obtained variationally, meaning the density that minimizes the total energy is the exact ground state density [18]. Kohn and Sham then suggested an approximate way of constructing the ground state functional for real systems with many electrons. Namely replacing the interacting problem with a non-interacting system where all the electron interactions, many-body effects, are included in an *exchange-correlation* (XC) functional [17]. This is commonly referred to as the *Kohn-Sham ansatz*. The Hamiltonian of the system will have several classical terms, such as the attractive Coulomb potential from the nuclei, as well as terms involving quantum effects - the XC term. To clarify, exchange and correlation refers to interactions between electrons with same spin and opposite spin respectively.

In practice one has to find suitable approximations for the XC functional. One widespread example is the simple approach *Local Density Approximation* (LDA), which is based on a homogeneous electron gas where the energy only depends on the local density at each point in space. A slightly more non-local treatment is given by *Generalized Gradient Approximation* (GGA), where the XC term is not only a functional of the electron density, as in LDA, but also of the gradient of the density. Despite their simplicity, LDA and GGA have proven to be incredibly successful. The enhancement that GGA provided to LDA improved the efficency and accuracy for atoms and molecules [18]. A particular version of GGA is suitable for electron structure calculations of such systems, named PBE-GGA after the authors Perdew, Burke and Ernzerhof [19, 20].

Additionally, so called *pseudopotentials* are often utilized to deal with the large numbers of electrons in many-body systems. Generally, the valence electrons feel the strong Coulomb potential of the nuclei, along with the effects of the tightly bound core electrons, this is replaced with a combined, effective ionic potential. Since the core electrons are normally quite inert to effects of the chemical environment, compared to the valence electrons, the core states remain more or less unchanged. This allows the pseudopotentials to be generated in a single atomic calculation for a particular element, and then used in the computations of solids and molecules where only the valence electrons are considered explicitly [18].

SIESTA (Spanish Initiative for Electronic Simulations with Thousands of Atoms) [21] is a method and computer implementation based on DFT. It is used to calculate electronic structures and run *ab initio* molecular dynamics simulations of molecules and solids. The program is built on the standard Kohn-Sham (KS) selfconsistent density functional method, with the LDA and GGA approximations along with a non-local functional that includes van der Waals interactions (VDW-DF). Pseudopotentials are used in SIESTA, particularly of the norm-conserving and non-local form. The basis sets are atomic orbitals, which are efficient but might require some trial and error in tuning parameters to achieve optimal convergence. All basis sets that are a products of strictly localized radial functions and a spherical harmonics is compatible with SIESTA, if they also vanish beyond a certain cutoff radius. This confinement of the orbitals to a finite range causes an increase in energy, which can be adjusted for by the parameter *energy shift* (with default value 0.02 Ry) [22].

The number of orbitals per atom can be varied in SIESTA using the *basis size* input parameter, where the hierarchy of basis set sizes corresponds to the quantum chemistry convention - from single-zeta to multiple-zeta with polarization and diffuse orbitals. A single-zeta (SZ) basis set has only one single radial function per significant angular momentum quantum number, meaning one for each angular momenta with enough electron population to be important. More flexible alternatives is the double-zeta (DZ) and triple-zeta (TZ) basis set sizes. They have two and three functions per angular momentum channel respectively, which enables more radial freedom in the energy optimization scheme. Orbitals with higher angular momentum can be added to the basis set to provide further flexibility in orbital occupations. SIESTA can generate such *polarization orbitals*, again to varying degrees such as P, DP, 3P [22]. For instance the basis set "SZP" corresponds to single-zeta basis plus polarization orbitals. The extension of polarization orbitals to the basis set is particularly relevant for simulations in external electric fields.

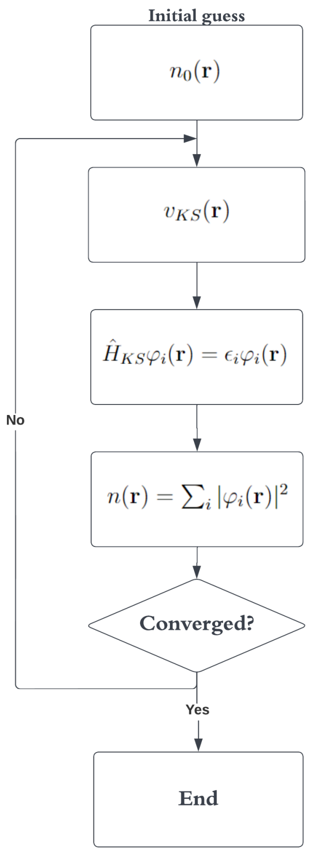

Initial guess
$n_0(\mathbf{r})$
$v_{KS}(\mathbf{r})$
$\hat{H}_{KS}\varphi_i(\mathbf{r}) = \epsilon_i\varphi_i(\mathbf{r})$
$n(\mathbf{r}) = \sum_i |\varphi_i(\mathbf{r})|^2$
Converged?
No
Yes
End

Figure 4: Outline of a DFT program, which is the foundation of the SCF cycle in SIESTA. Made with LucidChart [23].

The self-consistent field (SCF) cycle in SIESTA follows the typical outline of a DFT program, see figure 4. First an initial guess of the electron density is made, either from a previous run, which can be mixed in various ways, or constructed from the input information about the atoms if it is the very first loop. Then the effective potential is calculated based on this density, which is used to solve the KS equation for the KS orbitals. These are in turn used to calculate a new density. Since the input potential depends on the output density the KS equations must be solved self-consistently. In other words, the difference between the previous density (input) and the new density (output) must become lower than a given tolerance. Only then will the SCF cycle stop and output quantities such as energy, forces and eigenvalues are returned. If the difference is larger than the tolerance a new iteration is started, where the input density is given by a certain mix of previous densities. The number of SCF steps required for a simulation to reach self-consistency depends strongly on this mixing procedure. There are several mixing methods and parameters available in SIESTA which can be tuned to accelerate the convergence, which can vary between calculation types and systems. For example the parameter *mixing weight* controls the weight in the different mixing methods, and *number pulay* (or *mixer history*) is the number of previous SCF steps used in estimating the following input [22]. Both can play a role in the stability and pace of convergence.

3 Method

3.1 Polarizability calculations

All the ab initio quantum calculations, described in this section, were performed on the high performance computing cluster *Tetralith*, hosted by Linköping's University, using SIESTA [21] software (version 4.1.5). Specifically ground state simulations of the proteins ubiquitin, Trp-cage and lysozyme in vacuum, with net charges +7 e, +2 e and +8 e respectively. These particular net charges can be expected when the proteins are aerosolized using electrospray ionization, a method that could be used in SPI experiments [2, 24]. Additional calculations and analysis of simulation results were performed using Python language in the JupyterLab environment (version 1.1.4). See the Python script in appendix C.2. The SIESTA simulations were carried out both in static external electric fields with varying angles, with respect to a chosen body reference frame of the protein, and in absence of a field. Quantum effects are considered in SIESTA, as described in section 2.4, and in this context the relatively small proteins are very large systems that require high computational effort. Therefore the simulations did not include any molecular dynamics. Only the electronic polarizability is considered, since the nuclei are kept fixed, related to the distortion of the electron density due to the external electric field. The contribution from the displacement of the nuclei to the polarizability can be expected to occur in a longer timescale than the electronic response. Electrons respond in approximately 10^{-15} s, while other responses such as atomic, ionic or dipolar happen in roughly 10^{-12} to 10^{-6} s. Additionally, extremely strong fields are required to significantly displace the nuclei - above 45 V/nm to break protein bonds, as reported by Sinelnikova et al. [2].

The structural information about each protein, ubiquitin (1UBQ), Trp-cage (1L2Y) and lysozyme (1AKI), in vacuum was gathered from the corresponding PDB files (see section 2.1). The missing hydrogen atoms were added to the proteins using GROMACS software [25], which was also used to calculate the moment of inertia and the three principal axes of each protein. These simulations were performed by Emiliano De Santis at Uppsala University. Note that there was no relaxation process in these MD simulations. Atom species and coordinates (in Ångströms) were then extracted from these modified PDB files, and rewritten such that the structural input was compatible with SIESTA. An outline of a typical input-file for the SIESTA simulations can be found in appendix C.1. The atom positions were specified with (x, y, z) coordinates, were the unit vectors $\hat{x}, \hat{y}, \hat{z}$ define the lab frame and will be referred to as the fixed reference frame (FRF). Since the three principal axes are mutually perpendicular, and related to the structure of the proteins, they were chosen to define the body reference frame (BRF) with unit vectors $\hat{e}_1, \hat{e}_2, \hat{e}_3$. Namely the major principal axis, the middle principal axis and the minor principle axis with corresponding moments of inertia components I_i in the diagonalized inertia tensor. To transform between these reference frames a rotation matrix R was determined, see appendix A.2. The numerical values, provided by Emiliano De Santis, of the three principal axes as unit vectors and the moments of inertia are listed in table 1 for each protein.

Table 1: Unit vectors of the body reference frame (BRF) $\hat{e}_1, \hat{e}_2, \hat{e}_3$ given by the principal axes of the protein, along with the corresponding moments of inertia. Here m_u ($\approx 1.66 \cdot 10^{-27}$ kg) is the atomic mass unit, also called Dalton.

	$\hat{e}_1$	$\hat{e}_2$	$\hat{e}_3$	I_1, I_2, I_3 [$\mathrm{m}_u\mathrm{nm}^2$]
Ubiquitin	(0.13793203, 0.80497851, 0.57704797)	(-0.59855185, 0.53193899, -0.59897963)	(-0.78912004, -0.26277465, 0.5551928)	8701.098, 7250.633, 6310.081
Trp-cage	(0.8506957889, -0.5197643042, 0.0784965754)	(0.4867914617, 0.8353160620, 0.2555016577)	(-0.1983700842, -0.1791427284, 0.9636167288)	1001.969, 758.8640, 549.0527
Lysozyme	(-0.3703268170, 0.8485152721, -0.3779945076)	(-0.7241570354, -0.0088601764, 0.6895781755)	(0.5817685723, 0.5290966630, 0.6177394986)	22 562.27, 21 470.59, 12 002.60

In order to parameterize the polarizability of the proteins different angles of the applied static electric field was considered. The azimuthal and polar angle of the field in the BRF, for a chosen electric field strength, were varied in order to map out the 3D protein structure unambiguously. In a typical spherical coordinate system the polar angle is then the angle between the field and $\hat{e}_3$, and the azimuthal angle is the counterclockwise angle between $\hat{e}_1$ and the field projected on the $\hat{e}_1, \hat{e}_2$-plane. They will be referred to as γ and β, and they range between $[0, \pi]$ and $[0, 2\pi]$ respectively. See figure 5.

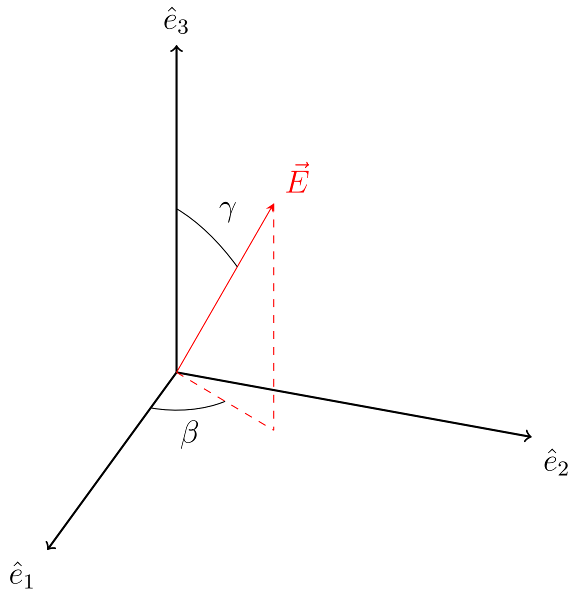


Figure 5: Definition of the azimuthal angle β and polar angle γ of the electric field $\vec{E}$ in the body reference frame (BRF), where axes $\hat{e}_1, \hat{e}_2, \hat{e}_3$ correspond to the principal axes of the protein.

Initially the convergence of the SIESTA simulations was tested for each protein structure. Mainly balancing the basis set size, execution time and potentially altering other parameters if convergence was not reached at all. In general the XC-functional was PBE-GGA and the XC integration grid was determined by a 100 Ry cutoff. The basis functions were confined with a shallow potential, *energy shift* set to 0.005 Ry, to allow more diffuse orbitals. The electronic temperature was set to 300 K. The parameters *number pulay* and *mixing weight*, both concerning the density mixing method, were initially chosen to be 5 and 0.05 respectively. To determine a suitable basis size for each protein, before knowing in which directions they are most polarizable, the convergence was tested in zero electric field. These simulations were also necessary to calculate the intrinsic dipole $\vec{\mu}_0$. Since the dipole is the desired output of the simulations, it is the relevant quantity to consider in the basis set comparison. Table 2 shows the tested basis set sizes for ubiquitin in absence of an external electric field, providing the simulation results for the intrinsic dipole (written in the BRF), the magnitude of the intrinsic dipole, the real time and the number of SCF steps.

Table 2: Convergence test for ubiquitin in zero electric field. The intrinsic dipole, execution time and number of SCF loops is given for different basis set sizes in the SIESTA simulations. Basis TZ3P was chosen for the following simulations of ubiquitin.

	$\vec{\mu}_0$ **in BRF** [Debye]	$\lvert\vec{\mu}_0\rvert$ [Debye]	**Execution time**	**SCF steps**
SZP	(-0.5843, -79.06, -16.27)	80.72	2m 57s	20
DZP	(-1.443, -81.74, -15.48)	83.21	5m 20s	22
DZDP	(-2.207, -82.38, -14.91)	83.74	15m 41s	30
TZ3P	(-2.251, -82.31, -14.82)	83.66	39m 51s	30

The simulation with the largest basis size, the triple-zeta basis plus three polarization orbitals (TZ3P), only took 40 minutes to converge. Even though the magnitude of the intrinsic dipole only varied slightly between the basis sets, one could expect the difference to be larger when a field is applied. Therefore the subsequent simulations with ubiquitin were performed with basis set size TZ3P. A similar convergence check was conducted for Trp-cage, see table 3, and with the same arguments the basis set size TZ3P was chosen.

Table 3: Convergence test for Trp-cage in zero electric field. The intrinsic dipole, execution time and number of SCF loops is given for different basis set sizes in the SIESTA simulations. Basis TZ3P was chosen for the following simulations of Trp-cage.

	$\vec{\mu}_0$ **in BRF** [Debye]	$\lvert\vec{\mu}_0\rvert$ [Debye]	**Execution time**	**SCF steps**
SZP	(-54.49, -23.77, -22.31)	63.50	7m 40s	266
DZP	(-57.13, -22.53, -19.66)	64.48	2m 27s	57
TZP	(-57.19, -22.56, -19.45)	64.48	4m 14s	83
TZ3P	(-57.88, -22.30, -19.12)	64.90	8m 15s	61

Lysozyme is the largest protein of the three, and it failed to converge with the initial parameters. After some trial and error, slightly changing one input parameter at the time, the simulation converged when the *mixing weight* was decreased from 0.05 to 0.01. Since this parameter only affects the mixing procedure in the SCF loop, as mentioned previously, it should not alter the results only the convergence stability and speed. With this new value of the *mixing weight*, a few basis set sizes was tested in zero field and presented in table 4.

Table 4: Convergence test for lysozyme in zero electric field. The intrinsic dipole, execution time and number of SCF loops is given for different basis set sizes in the SIESTA simulations. Basis DZP was chosen for the following simulations of lysozyme.

	$\vec{\mu}_0$ **in BRF** [Debye]	$\vert\vec{\mu}_0\vert$ [Debye]	**Execution time**	**SCF steps**
SZP	(-86.60, -37.24, -33.52)	100.0	42m 52s	111
DZP	(-92.45, -38.19, -34.80)	105.9	53m 16s	109
TZ3P	(-92.70, -39.00, -35.23)	106.6	5h 43m 1s	97

The intrinsic dipole barely varied between basis sets DZP and TZ3P, especially compared to the difference in execution time. Due to the almost five hour longer real time for basis set TZ3P, the smaller basis DZP was chosen for the rest of the lysozyme simulations.

After the convergence tests, from which the intrinsic dipoles were provided as well, simulations with different static external electric fields were performed. For one field strength, 1 V/nm, the orientation of the field was varied with respect to the BRF. This was characterized by the angles β and γ, see figure 5, which were mapped out in their respective ranges in steps of 30 degrees. This particular field strength was chosen in order to consider a field strong enough to orient proteins without unfolding them [2, 5], while hopefully weak enough such that linear polarizability is a good approximation. The field components in the FRF, for a particular set of angles, was computed by rotating one of the basis vectors of the BRF about two axes using the general Rodrigues' rotation formula (20). First $\hat{e}_3$ was rotated about the $\hat{e}_2$ axis γ degrees, then that vector was rotated about the $\hat{e}_3$ axis β degrees. For other field strengths than 1 V/nm the resulting electric field vector was then scaled accordingly, such that the magnitude of the vector equaled the field strength.

The electric dipole was extracted from the output of the SIESTA simulations, for each pair of angles β and γ respectively. The dipole vectors were then transformed to the BRF and interpolated using *CloughTocher2DInterpolator*, a piecewise cubic, C1 smooth, curvature-minimizing interpolant in 2D, from the SciPy library. This provided a function with input (β, γ) that returns the corresponding total dipole. Next this function was used to plot various properties with respect to the angle of the field. These included the magnitude of the total dipole as a function of β and γ, as well as the magnitude of the induced dipole (the total dipole minus the intrinsic dipole). The angle between the total dipole and the intrinsic dipole, how much the orientation of the dipole changed due to the field, was also calculated and plotted. Also the angle between the induced dipole and the field, which for large values indicates significant off-diagonal terms in the polarizability tensor. All these results were visualized with heatmaps, both in 2D and plotted on a sphere.

As a quick test of the interpolation itself, an extra simulation was run for ubiquitin with field angles between two sample points (165°, 75°) such that the resulting dipole could be compared to the dipole from the interpolating function. This particular set of field angles was chosen because it lies in a region where the dipole changed significantly within small angles. Moreover, the relevant output files from three chosen simulation runs of ubiquitin were used to obtain information about the charge density. This was performed using the DENCHAR [26] program that returns *cube*-files with the desired information. The chosen simulations were the zero field case, when the is field parallel to the intrinsic dipole and for field angles (150°, 60°). The latter corresponding to a case with a large angle between the field and the induced dipole. Files with the zero field charge density, along with the difference between the other two cases and the zero field case, were then opened in a 3D visualization program called VESTA [27].

In order to test the linearity of the polarizability four additional SIESTA simulations were performed per protein. Two sets of field angles (β, γ) were chosen, the parallel case and one case where the protein was most polarizable (largest induced dipole). These were chosen to be (90°, 60°) for ubiquitin, (0°, 90°) for Trp-cage and (30°, 90°) for lysozyme. For each of these field orientations simulations were carried out with field strengths 0.5 V/nm and 0.1 V/nm, as an addition to the results from 1 V/nm and zero electric field. The magnitude of the total dipole and the induced dipole was then plotted as a function of field strength, along with a linear or quadratic fit calculated using NumPy's *polyfit*.

3.2 Dipole orientation dynamics

Due to the complexity of the DFT-based simulations, the proteins considered here are too large systems for time-dependent DFT (TDDFT) to be applicable for the relevant timescales. Therefore the orientation dynamics part of the project was developed strictly based on classical mechanics, with the possibility of including the effects of the protein's polarizability from the results of the first part. After trying a few different approaches the most successful model was, in the end, designed in Wolfram Mathematica 13.0 [28] together with Thomas Mandl from Uppsala University/University of Applied Sciences Technikum Wien. See the script in appendix C.3. It is a 3D rigid rotor model that solves for the time evolution of a protein, assumed to be a rigid body with a known inertia tensor and intrinsic dipole, in a static electric field. The model was expanded by implementing the polarizability results for ubiquitin in a static field with strength 1 V/nm. It would be possible to extend the model to include oscillating fields and test other proteins. For the purpose of this study, the foundation of the model is described here and it was tested for one set of initial conditions to demonstrate its potential.

The same reference frames as in the polarizability calculations were used in the 3D rotor model, namely the fixed lab frame (FRF) and the body frame (BRF). At any point in time the BRF was calculated by rotating the FRF using the Euler angles (ϕ, θ, ψ). Mathematica's *EulerMatrix* function was applied with the z, x', z" sequence, referring to the order of operations: (1) rotate ϕ about the z-axis (precession), (2) rotate θ about the x'-axis (nutation) and (3) rotate ψ about the z"-axis (spin). A system of equations was then built up by expressing the angular velocities, see equation (9) , in terms of the Euler angles and plugging it into Euler's equations (8) for the BRF. The time evolution of the system was then calculated using Mathematica's internal solver for differential-algebraic systems *NDSolve* for a chosen time interval. This required a given static electric field, inertia tensor and initial conditions for Euler angles and their respective angular velocities (determines the starting orientation of the BRF with respect to the FRF).

Without including polarizability the torque was given by $\vec{\tau} = \vec{\mu}_0 \times \vec{E}$ at all points in time, where $\vec{\mu}_0$ is the protein's intrinsic dipole and $\vec{E}$ is the chosen static electric field. To include the effects of polarizability the dipole was allowed to vary in time. For compatibility with the interpolation results from the SIESTA simulations this was set up by letting the dipole be a function of the field angles (β, γ),

$$\vec{\tau} = \vec{\mu}(\beta, \gamma) \times \vec{E} = \vec{\mu}(\phi(t), \theta(t), \psi(t)) \times \vec{E} \tag{15}$$

which in turn were unambiguously determined by the Euler angles at any time t. Figure 6 depicts the necessary geometrical definitions of the system, including the FRF, BRF, electric field and angles (β, γ) at some point in time.

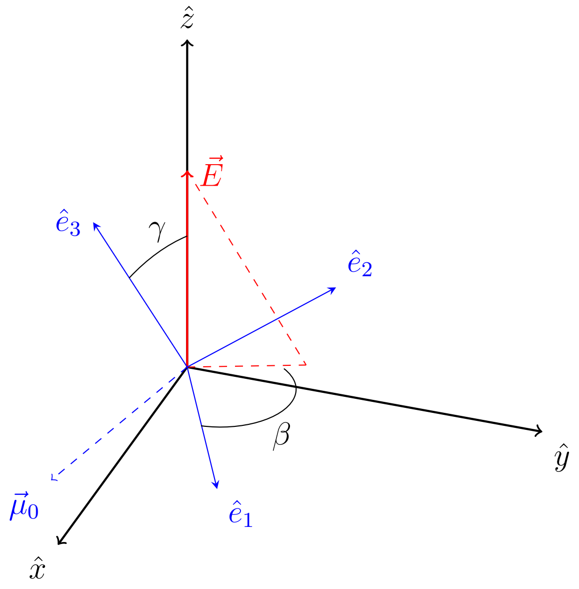


Figure 6: Definitions of the lab frame (black) and rotating body frame (blue) of the 3D rigid rotor model, visualized at some point in time. A chosen static electric field (red) is depicted along with the corresponding spherical angles (β, γ) with respect to the body frame. A scaled down intrinsic dipole $\vec{\mu}_0$ of ubiquitin is used as an example, which varies depending on the protein.

The dipole $\vec{\mu}$ is then the total dipole, the intrinsic plus the induced dipole, for a particular orientation of the field with respect to the BRF. Similarly to the Python approach, this was achieved by interpolating the total dipole results from SIESTA (written in BRF) since those values were calculated in steps of 30°. In Mathematica *ListInterpolation* was used for each vector component of the dipole separately. At each time t the orientation of the BRF, with respect to the FRF, is known from the Euler angles. Therefore the field angle γ was simply calculated using the internal *VectorAngle* between the field and $\hat{e}_3$, both written in the FRF. The azimuthal angle β was determined by the angle between $\hat{e}_1$ and the field projected on the $\hat{e}_1, \hat{e}_2$-plane. This was most efficiently done in the BRF. To distinguish between the first two and last two quadrants, the angle returned by *VectorAngle* was subtracted from 360° if the second component of the projected field was negative, meaning it lay in the third or fourth quadrant. Otherwise β was simply given by the calculated *VectorAngle*.

Many properties of the system can be extracted from the solution and visualized. For the purpose of this project some of these properties were considered, to provide an example of how the rotor model works and its potential value. Note that since the simulations were strongly dependent on the initial conditions, one particular example does not provide any general behaviour of the system. This would require many more simulations, perhaps determining the average orientation for each case. That being said, an illustrative example was found when considering ubiquitin in a static electric field $(1,\, 0,\, 0)_{FRF}$ V/nm with the following initial conditions.

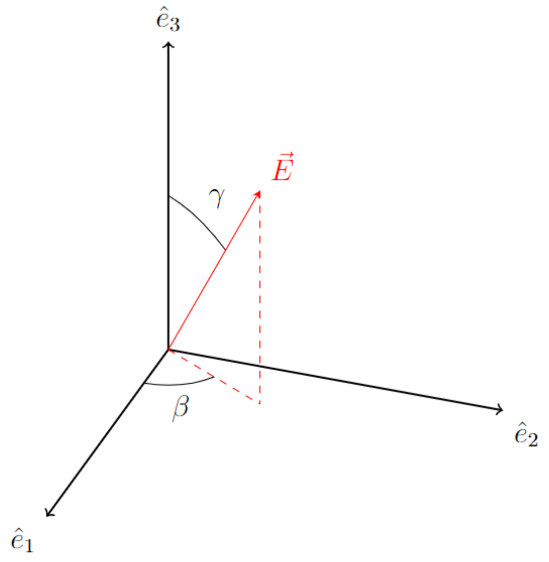
$$\phi_0 = 4\pi/5,\;\; \theta_0 = \pi/4,\;\; \psi_0 = \pi/2$$
$$\dot{\phi}_0 = 0,\;\; \dot{\theta}_0 = 0,\;\; \dot{\psi}_0 = 0$$

The inertia components were set to $I_1 = 8701.1, I_2 = 7250.6, I_3 = 6310.1$ [$\mathrm{m}_u\mathrm{nm}^2$], and the system was solved for 2000 time-steps for the cases with and without polarizability. Extracted values for the BRF basis, as well as the dipole, in the FRF was plotted in 3D for all time-steps. Additionally the angle between the dipole and the field was calculated and plotted as a function of time. The magnitude of the dipole was illustrated similarly. The code includes the possibility to calculate and visualize many other properties, along with movies of the whole orientation dynamics process. Finally, using Mathematica's *Quantity* and *UnitConvert* the simulation time-step was found to correspond to appoximately 0.07 ps. This was determined from the units of $\sqrt{(\frac{\mu E}{I})^{-1}}$, which corresponds to a time unit, when the field is given in V/nm, the dipole in Debye and the inertia in $\mathrm{m}_u\mathrm{nm}^2$.

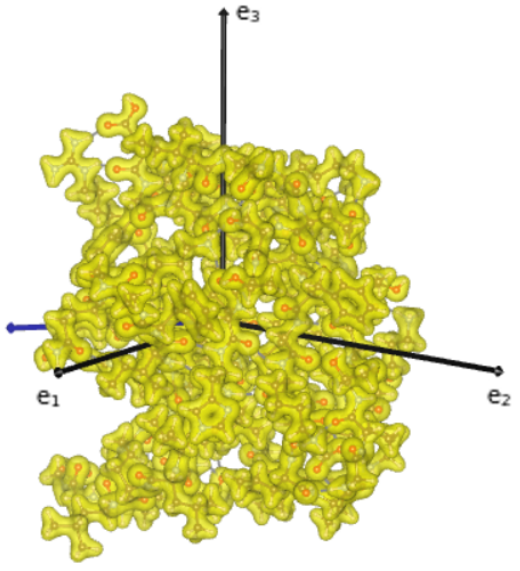
4 Results

4.1 Polarizability of small proteins

The following results were obtained according to the method described in section 3.1. A selection of figures are provided here, focusing first on the protein ubiquitin and then lysozyme. Similar results for Trp-cage can be found in appendix B. First the setup of ubiquitin is described, then heatmaps of the resulting dipole magnitudes and various angles for the different orientations of the applied field are presented, see figures 7, 8 and 9 respectively. Additionally a test of the accuracy of the interpolation is provided, for a chosen simulation of ubiquitin, see table 5. Visualizations of the electron density of ubiquitin, for two different cases, can be found in figure 10 and a brief linearity test of the polarizability is given in figure 11.



(a)



(b)

Figure 7: (a) Definition of the azimuthal angle β and polar angle γ of the electric field $\vec{E}$ in the body reference frame (BRF). (b) Electron density of **ubiquitin** in zero electric field, visualized using VESTA [27] with isosurface level 0.476836 electrons/Å^3. Black arrows correspond to the BRF basis, dark blue arrow is the intrinsic dipole $\vec{\mu}_0$.

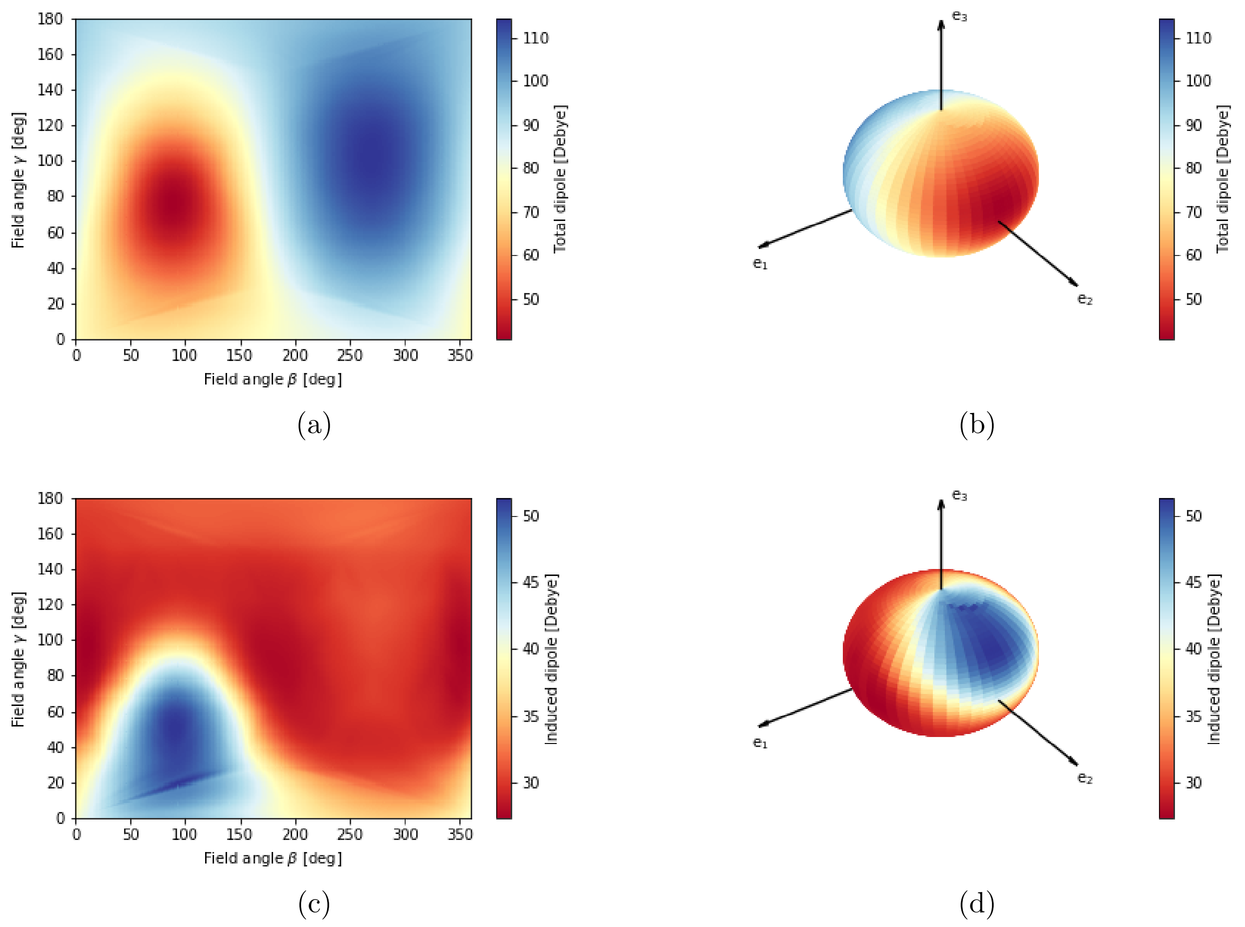


Figure 8: Results from the analysis of SIESTA simulations for **ubiquitin** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Magnitude of the total electric dipole (in Debye) as a function of the field angles (β, γ) as defined in figure 7. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Magnitude of the induced dipole (in Debye) as a function of field angles, visualized on a sphere in (d). For comparison, the magnitude of the intrinsic dipole was $|\vec{\mu}_0| = 83.66$ Debye, and its direction was given by ($\beta = 268°, \gamma = 100°$).

Table 5: Comparison of the total dipole of **ubiquitin**, in the BRF, between the simulation result for field angles ($\beta = 165°, \gamma = 75°$) and the result from the interpolating function (excluding this simulation). Absolute error (the difference) and the relative error (absolute error divided by the magnitude of the exact simulation value) in percent is given for the dipole components and magnitude separately.

	Total dipole in BRF [Debye]	**Magnitude [Debye]**
Interpolation	(-28.268, -72.448, -4.7833)	77.914
Simulation	(-28.392, -72.273, -4.7962)	77.798
Absolute error	(0.12414, 0.17459, 0.012936)	0.11630
Relative error	(0.43722 %, 0.24157 %, 0.26971 %)	0.14949 %

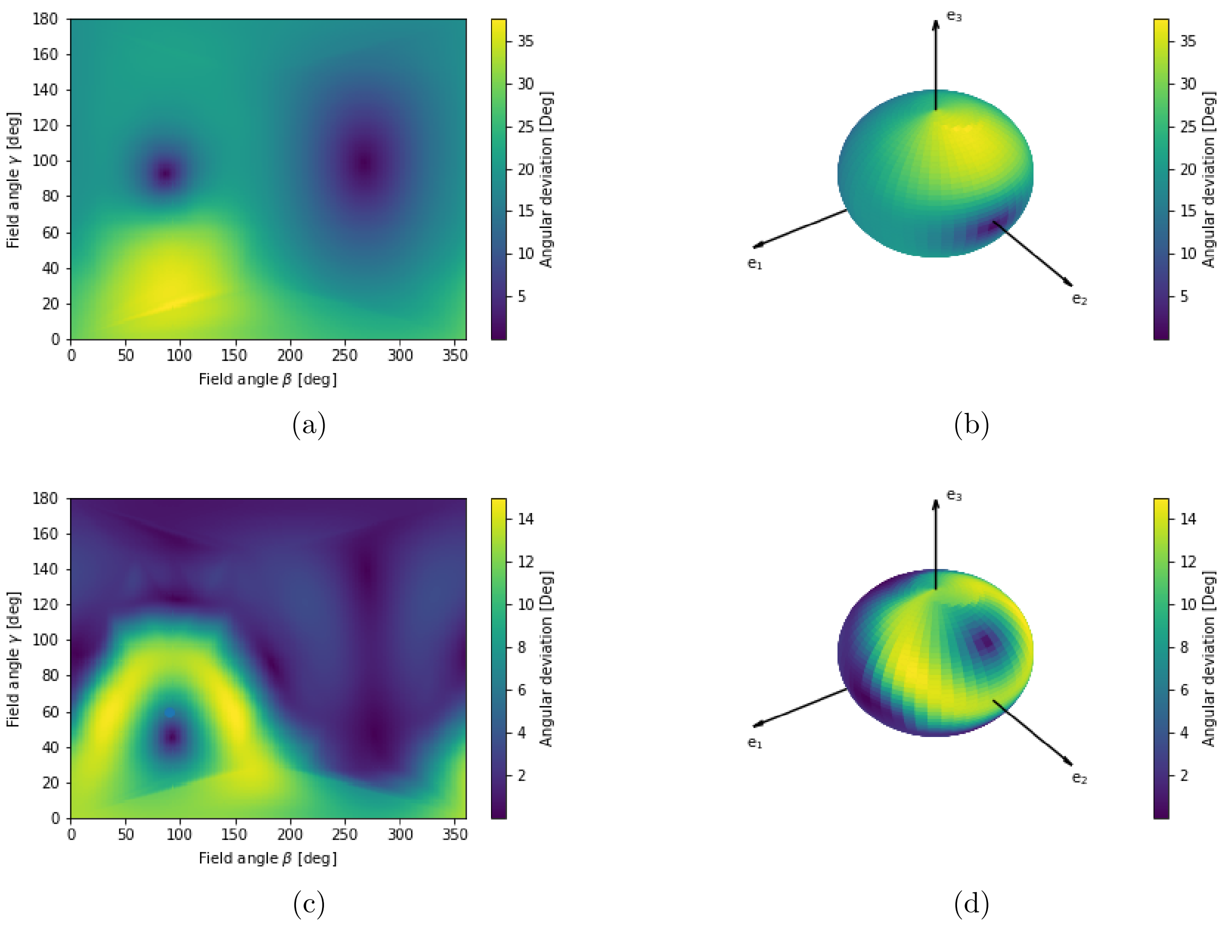
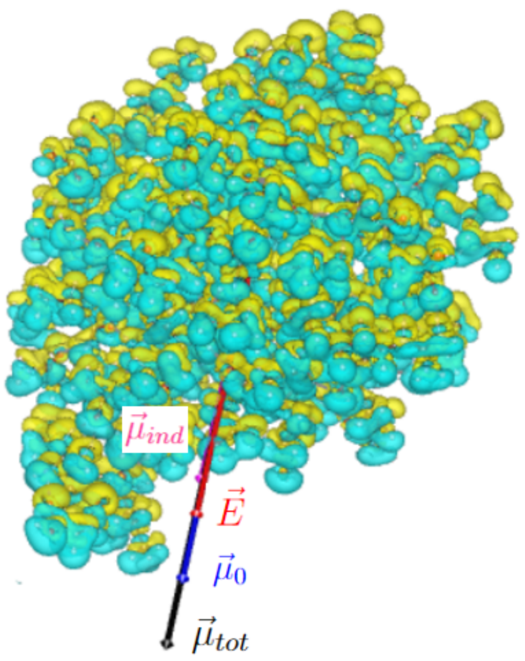
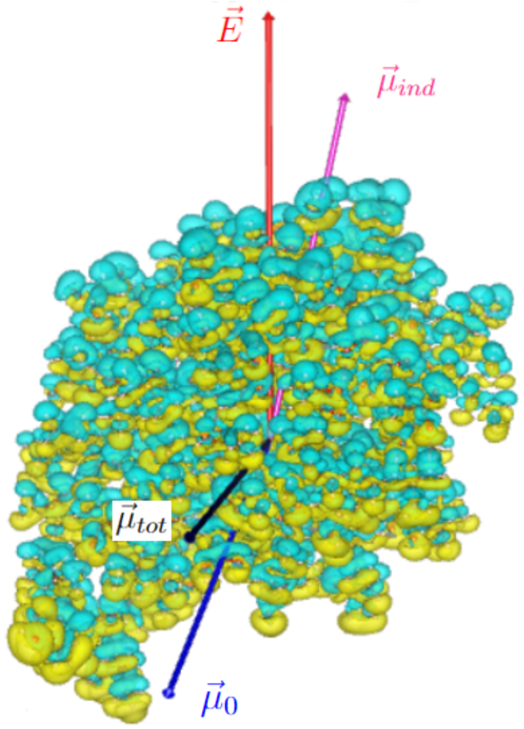


Figure 9: Results from the analysis of SIESTA simulations for **ubiquitin** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Angle between the total dipole and the intrinsic dipole (in degrees) as a function of the field angles (β, γ) as defined in figure 7. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Angle between the induced dipole and the applied field (in degrees) as a function of field angles, visualized on a sphere in (d).

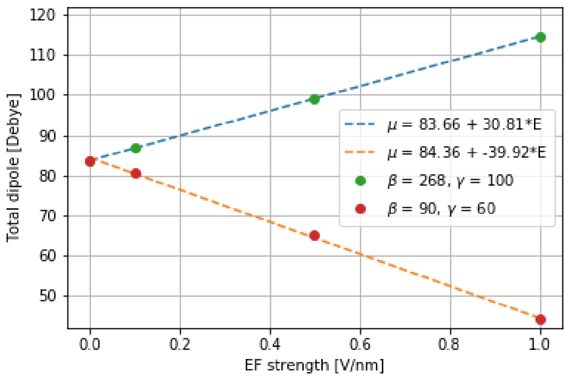


(a)

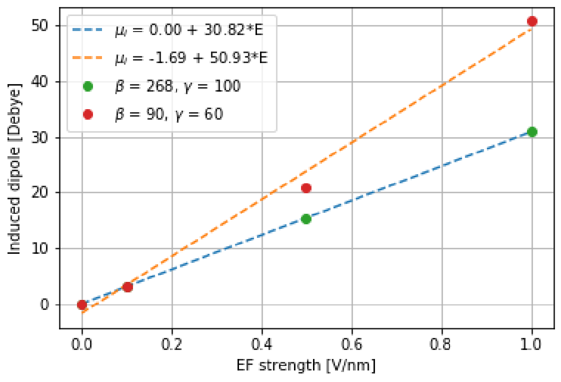


(b)

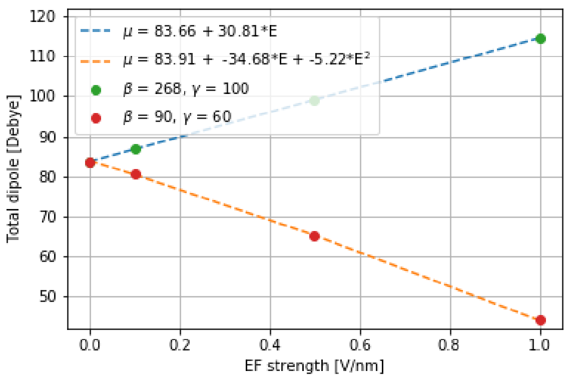
Figure 10: Visualizations of differences in electron density of **ubiquitin** in two different fields with respect to the density in zero field. Created from the SIESTA simulation output using VESTA [27] in the *positive and negative mode*. Yellow corresponds to an increase in electron density, and blue to decreased electron density. Plotted along with the total dipole $\vec{\mu}_{tot}$, the intrinsic dipole $\vec{\mu}_0$, the induced dipole $\vec{\mu}_{ind}$ and the field $\vec{E}$ (no arrows are to scale). (a) Difference in electron density between the case with field angles ($\beta = 268°, \gamma = 100°$), parallel to the intrinsic dipole, and zero field. Isosurface level 0.000542821 electrons/Å^3. (b) Difference in electron density between the case with field angles ($\beta = 150°, \gamma = 60°$) and zero field. Isosurface level 0.000583952 electrons/Å^3.



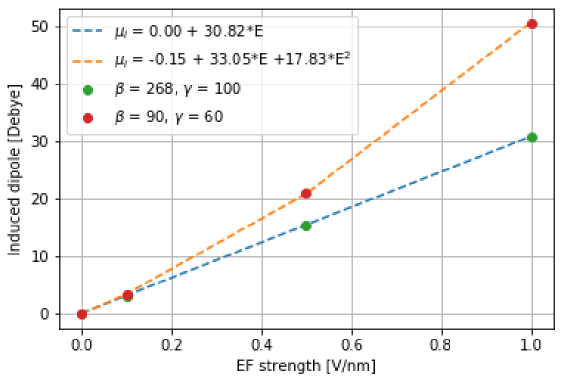
(a)



(b)



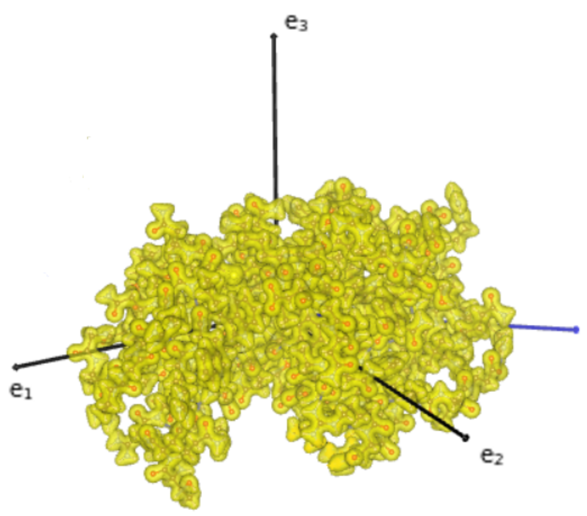
(c)



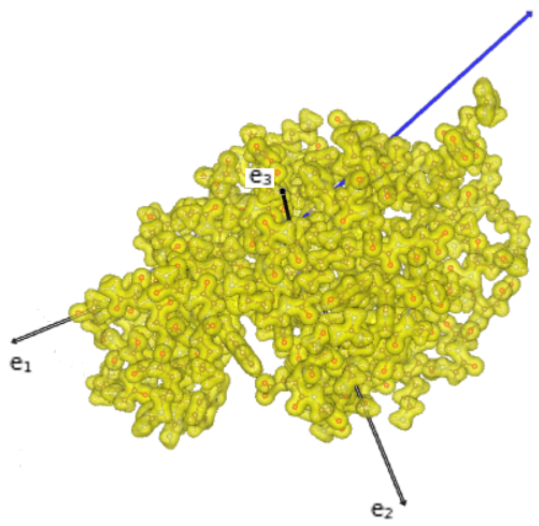
(d)

Figure 11: Linearity test for the polarizability of **ubiquitin**. The data points come form the SIESTA simulations with field angles ($\beta = 268°, \gamma = 100°$) and ($\beta = 90°, \gamma = 60°$), for field strengths 0, 0.1, 0.5 and 1 V/nm. These cases correspond to the field parallel with the intrinsic dipole and the field inducing one of the largest dipoles. (a) Magnitude of the total dipole (in Debye) as a function of electric field strength (in V/nm) for the two cases, along with linear fits. (b) Magnitude of the induced dipole (in Debye) as a function of electric field strength (in V/nm), along with linear fits. (c) Same data points as (a) but with a quadratic fit for the ($\beta = 90°, \gamma = 60°$) case. (d) Same data points as (b) but with a quadratic fit for the ($\beta = 90°, \gamma = 60°$) case.

Next some of the results corresponding to lysozyme are displayed in the same order as previously done for ubiquitin. The setup of lysozyme in figure 12, the total dipole and induced dipole in figure 13 and finally the angular deviations in figure 14.



(a)



(b)

Figure 12: Electron density of **lysozyme** in zero electric field, visualized using VESTA [27] with isosurface level 0.417518 electrons/Å^3, for (a) front view and (b) top view. Black arrows correspond to the BRF basis, dark blue arrow is the intrinsic dipole $\vec{\mu}_0$. See figure 5 for the definition of the azimuthal angle β and polar angle γ of the electric field $\vec{E}$ in the BRF.

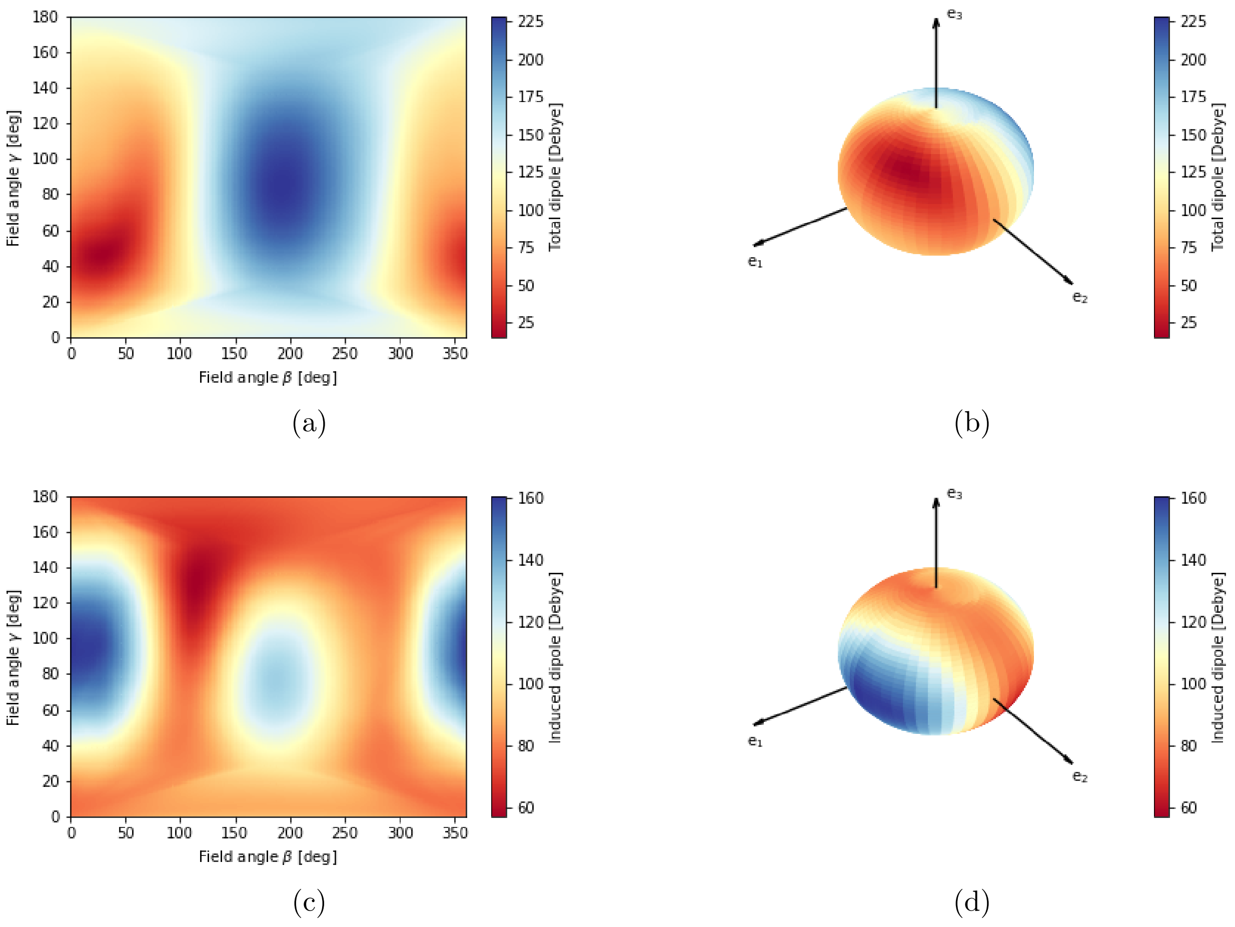


Figure 13: Results from the analysis of SIESTA simulations for **lysozyme** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Magnitude of the total electric dipole (in Debye) as a function of the field angles (β, γ) as defined in figure 12. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Magnitude of the induced dipole (in Debye) as a function of field angles, visualized on a sphere in (d). For comparison, the magnitude of the intrinsic dipole was $|\vec{\mu}_0| = 105.9$ Debye, and its direction was given by $(\beta = 202°, \gamma = 109°)$.

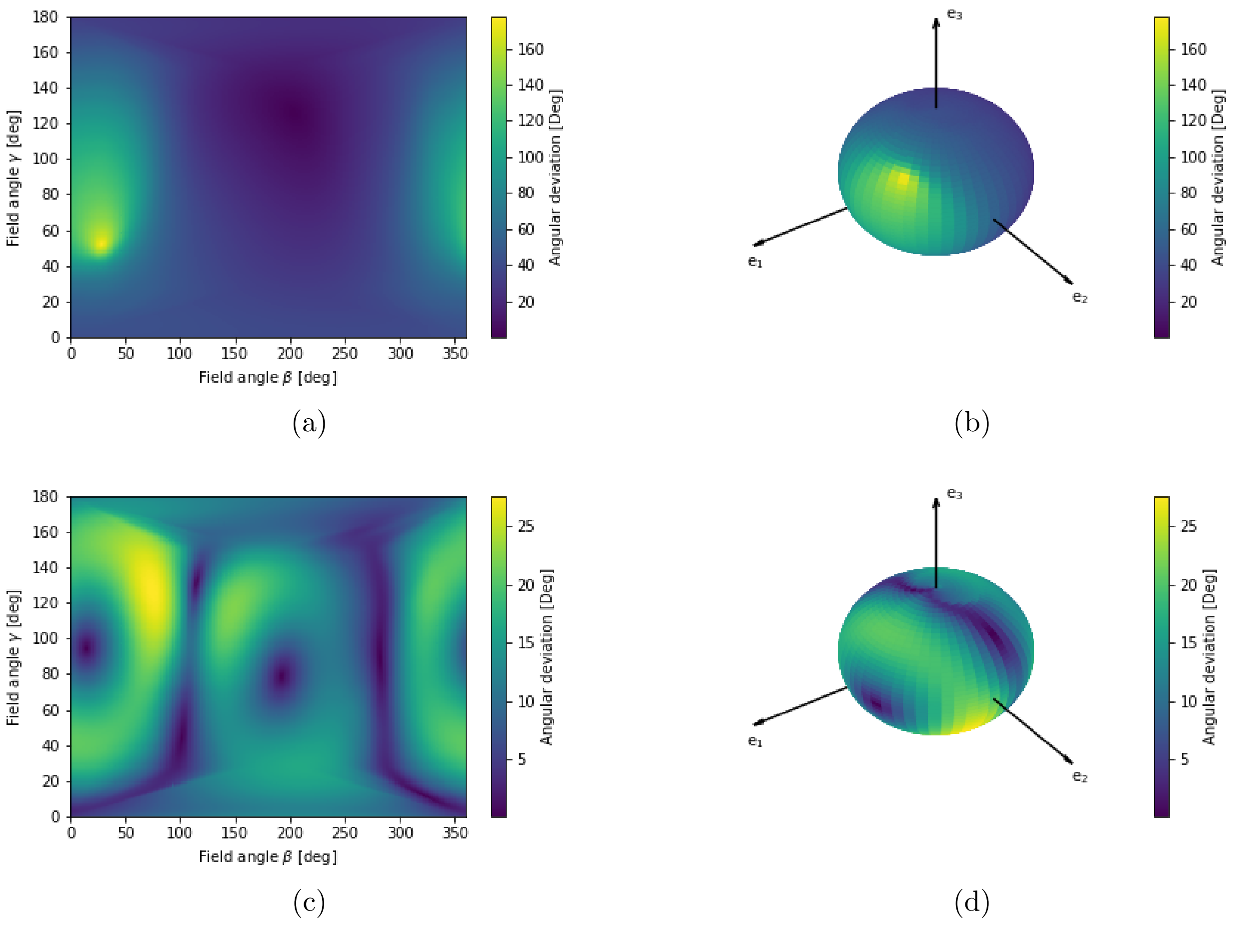


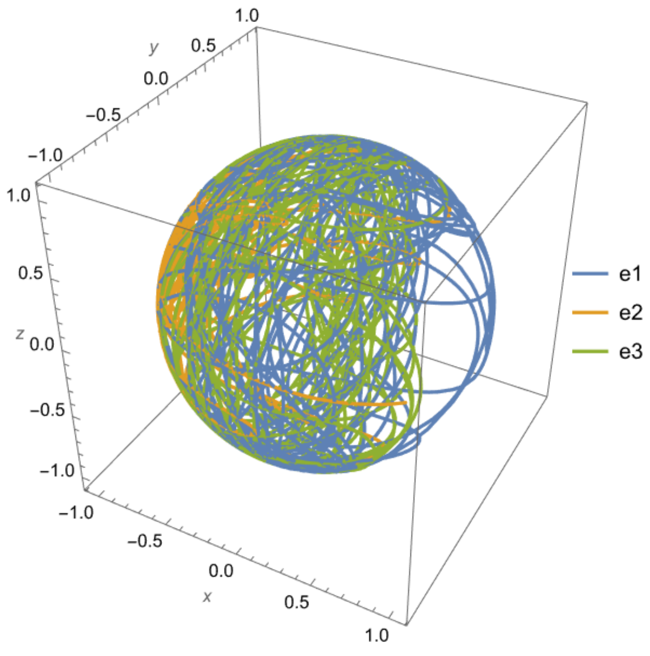
Figure 14: Results from the analysis of SIESTA simulations for **lysozyme** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Angle between the total dipole and the intrinsic dipole (in degrees) as a function of the field angles (β, γ) as defined in figure 12. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Angle between the induced dipole and the applied field as a function of (β, γ), visualized on a sphere in (d).

4.2 3D rigid rotor model

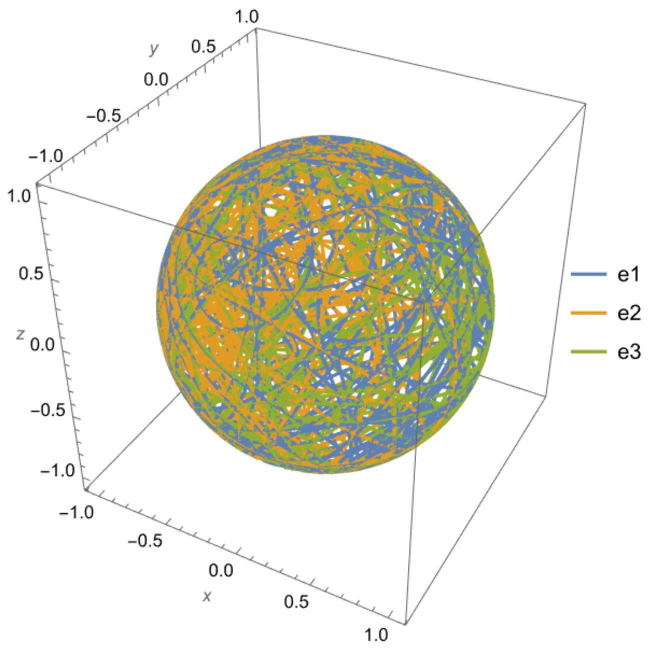
Here a selection of results from the protein orientation dynamics model is presented, obtained with the method described in section 3.2. The 3D rigid rotor model was developed in Mathematica. The script can be found in appendix C.3. The following results correspond to simulations of ubiquitin in a static electric field of strength 1 V/nm, in the x-direction of the FRF, with moment of inertia $I_1 = 8701.1, I_2 = 7250.6, I_3 = 6310.1$ $[\mathrm{m}_u\mathrm{nm}^2]$ and intrinsic dipole $\vec{\mu}_0 = (-2.251, -82.31, -14.82)$ Debye in the BRF. Initial conditions for the Euler angles and their velocities were set to

$$\phi_0 = 4\pi/5, \;\; \theta_0 = \pi/4, \;\; \psi_0 = \pi/2$$
$$\dot{\phi}_0 = 0, \;\; \dot{\theta}_0 = 0, \;\; \dot{\psi}_0 = 0$$

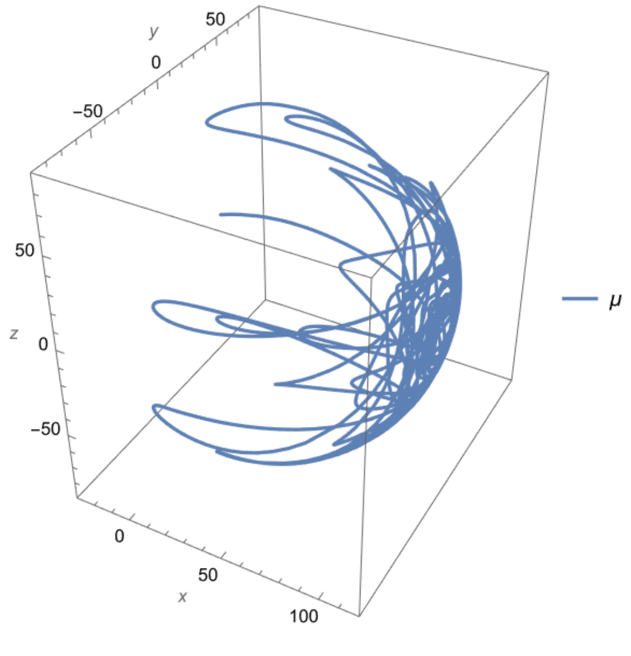
where one simulation was run with polarizability effects, and the other without. On a personal laptop these simulations were executed in 20 seconds and 3 seconds respectively.



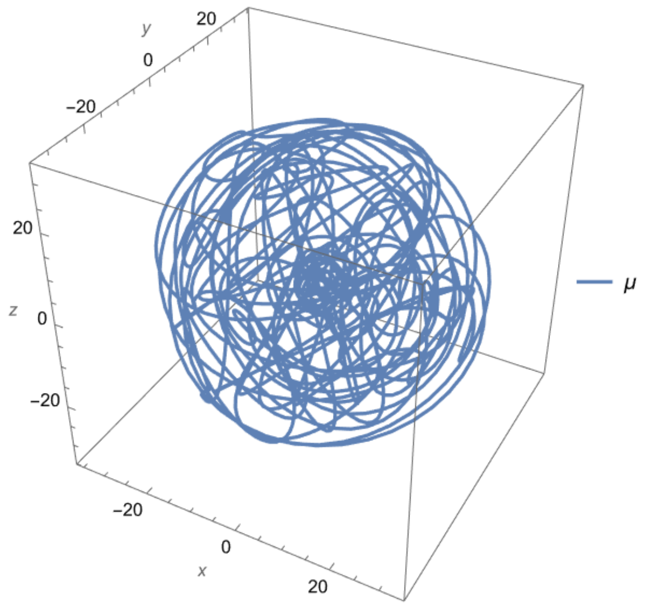
(a) With polarizability



(b) Without polarizability

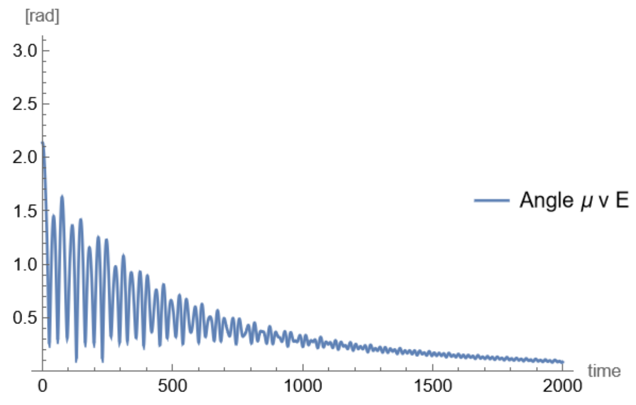


(c) With polarizability

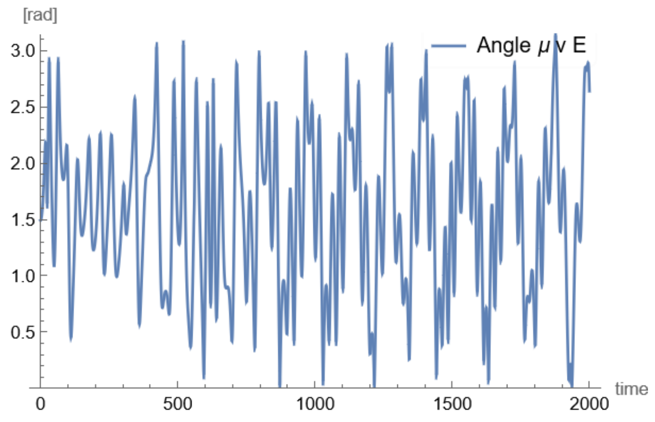


(d) Without polarizability

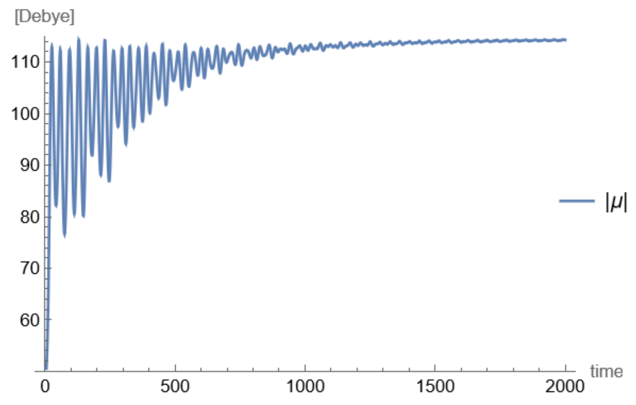
Figure 15: Results from the 3D rigid rotor model for **ubiquitin** in a static electric field of 1 V/nm, in the x-direction of the fixed reference frame (FRF), for the same set of initial conditions. On the left including the polarizability effects and on the left without polarizability, that is only with a static dipole corresponding to the intrinsic dipole of ubiquitin. (a) and (b) show the paths of the arrowheads of the body reference frame (BRF) basis vectors, $\hat{e}_1, \hat{e}_2$ and $\hat{e}_3$, plotted in the FRF for all 2000 time-steps (≈ 0.14 ns). (c) and (d) visualize the path of the electric dipole arrowhead in the FRF.



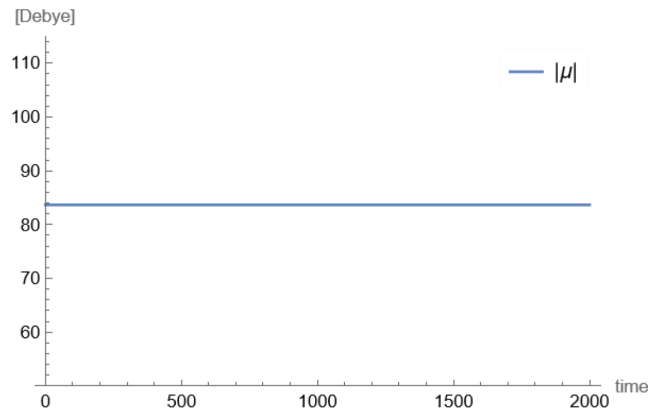
(a) With polarizability



(b) Without polarizability



(c) With polarizability



(d) Without polarizability

Figure 16: Results from the 3D rigid rotor model for **ubiquitin** in a static electric field of 1 V/nm, in the x-direction of the fixed reference frame (FRF), for the same set of initial conditions. On the left including the polarizability effects and on the left without polarizability, that is only with a static dipole corresponding to the intrinsic dipole of ubiquitin. (a) and (b) show the angle between the dipole and the field (in radians) as a function of time. (c) and (d) the magnitude of the dipole (in Debye) as a function of time. The simulation time of 2000 time-steps approximately corresponds to 0.14 ns.

5 Discussion

There were two main goals of this thesis. First to study how the electronic polarizability of the small proteins ubiquitin, Trp-cage and lysozyme depend on the orientation of the applied static electric field. Second to investigate whether the polarizability affects the orientation dynamics of the proteins, by developing rotor model for a rigid body and comparing simulations with and without induced dipoles. The motivation behind this is to gain a deeper understanding of the process of orienting proteins in electric fields, as previous research has mainly considered classical molecular dynamics without taking the polarizability into account. Improving our models and understanding of protein orientation has implications in various experimental techniques, such as single-particle imaging. In this section the limitations of this project will be discussed, followed by an analysis of the presented results and possible improvements for the future.

5.1 Limitations

It is important to recognize the limitations of the project, in order to bear them in mind when analyzing the results and drawing conclusions. First of all, approximations and assumptions are always made to some extent when a system is modeled. In the SIESTA simulations, for instance, the motion of the nuclei is neglected, only the movement of the electrons as a response to the applied electric field is considered in the parametrization of the polarizability. Additionally, the model proteins are simulated in perfect vacuum, while in experiments (such as SPI) there can for example be water molecules around it after the electrospray process. In other relevant techniques, such as Ion mobility spectrometry, the proteins can be in a gaseous environment as well. While it should be studied further, one might expect that surrounding water molecules would enhance the polarizability effect. Moreover, since the model proteins are assumed to be rigid bodies in the 3D rotor model, there is no damping included that would correspond to the absorption of energy in internal vibrations. This is something that should be added to the model by a damping parameter, but it is not considered in the results presented here. These points suggest that the setup here is a "extreme case scenario" and that the degree of orientation might be even better in more realistic conditions, with damping from internal energy conversion and enhanced polarizability from the surroundings.

As mentioned above, the proteins are assumed to be rigid bodies in the rotor model. This is not true in real life, and it would be relevant to investigate to what extent the non-rigidity affects the orientation process. Furthermore, the rotor model is built such that the induced dipole is instantaneously added to the intrinsic dipole at each point in time. This is an approximation, but it can be motivated by considering that the timescale of the electron displacements is much faster than the macroscopic rotation of the protein. In general, the polarizability calculations will of course also depend on the limitations of DFT, and the SIESTA program itself. Some of these approximations and assumptions were mentioned in section 2.4. The field orientations were sampled in steps of 30 degrees and then interpolated, as a part of the analysis of the simulation output performed in Python, which is a limitation as well. To estimate how much the interpolation varied from an actual simulation result, this was calculated for a field orientation between two data points. Table 5 shows the results of this sample test, where the relative error was only about 0.15 %. While this was only one sample point, it suggests that the interpolation was quite accurate and that the extra time to run simulations in smaller angle steps might not be worth it. The point of this project is, after all, to investigate trends rather than quantitative results.

Another limitation, connected to the motivation behind this study, is that the field strength 1 V/nm is larger than the typical devices produce in relevant experiments. One point to consider is that the added induced dipole could, in principle, be scaled down to correspond to lower field strengths. This would be a suitable method if the polarizability behaved linearly in the region between 1 V/nm and 0 V/nm. Two different field orientations were chosen to test this for ubiquitin, sampling field strengths 0.5 and 0.1 V/nm in addition to the previous simulations. These field strengths were chosen to provide two data points below 1 V/nm, such that a linear fit to the data could be tested in a simple way. While that is a small sample size, the expected "extreme" case where the protein was most polarizable was one of the considered orientations. Figure 11 suggests that linear polarizability is a good approximation when the field is parallel with the intrinsic dipole. For the other case a quadratic fit was preferred, suggesting that the polarizability should include at least second-order terms. In those cases the polarizability effect could not simply be scaled down linearly. Another relevant point in this discussion is that higher field strengths, such as 1 V/nm, could be achieved in practice with for instance an oscillating field. It would have to be low enough frequency to avoid complicated expressions for the polarizability, it is not the same in a rapid oscillating field as was considered here, but switching fast enough to not have the time to accelerate the proteins into the walls of the device.

5.2 Analysis of results

Analyzing the polarizability results presented in section 4.1, it is clear that the polarizabilities of the model proteins are not spherically symmetric. The magnitude of the total dipole had a similar trend for all three proteins, it was smaller than the intrinsic dipole in one region and larger in the other. These regions were more or less two halves of the sphere. Both the magnitude of the intrinsic dipole and the range of magnitudes of the total dipole seemed to increase with the size of the protein. Smallest for Trp-cage, then ubiquitin and largest for lysozyme. Looking at the magnitude of the induced dipole only, more differences between the proteins were revealed. Regions where the induced dipole is larger seems to correspond to the structure of the protein, and the effect is significant enough in these cases to differentiate the proteins. For example ubiquitin, the blue region in figure 8 (c) and (d) appears to be connected to the tail in the bottom left corner in figure 7 (b). The same trend is seen in figure 9 (c) and (d), where the induced dipole is up to around 14 degrees away from the applied field in a circle around the tail. This can be interpreted as a field applied exactly along the tail will polarize the protein along the tail, but a field applied in a ring around the tail will still polarize it mostly along the tail. An example of this feature is visualized in figure 10, where the field is more or less along the tail in (a) and closely around the tail in (b).

This feature is found in lysozyme and Trp-cage as well, but they have two such regions corresponding to two different "tails". This result is a sign that the polarizability of these proteins are anisotropic, and related to rather small, but observable, details in the structure. Comparing the magnitudes of the induced dipole, the results suggest that the induced dipole increases with the size of the protein. Moreover, the anisotropic effects seem to be more dependent on the varying structure of the protein rather than its size. The maximum angle between the induced dipole and the field was largest for Trp-cage, reaching around 30 degrees, while the corresponding values for ubiquitin and lysozyme were approximately 14 and 25 degrees. One factor could be that the tails in Trp-cage become more prevalent, the tail is larger relative to the rest of the small protein. However, the effect is significant in lysozyme as well, suggesting that another factor could be the distinctiveness of the bulging parts of the structure. In lysozyme, see figure 12, the "tails" stick out with quite clear "holes" between. How this effect turns out for much larger proteins, which is typically the point of interest in real life experiments, remains to be explored. Perhaps these "tails" are less likely to be seen, but the total dipole might still vary in two regions as was seen here. This could affect the orientation dynamics, which will be discussed next.

Despite only presenting the results of the 3D rigid rotor model for one set of initial conditions, see section 4.2, the chosen example illustrates the principle and potential of the method. Keep in mind that the results may vary significantly depending on the initial conditions, as well as the values related to the protein - the inertia, the dipole and the polarizability in that specific field strength. That being said, this particular case clearly suggests that the polarizability has a damping effect on the protein's orientation dynamics. Figures 15 and 16 both illustrate the difference between having the added induced dipole at each point in time (which depends on the field orientation with respect to the BRF) and only including the intrinsic dipole (as was the case in previous research such as the MD simulations in Marklund et al. [5]). The difference is perhaps most evident in figure 16, where the angle between the dipole and the field is quite rapidly damped in the case with polarizability and undamped in the case without. This alignment of the dipole along the field is the desired effect in protein orientation, and these results suggest that the polarizability might improve the orientation dynamics. If an additional damping was included, corresponding to an absorption in internal vibrations, as well as the polarizability, the orientation process might be even more efficient than previously considered in purely classical simulations.

5.3 Future improvements

The reason why a damping effect is observed in the case with polarizability could be due to the varying magnitude of the total dipole, as can be seen for ubiquitin in figure 8. A consequence is that the torque, which drives the rotation, is smaller in certain orientations of the protein in the field and larger in others. One way to interpret this is that the rotational energy is absorbed by the movement of the electrons as the protein polarizes, allowing the motion to be damped. It is important to note that this model is new and has not been sufficiently tested yet. The numerical stability is one of the aspects that should be tested in the future. Both of the model without polarizability, and with the effects of polarizability by for instance slowly turning up the added induced dipole. It would also be relevant to test the model for an even smaller molecule, such as water, so that the results can be compared with a time-dependent DFT simulation. Moreover, the timescale of the oscillations could be compared to MD simulations as one indication of the credibility of the results presented here.

If the rotor model turns out to be stable and plausible it could be very advantageous in research on proteins, and perhaps other polarizable molecules, in electric fields. It would be an efficient alternative to time consuming MD simulations, while still providing complete information about the 3D rotation. The model only depends on the inertia tensor, the intrinsic dipole and information about the polarizability. However, to draw any overall conclusions many different initial conditions has to be simulated to perform a statistical analysis. This is more reasonable to do with this model, compared with MD simulations, since the execution time is in the order of seconds. To include the polarizability in a simple and accessible way, for an arbitrary protein, the polarizability of common protein structures (α-helix, β-sheet) could perhaps be studied separately and combined. In this way one could obtain "puzzle-pieces" that could be used to build up an estimate of the polarizability of any protein structure.

6 Conclusions and outlook

Ground state quantum simulations of the small proteins ubiquitin, Trp-cage and lysozyme were performed in vacuum, using the SIESTA program, to investigate their polarizability. The electric dipole was extracted for varying orientations of an applied electric field with strength 1 V/nm. Analyzing the results in a Python program revealed connections between the polarizability and the structure, as well as the size, of the proteins. Subsequently a 3D rigid rotor model was developed in Mathematica. This model solves Euler's equations for a protein, assumed to be a rigid body, in a static electric field. It returns information about the orientation dynamics, which is driven by the torque a dipole experiences in an external electric field. Simulations of ubiquitin, for a chosen set of initial conditions, with and without the effect of polarizability were carried out. The results suggested that the polarizability has a damping effect on the orientation dynamics, at least for certain initial conditions. This is a promising indication that proteins might orient easier due to their polarizability, something that was not considered in previous research where classical molecular dynamics was used to model the orientation. Improving our understanding and modeling of this process has implications in single-particle imaging, and possibly other experiments that involve biomolecules in electric fields.

Since the method presented in this project is still new and fairly untested, there are many possibilities of improvements in future studies. First of all, the 3D rotor model should be investigated in more detail by for instance checking the numerical stability, both with and without polarizability effects. It would also be relevant to compare the results with molecular dynamics simulations of the same protein, or time-dependent quantum mechanical simulations of a smaller molecule such as water. Additionally, if the plausibility of the rotor model is strengthened, many simulations with varying initial conditions should be performed in order to make a statistical analysis of the results. For instance by considering average degrees of orientation. The model could also be extended to include oscillating fields. Another next step of this project could be to investigate the polarizability of common protein structures, for example α-helix and β-sheet, both separately and combined. This information could perhaps be used to estimate the polarizability of any protein, without performing new time-consuming quantum simulations each time. Combined with the rigid rotor model this could be a general, effective and accessible method of simulating protein orientation in electric fields.

Acknowledgements

I would like to thank my supervisors Oscar and Calle, as well as Nic, for all their support and helpful feedback! Also a big thank you to Emiliano and Thomas for the extra help and rewarding collaboration. To the whole research group, thank you for all the Monday meetings and social events. Finally I would like to thank my friends and family for their support.

References

[1] Daniel E. Koshland and Felix Haurowitz. Protein. *Encyclopædia Britannica*. `https://www.britannica.com/science/protein`. [Published 2020-12-01].

[2] Anna Sinelnikova, Thomas Mandl, Harald Agelii, Oscar Grånäs, Erik G. Marklund, Carl Caleman, and Emiliano De Santis. Protein orientation in time-dependent electric fields: orientation before destruction. *Biophysical Journal*, **120**(17):3709–3717, 2021.

[3] Janos Hajdu, Richard Neutze, Remco Wouts, David van der Spoel, and Edgar Weckert. Potential for biomolecular imaging with femtosecond X-ray pulses. *Nature (London)*, **406**(6797):752–757, 2000.

[4] Egor Sobolev, Serguey Zolotarev, Klaus Giewekemeyer, et al. Megahertz single-particle imaging at the European XFEL. *Communications Physics*, **3**(97), 2020.

[5] Erik G. Marklund, Tomas Ekeberg, Mathieu Moog, Justin L. P. Benesch, and Carl Caleman. Controlling protein orientation in vacuum using electric fields. *The journal of physical chemistry letters*, **8**(18):4540–4544, 2017.

[6] A. Aquila, A. Barty, C. Bostedt, Carini G. Boutet, S., et al. The linac coherent light source single particle imaging road map. *Structural dynamics (Melville, N.Y.)*, **2**(4):041701–041701, 2015.

[7] Oscar Grånäs, Nicusor Timneanu, Ibrahim Eliah Dawod, Davide Ragazzon, Sebastian Trygg, Petros Souvatzis, Tomas Edvinsson, and Carl Caleman. Femtosecond bond breaking and charge dynamics in ultracharged amino acids. *The Journal of chemical physics*, **151**(14):144307–144307, 2019.

[8] S. Vijay-Kumar, C. E. Bugg, and W. J. Cook. Structure of ubiquitin refined at 1.8 Å resolution. *Journal of molecular biology*, **194**(3):531–544, 1987.

[9] Fesinmeyer R. Neidigh, J. and N. Andersen. Designing a 20-residue protein. *Nature Structural Biology*, **9**:425–430, 2002.

[10] P. J. Artymiuk, C. C. F. Blake, D. W. Rice, and K. S. Wilson. The structures of the monoclinic and orthorhombic forms of hen egg-white lysozyme at 6 Å resolution. *Acta Crystallographica Section B*, **38**(3):778–783, 1982.

[11] Carl Nordling and Jonny Österman. *Physics handbook for science and engineering*. Studentlitteratur, Lund, 8., [rev.] edition, 2006.

[12] Dipole Moments. Libretexts; 2022 [Updated: May 1, 2022]. Available from:. `https://chem.libretexts.org/@go/page/1323`.

[13] Stephen T. Thornton and Jerry B. Marion. *Classical dynamics of particles and systems*. Brooks/Cole - Thomson learning, Belmont, Calif, 5. edition, 2004.

[14] David J. Griffiths. *Introduction to electrodynamics*. Prentice Hall, Upper Saddle River, N.J, 3rd, international edition, 1999.

[15] Paul N. Butcher and D. Cotter. *The elements of nonlinear optics*, volume 9. Cambridge University Press, Cambridge, New York, 1990.

[16] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical review*, **136**(3B):864–871, 1964.

[17] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, **140**(4A):1133–1138, 1965.

[18] Richard M. Martin. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press, 2004.

[19] JP Perdew, JA Chevary, SH Vosko, KA Jackson, MR Pederson, DJ Singh, and C. Fiolhais. Atoms, molecules, solids, and surfaces: Applications of the generalized gradient approximation for exchange and correlation. *Physical review. B, Condensed matter*, **46**(11):6671–6687, 1992.

[20] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical review letters*, **78**(7):1396–1396, 1997.

[21] José M. Soler et al. The SIESTA method for ab initio order-N materials simulation. *Journal of physics. Condensed matter*, **14**:2745, 2002.

[22] Siesta 4.1.5 *User's Guide*. `https://siesta-project.org/SIESTA_MATERIAL/Docs/Manuals/manuals.html`. [Published January 27, 2021].

[23] LucidChart. `https://www.lucidchart.com/`.

[24] Erik G. Marklund, Daniel S. D. Larsson, David van der Spoel, Alexandra Patriksson, and Carl Caleman. Structural stability of electrosprayed proteins: temperature and hydration effects. *Physical chemistry chemical physics : PCCP*, **11**(36):8069–8078, 2009.

[25] Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl. Gromacs 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of Chemical Theory and Computation*, **4**(3):435–447, 2008. PMID: 26620784.

[26] Pablo Ordejón Javier Junquera and Alberto García. *DENCHAR User's Guide*. `https://departments.icmab.es/leem/SIESTA_MATERIAL/Docs/Tutorials/zcam14/Manuals/denchar.pdf`. [Published 2012-05-04].

[27] Koichi Momma and Fujio Izumi. *VESTA3* for three-dimensional visualization of crystal, volumetric and morphology data. *Journal of Applied Crystallography*, **44**(6):1272–1276, 2011.

[28] Wolfram Research, Inc. Mathematica, Version 13.0.0. Champaign, IL, 2021.

[29] Serge Belongie. "Rodrigues' Rotation Formula." From *MathWorld*–A Wolfram Web Resource, created by Eric W. Weisstein. `https://mathworld.wolfram.com/RodriguesRotationFormula.html`.

A Additional background

A.1 Angles and rotations in 3D

The angle α between two 3D vectors is given by

$$cos(\alpha) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{a_x b_x + a_y b_y + a_z b_z}{|\vec{a}||\vec{b}|} \tag{16}$$

where $\vec{a} = (a_x, a_y, a_z)$ and $\vec{b} = (b_x, b_y, b_z)$ [11].

Rodrigues' rotation formula [29] gives the 3D rotation matrix R corresponding to a rotation by the angle θ about a fixed axis defined by the unit vector $\vec{u} = (u_x, u_y, u_z)$. The rotation matrix is given by the following expression.

$$R_u(\theta) = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_y\sin\theta + u_x u_z(1-\cos\theta) \\ u_z\sin\theta + u_x u_y(1-\cos\theta) & \cos\theta + u_y^2(1-\cos\theta) & -u_x\sin\theta + u_y u_z(1-\cos\theta) \\ -u_y\sin\theta + u_x u_z(1-\cos\theta) & u_x\sin\theta + u_y u_z(1-\cos\theta) & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix} \tag{17}$$

A.2 Rotation matrix between two coordinate systems

To find the 3D rotation matrix R that rotates between the FRF (lab frame with basis (1,0,0), (0,1,0) and (0,0,1)) and the BRF (with basis $\hat{e}_1, \hat{e}_2, \hat{e}_3$), the matrix A is set up such that the following matrix equation provides the desired components of R.

$$A = \begin{bmatrix} e_1[0] & e_1[1] & e_1[2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_1[0] & e_1[1] & e_1[2] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_1[0] & e_1[1] & e_1[2] \\ e_2[0] & e_2[1] & e_2[2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_2[0] & e_2[1] & e_2[2] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_2[0] & e_2[1] & e_2[2] \\ e_3[0] & e_3[1] & e_3[2] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & e_3[0] & e_3[1] & e_3[2] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & e_3[0] & e_3[1] & e_3[2] \end{bmatrix} \tag{18}$$

If B is defined to be $B = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ and the unknown X is defined as $X = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{21} & R_{22} & R_{23} & R_{31} & R_{32} & R_{33} \end{bmatrix}$, then the matrix equation to solve is:

$$XA = B \implies X = BA^{-1} \tag{19}$$

and the rotation matrix R between the coordinate systems is then given by the components of X in the following way.

$$R = \begin{bmatrix} X[0] & X[1] & X[2] \\ X[3] & X[4] & X[5] \\ X[6] & X[7] & X[8] \end{bmatrix} \tag{20}$$

B Trp-cage polarizability results

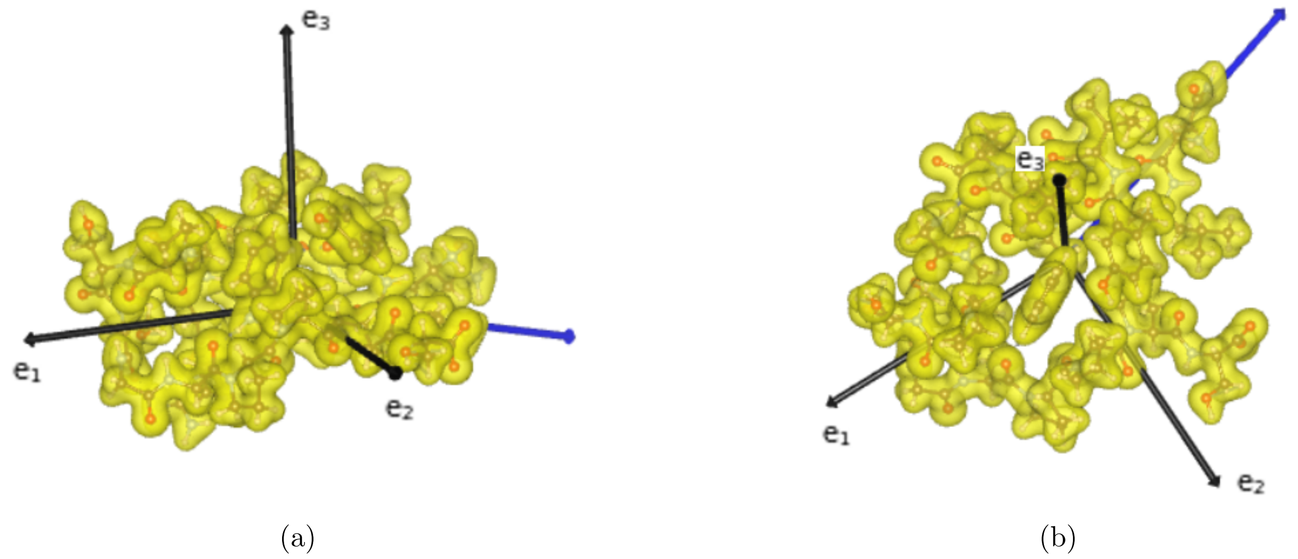
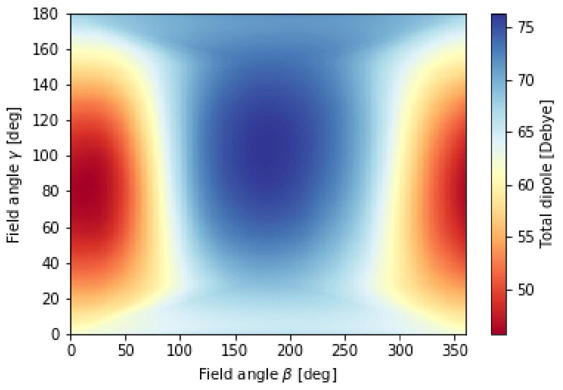
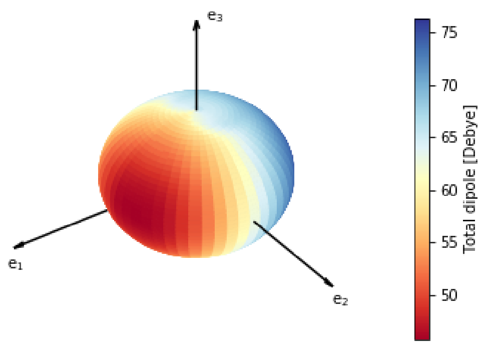


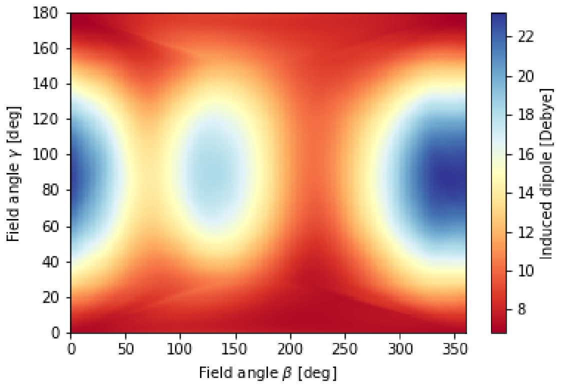
Figure 17: Electron density of **Trp-cage** in zero electric field, visualized using VESTA [27] with isosurface level 0.291587 electrons/Å^3, for (a) front view and (b) top view. Black arrows correspond to the BRF basis, dark blue arrow is the intrinsic dipole $\vec{\mu}_0$.



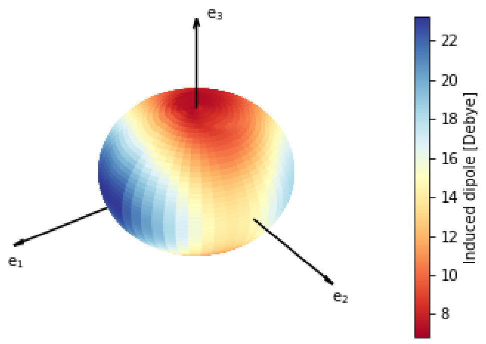
(a)



(b)



(c)



(d)

Figure 18: Results from the analysis of SIESTA simulations for **Trp-cage** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Magnitude of the total electric dipole (in Debye) as a function of the field angles (β, γ) as defined in figure 5. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Magnitude of the induced dipole (in Debye) as a function of field angles, visualized on a sphere in (d). For comparison, the magnitude of the intrinsic dipole was $|\vec{\mu}_0| = 64.90$ Debye, and its direction was given by ($\beta = 201°, \gamma = 107°$).

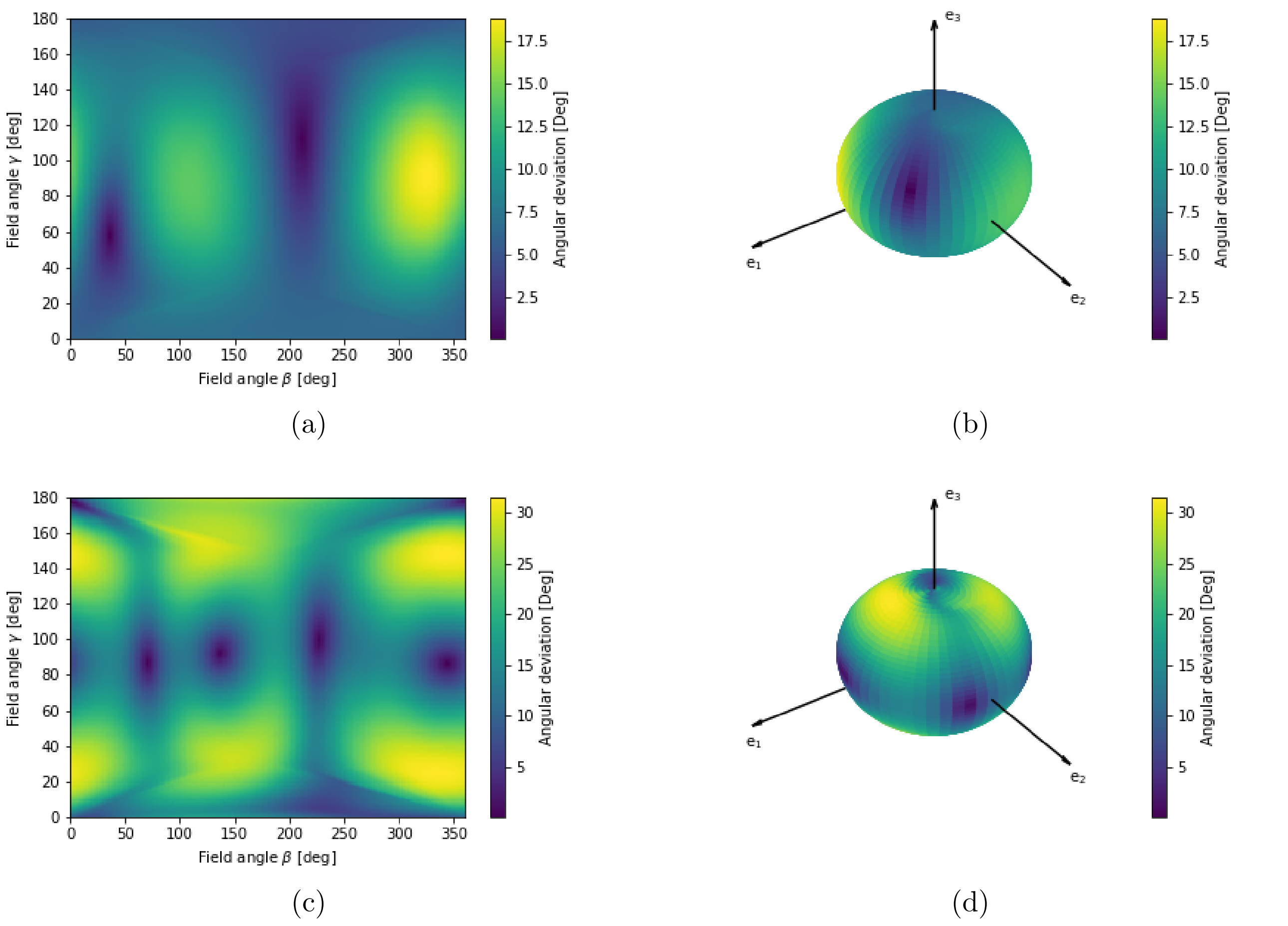


Figure 19: Results from the analysis of SIESTA simulations for **Trp-cage** in a 1 V/nm strong electric field with varying orientations with respect to the BRF. (a) Angle between the total dipole and the intrinsic dipole (in degrees) as a function of the field angles (β, γ) as defined in figure 5. (b) Data of (a) visualized on a sphere along with the BRF axes. (c) Angle between the induced dipole and the applied field (in degrees) as a function of field angles, visualized on a sphere in (d).

C Scripts

C.1 Outline of SIESTA input

```
ElectronicTemperature 300.0 K
SystemName UBI
SystemLabel UBI


PAO.BasisSize TZ3P
MeshCufoff 100 Ry
PAO.EnergyShift        0.005 Ry


NetCharge 7


XC.functional        GGA
XC.authors        PBE


DM.NumberPulay 5
DM.MixingWeight 0.05


Write.Denchar .true.


MaxSCFIterations 500
number and species


UseSaveData True
WriteDM.NetCDF true


PartialChargesAtEveryGeometry .true.


LatticeConstant 44.00 Ang


%block LatticeVectors
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000
%endblock LatticeVectors


# ——————————————————————————-
# Species and Atoms
# ——————————————————————————-
```

```
#number and species
NumberOfSpecies 5
NumberOfAtoms 1238

%block ChemicalSpeciesLabel
1 8 O
2 1 H
3 7 N
4 6 C
5 16 S
%endblock ChemicalSpeciesLabel

#————————-
# FIELD
#————————-

%block ExternalElectricField
0 0 0 V/nm
%endblock ExternalElectricField

# ——————————————————————————-
# Atomic Coordinates
# ——————————————————————————-

AtomicCoordinatesFormat Ang

%block AtomicCoordinatesAndAtomicSpecies
-7.260000 -6.280000 1.240000     3
-8.470000 -5.480000 1.480000     4
-8.380000 -4.410000 0.440000     4
-7.300000 -3.850000 0.280000     1
⋮

%endblock AtomicCoordinatesAndAtomicSpecies
```

C.2 Python code for polarizability analysis

```python
# Polarizability and Orientation Dynamics of Small Proteins
# (Master project - Department of Physics and Astronomy - Uppsala University)

# Author: Ebba Koerfer Supervisors: Oscar Granas and Carl Caleman Spring 2022

# A script to analyze the polarizability of proteins ubiquitin, Trp-cage and lysozyme,
#using the resulting dipoles from SIESTA simulations (given in the fixed reference frame
# (FRF)) for different angles of the field w.r.t a body fixed reference frame (BRF).
# Import all the functions from the library "polarizability_lib" (also written
# by Ebba Koerfer), which are used in the analysis

from polarizability_lib import *
import numpy as np

# Pick a protein to analyze, either "UBI" for ubiquitin, "TRP" for Trp-cage or
# "LYS" for lysozyme. All following calculations and plots
# will correspond to that chosen protein.

protein = "LYS"

# Defines some important values for each protein, the BRF (e1, e2, e3) from the
# inertia principal axes. The resulting intrinsic dipole in the FRF (from SIESTA),
# along with the (beta, gamma) corresponding to the parallel case and
# another case for the linearity test (also) the resulting dipoles from
# these simulations. All the resulting total dipoles for 1 V/nm is given
# in a dipole dictionary.

if protein == "UBI":
    e1 = np.array([0.13793203, 0.80497851, 0.57704797])
    e2 = np.array([-0.59855185, 0.53193899, -0.59897963])
    e3 = np.array([-0.78912004, -0.26277465, 0.5551928])
    p_intr = np.array([60.646670, -41.700274, 39.773621]) # TZ3P basis set
    par_beta, par_gamma = 268, 100
    pol_beta, pol_gamma = 90, 60
    p_par01 = np.array([62.932112, -43.193662, 41.204690]) # Parallel 0.1 V/nm
    p_par05 = np.array([72.074084, -49.165539, 46.927405]) # Parallel 0.5 V/nm
    p_pol01 = np.array([62.932112, -43.193662, 41.204690]) # (beta=90, gamma=60) 0.1 V/nm
    p_pol05 = np.array([72.074084, -49.165539, 46.927405]) # (beta=90, gamma=60) 0.5 V/nm
    p_pol = np.array([83.502207, -56.630534, 54.079499]) # (beta=90, gamma =60) 1 V/n

    # For total dipole, intrinsic+induced, in FRF in 1 V/nm, angles: (beta, gamma) in deg
    dipole_dict = {(0,0): (24.570574, -47.329663, 55.783507),
                   (0,90): (64.497768, -19.350720, 56.036647),
                   (0,180): (85.356193, -34.090139, 23.046963),
                   (90,90): (29.872375, -23.677322, 21.015129),
                   (268,100): (83.502207, -56.630534, 54.079499),
                   (180,90): (56.789657, -64.040370, 23.504809),
                   (270,90): (79.535172, -57.634192, 57.769426),
                   (0,30): (33.025284, -35.687341, 61.872908),
                   (0,60): (50.453390, -25.943733, 62.144765),
                   (0,120): (76.337897, -18.542459, 45.497956),
                   (0,150): (83.971421, -23.936349, 33.422558),
                   (30,30): (22.728799, -32.278943, 55.956716),
                   (30,60): (33.504827, -20.172996, 51.938893),
                   (30,90): (53.113239, -14.122591, 44.769148),
                   (30,120): (67.712003, -14.233256, 35.816326),
                   (30,150): (78.992494, -21.448985, 27.832299),
                   (60,30): (14.439216, -32.746143, 49.437191),
                   (60,60): (19.308392, -20.997080, 40.647685),
                   (60,90): (37.566943, -15.198528, 31.765412),
                   (60,120): (60.496028, -16.262958, 24.950619),
                   (60,150): (74.829948, -22.620877, 21.557736),
                   (90,30): (10.475258, -36.975896, 44.060255),
                   (90,60): (12.468843, -28.321712, 31.331154),
                   (90,120): (56.218846, -24.015183, 15.785636),
                   (90,150): (72.600568, -27.138588, 16.281164),
                   (120,30): (11.968917, -43.840255, 41.268626),
                   (120,60): (15.046149, -40.208508, 26.499441),
                   (120,90): (32.771276, -37.389257, 15.434839),
                   (120,120): (57.140488, -35.612835, 10.854155),
                   (120,150): (72.901157, -33.792218, 13.418761),
                   (150,30): (18.492082, -51.491503, 41.814732),
                   (150,60): (26.253207, -53.451549, 27.447338),
                   (150,90): (45.128620, -52.600195, 16.503730),
                   (150,120): (61.928496, -47.751190, 11.408438),
                   (150,150): (75.652670, -40.800162, 13.739262),
                   (180,30): (28.172801, -57.866379, 45.547790),
                   (180,60): (42.491293, -64.419828, 33.891474),
                   (180,120): (69.660111, -57.242271, 17.319351),
                   (180,150): (80.116300, -46.279871, 17.151986),
                   (210,30): (38.263931, -61.242349, 51.458334),
                   (210,60): (53.576381, -69.158982, 43.731110),
                   (210,90): (66.751880, -69.012615, 34.687799),
                   (210,120): (78.288745, -61.550538, 27.007302),
                   (210,150): (85.098665, -48.769947, 22.745706),
                   (240,30): (45.661397, -60.635265, 57.932859),
                   (240,60): (60.784023, -67.126475, 54.590701),
                   (240,90): (75.076169, -66.666653, 47.228930),
                   (240,120): (85.497283, -59.520897, 37.871705),
                   (240,150): (89.261113, -47.597511, 29.019005),
                   (270,30): (48.552941, -56.232205, 63.246584),
                   (270,60): (64.645566, -59.302127, 63.719070),
                   (270,120): (89.358404, -51.697113, 47.000967),
                   (270,150): (91.490425, -43.079855, 34.291643),
                   (300,30): (47.856137, -49.510994, 66.079552),
                   (300,60): (64.123399, -47.778652, 68.671179),
                   (300,90): (78.932011, -44.330366, 63.488177),
                   (300,120): (88.836089, -40.175951, 51.954096),
                   (300,150): (91.189149, -36.426723, 37.152065),
                   (330,30): (42.363381, -42.021910, 65.585969),
                   (330,60): (59.359467, -35.645094, 68.122116),
                   (330,90): (73.426897, -30.315417, 62.854497),
                   (330,120): (84.071340, -28.039345, 51.404422),
                   (330,150): (88.436124, -29.417110, 36.833217),
                   (360,0): (24.570574, -47.329663, 55.783507),
                   (360,90): (64.497768, -19.350720, 56.036647),
                   (360,180): (85.356193, -34.090139, 23.046963),
                   (360,30): (33.025284, -35.687341, 61.872908),
                   (360,60): (50.453390, -25.943733, 62.144765),
                   (360,120): (76.337897, -18.542459, 45.497956),
                   (360,150): (83.971421, -23.936349, 33.422558)}

 elif protein == "TRP":
     e1 = np.array([0.8506957889, -0.5197643042, 0.0784965754])
     e2 = np.array([0.4867914617, 0.8353160620, 0.2555016577])
     e3 = np.array([-0.1983700842, -0.1791427284, 0.9636167288])
     p_intr = np.array([-56.298646, 14.875718, -28.665898]) # TZ3P basis set
     par_beta, par_gamma = 201, 107
     pol_beta, pol_gamma = 0, 90
     p_par01 = np.array([-57.740107, 15.610503, -29.135369]) # Parallel 0.1 V/nm
     p_par05 = np.array([-62.023135, 17.982792, -30.765662]) # Parallel 0.5 V/nm
     p_pol01 = np.array([-54.763507, 13.564893, -28.475801]) # (beta=0, gamma=90) 0.1 V/nm
     p_pol05 = np.array([-47.990610, 8.027665, -27.647194]) # (beta=0, gamma=90) 0.5 V/nm
     p_pol = np.array([-38.161480, 0.634642, -26.612329]) # (beta=0, gamma=90) 1 V/nm

     # For total dipole, intrinsic+induced, in FRF in 1 V/nm, angles: (beta, gamma) in deg
     dipole_dict = {(0,0): (-56.480822, 12.951841, -21.722560),
                   (0,90): (-38.161480, 0.634642, -26.612329),
                   (0,180): (-55.563821, 16.504663, -35.323031),
                   (90,90): (-51.518827, 28.746332, -25.000052),
                   (201,107): (-65.719597, 19.948655, -32.369107),
                   (180,90): (-65.608606, 25.284115, -29.339426),
                   (270,90): (-60.335598, 1.309316, -31.991142),
                   (0,30): (-48.171971, 6.249344, -21.377438),
                   (0,60): (-41.334511, 1.625289, -23.083426),
                   (0,120): (-40.252099, 3.589758, -30.669238),
                   (0,150): (-46.893745, 9.491160, -33.856896),
                   (30,30): (-48.144222, 10.670191, -20.658244),
                   (30,60): (-41.310020, 9.340190, -21.838654),
                   (30,90): (-38.079853, 9.524234, -25.161906),
                   (30,120): (-40.144027, 11.207183, -29.372904),
                   (30,150): (-46.821902, 13.826560, -33.103562),
                   (60,30): (-50.405895, 15.806705, -20.476520),
                   (60,60): (-45.577414, 18.405505, -21.451250),
                   (60,90): (-43.291936, 19.990841, -24.535196),
                   (60,120): (-44.590689, 20.148247, -28.738639),
                   (60,150): (-49.247429, 18.871183, -32.690979),
                   (90,30): (-54.171190, 20.167781, -20.849315),
                   (90,60): (-52.333405, 25.981683, -22.041058),
                   (90,120): (-51.869177, 27.694386, -28.939027),
                   (90,150): (-53.304089, 23.154241, -32.723133),
                   (120,30): (-58.231619, 22.493840, -21.650033),
                   (120,60): (-59.158529, 29.786083, -23.327393),
                   (120,90): (-58.964260, 32.660046, -26.232818),
                   (120,120): (-58.282275, 30.991288, -29.808365),
                   (120,150): (-57.348226, 25.316009, -33.205835),
                   (150,30): (-61.059803, 22.092500, -22.581124),
                   (150,60): (-62.877775, 28.462556, -24.621347),
                   (150,90): (-62.976158, 30.887319, -27.580142),
                   (150,120): (-61.920290, 29.463254, -30.974779),
                   (150,150): (-59.957536, 24.637235, -33.926188),
                   (180,30): (-62.458390, 19.273330, -23.496105),
                   (180,60): (-65.167814, 23.502586, -26.144203),
                   (180,120): (-64.189549, 24.655479, -32.493767),
                   (180,150): (-61.288742, 21.826728, -34.794730),
                   (210,30): (-62.615098, 14.980120, -24.257036),
                   (210,60): (-65.779189, 16.388167, -27.575407),
                   (210,90): (-66.358430, 17.196407, -31.023815),
                   (210,120): (-64.824665, 17.709092, -33.959133),
                   (210,150): (-61.614547, 17.785852, -35.634703),
                   (240,30): (-61.444915, 10.350257, -24.636061),
                   (240,60): (-64.052718, 8.621129, -28.351711),
                   (240,90): (-64.760014, 8.613745, -32.089241),
                   (240,120): (-63.481677, 10.356298, -34.927028),
                   (240,150): (-60.663200, 13.412027, -36.155363),
                   (270,30): (-58.638163, 6.368169, -24.417983),
                   (270,60): (-60.031102, 2.149852, -28.172423),
                   (270,120): (-59.552992, 4.002300, -34.891779),
                   (270,150): (-57.849713, 9.526624, -36.113673),
                   (300,30): (-54.453770, 3.894582, -23.582026),
                   (300,60): (-52.790252, -2.154426, -26.827352),
                   (300,90): (-52.022336, -3.591683, -30.607882),
                   (300,120): (-52.343492, -0.189711, -33.832293),
                   (300,150): (-53.636047, 7.139889, -35.562752),
                   (330,30): (-50.530466, 3.796164, -22.455068),
                   (330,60): (-45.685192, -2.516040, -24.908657),
                   (330,90): (-43.562586, -4.095816, -28.567187),
                   (330,120): (-44.924542, -0.519603, -32.288816),
                   (330,150): (-49.473830, 7.069661, -34.749737),
                   (360,0): (-56.480822, 12.951841, -21.722560),
                   (360,90): (-38.161480, 0.634642, -26.612329),
                   (360,180): (-55.563821, 16.504663, -35.323031),
                   (360,30): (-48.171971, 6.249344, -21.377438),
                   (360,60): (-41.334511, 1.625289, -23.083426),
                   (360,120): (-40.252099, 3.589758, -30.669238),
                   (360,150): (-46.893745, 9.491160, -33.856896)}

 elif protein == "LYS":
     e1 = np.array([-0.3703268170, 0.8485152721, -0.3779945076])
     e2 = np.array([-0.7241570354, -0.0088601764, 0.6895781755])
     e3 = np.array([0.5817685723, 0.5290966630, 0.6177394986])
     p_intr = np.array([41.647970, -96.515472, -12.891807]) # DZP basis set
     par_beta, par_gamma = 202, 109
     pol_beta, pol_gamma = 30, 90
```

```
p_par01 = np.array([48.786306, -108.135591, -12.693330]) # Parallel 0.1 V/nm
p_par05 = np.array([71.834257, -147.429935, -12.240006]) # Parallel 0.5 V/nm
p_pol01 = np.array([30.847233, -84.573607, -14.424807]) # (beta = 30, gamma = 90)
                                                        # 0.1 V/nm
p_pol05 = np.array([-12.110091, -34.325446, -21.955891]) # (beta=30, gamma=90)
                                                         # 0.5 V/nm
p_pol = np.array([-59.112187, 21.491878, -30.433317]) # (beta=30, gamma=90)
                                                      # 1 V/nm

# For total dipole, intrinsic+induced, in FRF in 1 V/nm, angles: (beta, gamma) in deg
dipole_dict = {(0,0): (86.706681, -58.337778, 40.184096),
              (0,90): (-35.142170, 34.959429, -63.641417),
              (0,180): (-3.859342, -123.812712, -67.420471),
              (90,90): (-21.650126, -70.781418, 37.244447),
              (202,109): (101.968295, -193.652062, -7.817917),
              (180,90): (105.930414, -196.372013, 33.643992),
              (270,90): (108.471776, -102.544393, -70.349939),
              (0,30): (45.870310, -0.803835, 5.465381),
              (0,60): (0.005256, 34.709925, -32.636147),
              (0,120): (-50.633660, -1.784938, -79.962706),
              (0,150): (-38.710965, -63.758668, -79.469921),
              (30,30): (32.175268, -7.601522, 24.619440),
              (30,60): (-22.153464, 23.528988, -3.333543),
              (30,90): (-59.112187, 21.491878, -30.433317),
              (30,120): (-68.544565, -10.835251, -52.569699),
              (30,150): (-48.146125, -64.319194, -66.931468),
              (60,30): (34.490749, -29.461203, 43.732318),
              (60,60): (-18.539426, -12.704326, 29.210309),
              (60,90): (-54.768516, -15.760689, 3.951982),
              (60,120): (-63.130699, -38.711320, -24.799138),
              (60,150): (-44.113095, -79.427499, -52.264097),
              (90,30): (48.467597, -57.930858, 59.280043),
              (90,60): (8.335125, -62.906591, 57.762032),
              (90,120): (-33.758689, -86.360334, 1.063627),
              (90,150): (-27.114731, -106.460770, -38.782752),
              (120,30): (68.459011, -86.781981, 66.667037),
              (120,60): (46.543580, -113.301057, 71.960099),
              (120,90): (25.363031, -130.296571, 53.594148),
              (120,120): (8.922548, -139.036833, 16.050586),
              (120,150): (-1.961772, -137.470341, -30.014433),
              (150,30): (92.680183, -110.214777, 66.303856),
              (150,60): (86.541984, -152.025393, 72.043209),
              (150,90): (70.182726, -175.726988, 53.089175),
              (150,120): (46.236761, -176.459775, 14.720327),
              (150,150): (18.990380, -156.347424, -30.557847),
              (180,30): (113.245235, -121.006495, 56.821039),
              (180,60): (120.158720, -169.639190, 54.573293),
              (180,120): (75.455555, -193.387579, -0.004407),
              (180,150): (35.150874, -164.827528, -38.235184),
                    (210,30): (126.060852, -116.047691, 37.987259),
                    (210,60): (137.636996, -160.106677, 24.856831),
                    (210,90): (123.949135, -186.304987, 1.515090),
                    (210,120): (90.478539, -186.426051, -26.174978),
                    (210,150): (43.527710, -162.372251, -52.237965),
                    (240,30): (127.633358, -97.648597, 17.190596),
                    (240,60): (139.016457, -128.102050, -10.193342),
                    (240,90): (125.711074, -150.185177, -38.402120),
                    (240,120): (91.655274, -157.379884, -59.832371),
                    (240,150): (43.268339, -148.877694, -69.220513),
                    (270,30): (116.718684, -69.238976, 0.739745),
                    (270,60): (124.092144, -85.686129, -38.869021),
                    (270,120): (76.762170, -119.963433, -84.510361),
                    (270,150): (35.280949, -129.604936, -82.278608),
                    (300,30): (96.475053, -36.449320, -7.594018),
                    (300,60): (90.072687, -30.208929, -52.827599),
                    (300,90): (70.182502, -44.094386, -82.741502),
                    (300,120): (41.795138, -73.800961, -95.131370),
                    (300,150): (12.444738, -102.767668, -89.165504),
                    (330,30): (70.671193, -11.885888, -6.348887),
                    (330,60): (43.082446, 14.671012, -51.930938),
                    (330,90): (13.845828, 11.796180, -83.151013),
                    (330,120): (-9.061542, -21.911084, -95.728436),
                    (330,150): (-15.612516, -78.034268, -87.783755),
                    (360,0): (86.706681, -58.337778, 40.184096),
                    (360,90): (-35.142170, 34.959429, -63.641417),
                    (360,180): (-3.859342, -123.812712, -67.420471),
                    (360,30): (45.870310, -0.803835, 5.465381),
                    (360,60): (0.005256, 34.709925, -32.636147),
                    (360,120): (-50.633660, -1.784938, -79.962706),
                    (360,150): (-38.710965, -63.758668, -79.469921)}

else:
    print("Invalid protein")

# Get the rotation matrix from FRF to BRF and vice versa. np.matmul(R, v)) takes
# v from FRF to BRF, np.matmul(np.linalg.inv(R), v) takes v from BRF to FRF

R = get_rotmatrix(e1, e2, e3)

print(f"Magnitude of intrinsic dipole: \t {magnitude(p_intr)} Debye")
print(f"Intrinsic dipole in FRF:\t {p_intr} (Debye) \nIntrinsic dipole in BRF:\t
       {np.matmul(R,p_intr)} (Debye) \n")

# Get a dictionary with all the electric field vectors (OBS in the FRF) for 1 V/nm.
# Keys are (beta, gamma).

EF_dict = get_EFdict(e1, e2, e3, p_intr, par_beta, par_gamma)
```

```
# Check the EF vectors in the FRF (for SIESTA input) for all the different angles,
# can look at one section from start to end.
# Starts with (beta, gammma) equal to (0, 0) and ends with (360,90).

start = 0
end = 0

check_fields(EF_dict, start, end, e1, e3, R)

# Get the interpolating functions for the EF vectors and dipoles respectively.
# (OBS in the BRF)

field_interpolation = get_interpolation(EF_dict, R)
dipole_interpolation = get_interpolation(dipole_dict, R)

# Plot the magnitude of the total dipole, the induced dipole, the angle between the
# intrinsic dipole and the total dipole, the angle between the induced dipole and
# the field - all as functions of (beta, gamma), all as heatmaps and on a sphere.

plot_dipoles(protein, p_intr, R, dipole_interpolation, field_interpolation)

# Plot the magnitude of the total dipole and the induced dipole as a function of
# EF strength, along with a linear fit, for two cases - when the field was
# parallel with the intrinsic dipole, and a chosen case where the polarizability
# was large, these dipoles are called "p_par" and "p_pol" respectively, for strengths
# 0.1, 0.5, 1 V/nm

plot_linearitytest(protein, p_intr, p_par01, p_par05,
                   np.array(dipole_dict[(par_beta, par_gamma)]), p_pol01, p_pol05,
                   p_pol, pol_beta, pol_gamma, par_beta, par_gamma)
```

Library: polarizability_lib.py

```
# Library of functions used in the polarizability analysis. Part of master thesis,
# written by Ebba Koerfer (spring 2022).
# Supervisors: Oscar Granas and Carl Caleman.

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from numpy import pi
from scipy.interpolate import LinearNDInterpolator
from scipy.interpolate import CloughTocher2DInterpolator
from matplotlib import cm, colors


def magnitude(vec):
    """Returns the magnitude of a 3D vector."""
    return np.sqrt(vec[0]**2 + vec[1]**2 + vec[2]**2)

def normalize(vec):
    """Takes a 3D vector and returns its normalized vector."""
    c = np.sqrt(vec[0]**2 + vec[1]**2 + vec[2]**2)
    return np.divide(vec,c)

def angle(vec1, vec2):
    """Returns the angle between two 3D vectors in degrees."""
    if vec1[0] == 0 and vec1[1] == 0 and vec1[2] == 0:
        return 0
    elif vec2[0] == 0 and vec2[1] == 0 and vec2[2] == 0:
        return 0
    else:
        return np.arccos((vec1[0]*vec2[0] + vec1[1]*vec2[1] +
               vec1[2]*vec2[2])/(magnitude(vec1)*magnitude(vec2))) *(180/pi)

def angle_rad(vec1, vec2):
    """Returns the angle between two 3D vectors in radians."""
    if vec1[0] == 0 and vec1[1] == 0 and vec1[2] == 0:
        return 0
    elif vec2[0] == 0 and vec2[1] == 0 and vec2[2] == 0:
        return 0
    else:
        return np.arccos((vec1[0]*vec2[0] + vec1[1]*vec2[1] +
                 vec1[2]*vec2[2])/(magnitude(vec1)*magnitude(vec2)))

def rot(vec, axis, angle):
    """Rotates a 3D vector 'vec' about the 3D vector 'axis' for 'angle' number
    of radians. Returns the rotated vector."""
    ux = axis[0]
    uy = axis[1]
    uz = axis[2]
    R = np.array([[np.cos(angle)+ux**2*(1-np.cos(angle)),
                  ux*uy*(1-np.cos(angle))-uz*np.sin(angle),
                  ux*uz*(1-np.cos(angle))+uy*np.sin(angle)],
                  [ux*uy*(1-np.cos(angle))+uz*np.sin(angle),
                   np.cos(angle)+uy**2*(1-np.cos(angle)),
                   uy*uz*(1-np.cos(angle))-ux*np.sin(angle)],
                  [uz*ux*(1-np.cos(angle))-uy*np.sin(angle),
                  uz*uy*(1-np.cos(angle))+ux*np.sin(angle),
                  np.cos(angle)+uz**2*(1-np.cos(angle))]])
    rotvec = R.dot(np.array(vec))
    return rotvec

def get_rotmatrix(Ix, Iy, Iz):
    """Returns a rotation matrix that can be used to rotate vectors between BRF,
    defined by the input (Ix=e1, Iy=e2, Iz=e3), and FRF."""
    Ix = np.array([0.13793203, 0.80497851, 0.57704797])
    Iy = np.array([-0.59855185, 0.53193899, -0.59897963])
    Iz = np.array([-0.78912004, -0.26277465, 0.5551928])

    A = np.array([[Ix[0], Ix[1],Ix[2], 0, 0, 0, 0, 0, 0],
            [0, 0, 0, Ix[0], Ix[1], Ix[2], 0, 0, 0],
            [0, 0, 0, 0, 0, 0, Ix[0], Ix[1], Ix[2]],
            [Iy[0], Iy[1], Iy[2], 0, 0, 0, 0, 0, 0],
            [0, 0, 0, Iy[0], Iy[1], Iy[2], 0, 0, 0],
            [0, 0, 0, 0, 0, 0, Iy[0], Iy[1], Iy[2]],
            [Iz[0], Iz[1], Iz[2], 0, 0, 0, 0, 0, 0],
            [0, 0, 0, Iz[0], Iz[1], Iz[2], 0, 0, 0],
            [0, 0, 0, 0, 0, 0, Iz[0], Iz[1], Iz[2]]])

    B = np.array([1, 0, 0, 0, 1, 0, 0, 0, 1])

    X = np.matmul(B, np.linalg.inv(A))

    R = np.array([[X[0], X[1], X[2]],
            [X[3], X[4], X[5]],
            [X[6], X[7], X[8]]])
    return R

def get_EFdict(Ix, Iy, Iz, p_intr, par_beta, par_gamma):
    """Returns a dictionary with all the electric fields with strength 1
    V/nm, in the FRF. Keys correspond to field angles (beta, gamma) in degrees."""

    # First the fields aligned or anti-aligned with the principal axes,
    # a special case and also matching the fields for beta = 0 and
    # beta = 360. (OBS: all fields with gamma = 0 or 180 are independent
    # of beta, and will not be calculated more than once)
    EF_dict = {(0,0): Iz, (0,90): Ix, (0,180): np.multiply(Iz, -1), (90,90): Iy,
           (180,90): np.multiply(Ix, -1), (270,90): np.multiply(Iy, -1),
           (par_beta, par_gamma): normalize(p_intr),
           (165,75): rot(rot(Iz, Iy, 5*(pi/12)), Iz, 11*(pi/12)),
           (360,0): Iz, (360,180): np.multiply(Iz, -1),
           (360,30): rot(Iz, Iy, pi/6), (360,60): rot(Iz, Iy, 2*(pi/6)),
           (360,120): rot(Iz, Iy, 4*(pi/6)),
           (360,150): rot(Iz, Iy, 5*(pi/6)), (360,90): Ix}

    # For beta ranging between 0 and 330 deg, and gamma between 30 and 150
    # deg, in steps of 30 deg.
    for i in range(0,12):
        for j in range(1,6):
            EF_dict[(i*30, j*30)] = rot(rot(Iz, Iy, j*(pi/6)), Iz, i*pi/6)

    return EF_dict

def get_interpolation(xdict, R):
    """Returns an interpolating function (using SciPy's CloughTocher2DInterpolator)
    for either the dipoles or the fields, as functions of field angles (beta, gamma)."""
    points = [list(xdict.items())[i][0] for i in range(len(xdict))]
    values = [np.matmul(R, list(xdict.items())[i][1]) for i in range(len(xdict))]

    f = CloughTocher2DInterpolator(points,values) # Piecewise cubic, C1 smooth,
                                                  # curvature-minimizing interpolant in 2D.
    return f

def check_fields(EF_dict, a, b, Ix, Iz, R):
    """A function to check the electric field vectors in the FRF, to rotate them
    according to (beta, gamma) and see if the angles are correct and extract
    the field vector for the SIESTA input."""
    check_EFs = list(EF_dict.items())
    for field in check_EFs[a:b]:
        print(f"Check field for angles: {field[0]}")
        print(f"Field:\t {field[1]}")
        print(f"|E|:\t {magnitude(field[1])}")
        proj = field[1] - np.multiply(np.dot(field[1],Iz),Iz)
        projI = np.matmul(R,proj)
        if projI[1] < 0 and projI[1] < -1e-6:
            print(f"Theta:\t {(360 - angle(proj,Ix)):.4f}") # angle between Ix and
                                                            # field projected on the Ix-Iy plane
        else:
            print(f"Theta:\t {(angle(proj,Ix)):.4f}") # angle between Ix and field
                                                      # projected on the Ix-Iy plane
        print(f"Phi:\t {(angle(field[1],Iz)):.4f}\n")
    pass

def plot_dipoles(protein, p_intr, R, f, g):
    """A function to plot the magnitude of the total dipole, the induced dipole,
    the angle between the intrinsic dipole and the total dipole, the angle between
    the induced dipole and the field - all as functions of (beta, gamma), all as
    heatmaps and on a sphere."""
    phi, theta = np.meshgrid(np.linspace(0, 180, 200), np.linspace(0, 360, 200))
    z1 = np.zeros((len(theta),len(phi)))
    z2 = np.zeros((len(theta),len(phi)))
    z3 = np.zeros((len(theta),len(phi)))
    z4 = np.zeros((len(theta),len(phi)))

    for i in range(len(theta)):
        for j in range(len(phi)):
            total_dipole = f(theta[i][j],phi[i][j]) # In BRF
            field = g(theta[i][j],phi[i][j]) # In BRF
            # magnitude of total dipole
            z1[i][j] = magnitude(total_dipole)
            # magnitude of induced dipole
            z2[i][j] = magnitude(total_dipole-np.matmul(R, p_intr))
            # angle between total dipole and intrinsic dipole
            z3[i][j] = angle(total_dipole, np.matmul(R, p_intr))
            # angle between the induced dipole and the field
            z4[i][j] = angle(total_dipole-np.matmul(R, p_intr), field)

    z1_min, z1_max = z1.min(), z1.max()
    z2_min, z2_max = z2.min(), z2.max()
    z3_min, z3_max = z3.min(), z3.max()
    z4_min, z4_max = z4.min(), z4.max()
    x = np.sin(phi*pi/180) * np.cos(theta*pi/180)
    y = np.sin(phi*pi/180) * np.sin(theta*pi/180)
    z = np.cos(phi*pi/180)

    fig, ax = plt.subplots()
    c = ax.pcolormesh(theta, phi, z1, cmap='RdYlBu', vmin=z1_min, vmax=z1_max)
    ax.axis([theta.min(), theta.max(), phi.min(), phi.max()])
    fig.colorbar(c, ax=ax, label="Total dipole [Debye]")
    plt.xlabel("Field angle $\\beta$ [deg]")
    plt.ylabel("Field angle $\gamma$ [deg]")
    plt.savefig(f"{protein}_total_dipole.png")
    plt.show()

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    norm = colors.Normalize(vmin=z1_min, vmax=z1_max)
    ax.plot_surface(x, y, z, facecolors=plt.cm.RdYlBu(norm(z1)), shade=False)
    ax.quiver(1,0,0,1.2,0,0, color='black', arrow_length_ratio=0.1)
    ax.quiver(0,1,0,0,1.3,0, color='black', arrow_length_ratio=0.1)
    ax.quiver(0,0,1,0,0,1.3, color='black', arrow_length_ratio=0.1)
    m = cm.ScalarMappable(cmap=plt.cm.RdYlBu, norm=norm)
    m.set_array([])
    plt.colorbar(m, label="Total dipole [Debye]")
    ax.view_init(45,55)

    ax.text(2.3, 0, -0.3, "e$_1$")
    ax.text(0, 2.3, -0.3, "e$_2$")
    ax.text(0, 0.15, 2.4, "e$_3$")
    ax.set_axis_off()
    plt.savefig(f"{protein}_total_dipole_sphere.png")
    plt.show()

    fig, ax = plt.subplots()
    c = ax.pcolormesh(theta, phi, z2, cmap='RdYlBu', vmin=z2_min, vmax=z2_max)
    ax.axis([theta.min(), theta.max(), phi.min(), phi.max()])
    fig.colorbar(c, ax=ax, label="Induced dipole [Debye]")
    plt.xlabel("Field angle $\\beta$ [deg]")
    plt.ylabel("Field angle $\gamma$ [deg]")
    plt.savefig(f"{protein}_induced_dipole.png")
    plt.show()

    fig = plt.figure()
```

```
ax = fig.add_subplot(111, projection='3d')
norm = colors.Normalize(vmin=z2_min, vmax=z2_max)
ax.plot_surface(x, y, z, facecolors=plt.cm.RdYlBu(norm(z2)), shade=False)
ax.quiver(1,0,0,1.2,0,0, color='black', arrow_length_ratio=0.1)
ax.quiver(0,1,0,0,1.3,0, color='black', arrow_length_ratio=0.1)
ax.quiver(0,0,1,0,0,1.3, color='black', arrow_length_ratio=0.1)
m = cm.ScalarMappable(cmap=plt.cm.RdYlBu, norm=norm)
m.set_array([])
plt.colorbar(m, label="Induced dipole [Debye]")
ax.view_init(45,55)
ax.text(2.3, 0, -0.3, "e$_1$")
ax.text(0, 2.3, -0.3, "e$_2$")
ax.text(0, 0.15, 2.4, "e$_3$")
ax.set_axis_off()
plt.savefig(f"{protein}_induced_dipole_sphere.png")
plt.show()

fig, ax = plt.subplots()
c = ax.pcolormesh(theta, phi, z3, cmap='viridis', vmin=z3_min, vmax=z3_max)
ax.axis([theta.min(), theta.max(), phi.min(), phi.max()])
fig.colorbar(c, ax=ax, label="Angular deviation [Deg]")
plt.xlabel("Field angle $\\beta$ [deg]")
plt.ylabel("Field angle $\gamma$ [deg]")
plt.savefig(f"{protein}_angle_intr_vs_total.png")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
norm = colors.Normalize(vmin=z3_min, vmax=z3_max)
ax.plot_surface(x, y, z, facecolors=plt.cm.viridis(norm(z3)), shade=False)
ax.quiver(1,0,0,1.2,0,0, color='black', arrow_length_ratio=0.1)
ax.quiver(0,1,0,0,1.3,0, color='black', arrow_length_ratio=0.1)
ax.quiver(0,0,1,0,0,1.3, color='black', arrow_length_ratio=0.1)
m = cm.ScalarMappable(cmap=plt.cm.viridis, norm=norm)
m.set_array([])
plt.colorbar(m, label="Angular deviation [Deg]")
ax.view_init(45,55)
ax.text(2.3, 0, -0.3, "e$_1$")
ax.text(0, 2.3, -0.3, "e$_2$")
ax.text(0, 0.15, 2.4, "e$_3$")
ax.set_axis_off()
plt.savefig(f"{protein}_angle_intr_vs_total_sphere.png")
plt.show()

fig, ax = plt.subplots()
c = ax.pcolormesh(theta, phi, z4, cmap='viridis', vmin=z4_min, vmax=z4_max)
ax.axis([theta.min(), theta.max(), phi.min(), phi.max()])
fig.colorbar(c, ax=ax, label="Angular deviation [Deg]")
plt.xlabel("Field angle $\\beta$ [deg]")
```

```
    plt.ylabel("Field angle $\gamma$ [deg]")
    plt.savefig(f"{protein}_angle_induc_field.png")
    plt.show()

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    norm = colors.Normalize(vmin=z4_min, vmax=z4_max)
    ax.plot_surface(x, y, z, facecolors=plt.cm.viridis(norm(z4)), shade=False)
    ax.quiver(1,0,0,1.2,0,0, color='black', arrow_length_ratio=0.1)
    ax.quiver(0,1,0,0,1.3,0, color='black', arrow_length_ratio=0.1)
    ax.quiver(0,0,1,0,0,1.3, color='black', arrow_length_ratio=0.1)
    m = cm.ScalarMappable(cmap=plt.cm.viridis, norm=norm)
    m.set_array([])
    plt.colorbar(m, label="Angular deviation [Deg]")
    ax.view_init(45,55)

    ax.text(2.3, 0, -0.3, "e$_1$")
    ax.text(0, 2.3, -0.3, "e$_2$")
    ax.text(0, 0.15, 2.4, "e$_3$")
    ax.set_axis_off()
    plt.savefig(f"{protein}_angle_induc_field_sphere.png")
    plt.show()
    pass

def plot_linearitytest(protein, p_intr, minst, halv, p_par, pol_minst, pol_halv,
pol, pol_beta, pol_gamma, par_beta, par_gamma):
    """A function to plot the magnitude of the total dipole and the induced dipole
    as a function of EF strength, along with a linear fit, for two cases when the
    field was parallel with the intrinsic dipole, and a chosen case
    where the polarizability was large, these dipoles are called
    "p_par" and "p_pol" respectively, for strengths 0.1, 0.5, 1 V/nm."""

    # Define the x-values (EF strengths) and different y-values (dipole magnitudes)
    x = np.array([0, 0.1, 0.5, 1])
    y = np.array([magnitude(p_intr), magnitude(minst), magnitude(halv),
    magnitude(p_par)])
    y2 = np.array([magnitude(p_intr), magnitude(pol_minst),
    magnitude(pol_halv), magnitude(pol)])
    y3 = np.array([0, magnitude(minst-p_intr), magnitude(halv-p_intr),
    magnitude(p_par-p_intr)])
    y4 = np.array([0, magnitude(pol_minst-p_intr),
    magnitude(pol_halv-p_intr), magnitude(pol-p_intr)])

    # Find the linear fit to the total dipole values and plot along with the
    # simulation data
    param = np.polyfit(x,y,1)
    trendpoly = np.poly1d(param)
    plt.plot(x,trendpoly(x), '--', label = f"$\mu$ = {param[0]:.2f}*E + {param[1]:.2f}")
    param2 = np.polyfit(x,y2,1)
    trendpoly2 = np.poly1d(param2)
    plt.plot(x,trendpoly2(x), '--', label = f"$\mu$ = {param2[0]:.2f}*E +
                                               {param2[1]:.2f}")
    plt.plot(x, y, 'o', label = f"$\\beta$ = {par_beta}, $\gamma$ =
                                                {par_gamma} (Parallel)")
    plt.plot(x, y2, 'o', label = f"$\\beta$ = {pol_beta}, $\gamma$ =
                                                {pol_gamma}")
    plt.xlabel("EF strength [V/nm]")
    plt.ylabel("Total dipole [Debye]")
    plt.grid()
    plt.legend()
    plt.savefig(f"{protein}_total_dipole_vs_field.png")
    plt.show()

    # Find the linear fit to the induced dipole values and plot along with
    the simulation data
    param3 = np.polyfit(x,y3,1)
    trendpoly3 = np.poly1d(param3)
    plt.plot(x,trendpoly3(x), '--', label = f"$\mu_i$ = {param3[0]:.2f}*E +
                                        {param3[1]:.2f}")
    param4 = np.polyfit(x,y4,1)
    trendpoly4 = np.poly1d(param4)
    plt.plot(x,trendpoly4(x), '--', label = f"$\mu_i$ = {param4[0]:.2f}*E +
                                        {param4[1]:.2f}")
    plt.plot(x, y3, 'o', label = f"$\\beta$ = {par_beta}, $\gamma$ =
                                         {par_gamma} (Parallel)")
    plt.plot(x, y4, 'o', label = f"$\\beta$ = {pol_beta}, $\gamma$ =
                                        {pol_gamma}")
    plt.xlabel("EF strength [V/nm]")
    plt.ylabel("Induced dipole [Debye]")
    plt.grid()
    plt.legend()
    plt.savefig(f"{protein}_induced_dipole_vs_field.png")
    plt.show()
    pass
```

C.3 Mathematica code for 3D rotor model

```
(*3D Rigid Rotor (By Thomas Mandl with contributions from Ebba Koerfer, Spring 2022)
Equations and setup
We have two reference frames

1) FixedReferenceFrame (FRF)
    The lab frame;
    The orthonormal base of FRF is {ex, ey, ez}.
    Quantities given in this system are suffixed with 'xyz'.
2) BodyRefereceFrame (BRF)
     The reference frame fixed with the molecule;
     Aligned with the principal axes of the inertia tensor.
     The orthonormal base of BRF is {e1, e2, e3}.
     Quantities given in this system are suffixed with '123'.

At any time t the BRF can be described by a rotation of FRF given by three Euler
angles (EA) {phi, theta, psi}. We build a system of equations by

1) expressing angular velocity omega in terms of the EA
2) plugging it into the Euler equations for the BRF

The we solve with NDSolve for given dipole, inertia, field and initial
conditions, relying on the internal solver for Differential-algebraic
systems. To include polarizability the results from SIESTA simulations,
for the total dipole of ubiquitin in a 1 V/nm electric field, is
interpolated such that we can find the dipole at any angle of the field
w.r.t the BRF. Without polarizability this is simply the intrinsic dipole
at each point in time.

First the simulation results are interpolated, the total dipole components separately.
*)

xdata = {{-2.520532717618103, 11.531338040975143, 21.93550166734237,
    25.65522795389619, 21.857663539969067,
    11.600521660394719, -2.369272605698308}, {-2.520532717618103,
    9.440883474690374, 18.353793440252648, 21.79158077558249,
    18.549927305029964,
    9.69019508930884, -2.369272605698308}, {-2.520532717618103,
    4.159319823687277, 9.216711753667056, 11.27736296304185,
    9.650712578480451,
    4.551974958851076, -2.369272605698308}, {-2.520532717618103, \
    -2.8950471960097666, -2.998937734901979, -2.8126403122938637, \
    -2.468257125720079, -2.437023346825651, -2.369272605698308}, \
    {-2.520532717618103, -9.825589047764012, -15.000190097688895, \
    -16.670696919864074, -14.522694808899061, -9.403335448205203, \
    -2.369272605698308}, {-2.520532717618103, -14.769796438954767, \
    -23.567759081577172, -26.59390005647226, -23.31354209137541, \
    -14.480113486894027, -2.369272605698308}, {-2.520532717618103, \
    -16.411999788029878, -26.438660086179286, -30.154606049367747, \
    -26.476340556364846, -16.306178425021656, -2.369272605698308}, \
    {-2.520532717618103, -14.327025578602726, -23.046646492157958, \
    -26.329925057939565, -23.163825407571288, -14.395563582289235, \
    -2.369272605698308}, {-2.520532717618103, -9.081876985556022, \
    -14.149852413914225, -16.056455811765918, -14.266438055797135, \
    -9.257648449495655, -2.369272605698308}, {-2.520532717618103, \
    -2.072397579767241, -2.0512832120778057, -2.0881077269687154, \
    -2.167865765320201, -2.270973903718641, -2.369272605698308}, \
    {-2.520532717618103, 4.8766796260722245, 10.010447325414056,
    11.837984634943162, 9.89257103187866,
    4.693699427035359, -2.369272605698308}, {-2.520532717618103,
    9.862783182870842, 18.803766168050274, 21.99472198063611,
    18.687878182213787,
    9.772566281901225, -2.369272605698308}, {-2.520532717618103,
    11.531338040975143, 21.93550166734237, 25.65522795389619,
    21.857663539969067, 11.600521660394719, -2.369272605698308}};
xdata // MatrixForm;

ydata = {{-73.2964397806068, -75.81144429621762, -81.22290121809125, \
    -82.46347072837872, -82.80799505268061, -83.01335789316524, \
    -83.02864241890244}, {-73.2964397806068, -64.29172589597903, \
    -61.895518083592876, -66.1191918307242, -69.55361802490597, \
    -75.36163478258756, -83.02864241890244}, {-73.2964397806068, \
    -55.67343988581693, -47.07337448219829, -49.59728747300889, \
    -59.80582330717858, -69.73517479325433, -83.02864241890244}, \
    {-73.2964397806068, -52.33010087555643, -41.295394799091795, \
    -43.06269010628464, -55.879780680032134, -67.64336272922017, \
    -83.02864241890244}, {-73.2964397806068, -55.20342448182705, \
    -46.266998598783665, -48.74926539959103, -59.646817766301325, \
    -69.6480849268096, -83.02864241890244}, {-73.2964397806068, \
    -63.50498041446883, -60.587264661206575, -64.87731134739352, \
    -69.30155726745943, -75.21478030643647, -83.02864241890244}, \
    {-73.2964397806068, -74.92646343618351, -80.00096244954489, \
    -82.13602538714409, -82.518522193881, -82.84551729683568, \
    -83.02864241890244}, {-73.2964397806068, -86.30263346720437, \
    -95.05064466598415, -97.44224655447324, -95.77782750907502, \
    -90.50281391569641, -83.02864241890244}, {-73.2964397806068, \
    -94.2855773615439, -104.78829620288526, -108.68873849075116, \
    -105.52042217554443, -96.12816876188788, -83.02864241890244}, \
    {-73.2964397806068, -96.85697013859914, -108.40526126264896, \
    -112.86650724388677, -109.1379695488254, -98.21761299517436, \
    -83.02864241890244}, {-73.2964397806068, -94.56151278636965, \
    -104.92914406598436, -108.85407577720346, -105.66360510058198, \
    -96.21155791943147, -83.02864241890244}, {-73.2964397806068, \
    -86.99443162613588, -95.29449368411369, -97.72432046125061, \
    -96.02647837527131, -90.6440598365731, -83.02864241890244}, \
    {-73.2964397806068, -75.81144429621762, -81.22290121809125, \
    -82.46347072837872, -82.80799505268061, -83.01335789316524, \
    -83.02864241890244}};
ydata // MatrixForm;

zdata = {{24.018504757113128, 17.668208121138225,
    1.5059002436444047, -14.700459725131836, -30.107138659540656, \
    -41.41770187037278, -45.60275019413376}, {24.018504757113128,
    21.6131029917687,
    7.697720923495616, -13.34615381955081, -29.807793409306868, \
    -41.24601854249383, -45.60275019413376}, {24.018504757113128,
    24.657754055035696,
    12.848163308804995, -8.015111695185865, -29.612730962730097, \
    -41.13691875172853, -45.60275019413376}, {24.018504757113128,
    25.91202848309299,
    14.997645204724659, -5.683641437925166, -29.288765258432733, \
    -41.12004515255914, -45.60275019413376}, {24.018504757113128,
    24.987239465657225,
    13.404857814394417, -7.4662101809065735, -29.70640519461322, \
    -41.19802617766862, -45.60275019413376}, {24.018504757113128,
    22.15342738701994,
    8.567344832266501, -12.627148450567754, -29.987332316177245, \
    -41.34985033217071, -45.60275019413376}, {24.018504757113128,
    18.261900757915686,
    2.213469347184404, -14.935969791259543, -30.312792818564354, \
    -41.537541803497284, -45.60275019413376}, {24.018504757113128,
    14.467398647927482,
    0.1742288450331344, -15.282064145340168, -30.61103684110487, \
    -41.709303958943195, -45.60275019413376}, {24.018504757113128,
    12.064993342764659, -0.018390506299475362, -15.504641153096138, \
    -30.80093854409301, -41.81917112905292 6, -45.60275019413376}, \
    {24.018504757113128,
    11.57634731556034, -0.053647071306102134, -15.544824110375743, \
    -30.835218108369183, -41.838160743923424, -45.60275019413376}, \
    {24.018504757113128, 11.932888863859013,
    0.07970348898231805, -15.389756559354609, -30.70057666388883, \
    -41.76060656850297, -45.60275019413376}, {24.018504757113128,
    14.025357530376958,
    0.3457903928584116, -15.080148700634751, -30.434985208880178, \
    -41.60711010169797, -45.60275019413376}, {24.018504757113128,
    17.668208121138225,
    1.5059002436444047, -14.700459725131836, -30.107138659540656, \
    -41.41770187037278, -45.60275019413376}};

zdata // MatrixForm;

dipole1 =
  ListInterpolation[xdata, {{0, 2 Pi}, {0, Pi}},
   InterpolationOrder -> 3];
dipole2 =
  ListInterpolation[ydata, {{0, 2 Pi}, {0, Pi}},
   InterpolationOrder -> 3];
dipole3 =
  ListInterpolation[zdata, {{0, 2 Pi}, {0, Pi}},
   InterpolationOrder -> 3];


dipole1b[itheta_, iphi_] := 2.25;
dipole2b[itheta_, iphi_] := -82.30;
dipole3b[itheta_, iphi_] := -14.81;

(*To check that the interpolation works, and gives the same result as the Python
code, it is plotted as a function of beta and gamma. *)
```

```
m = Table[
   Norm[{dipole1[beta, gamma], dipole2[beta, gamma],
     dipole3[beta, gamma]}], {beta, 0, 2 Pi, Pi/12}, {gamma, 0, Pi,
    Pi/12}];
ListContourPlot[m, InterpolationOrder -> 3,
 PlotLegends -> BarLegend[Automatic, All],
 DataRange -> {{0, 2 Pi}, {0, Pi}}]

(*Now we set up the rotor model. *)

(*the base of BRF as a function of EA*)
e1[phi_, theta_, psi_] :=
 EulerMatrix[{phi, theta, psi}, {3, 1, 3}] . {1, 0, 0}
e2[phi_, theta_, psi_] :=
 EulerMatrix[{phi, theta, psi}, {3, 1, 3}] . {0, 1, 0}
e3[phi_, theta_, psi_] :=
 EulerMatrix[{phi, theta, psi}, {3, 1, 3}] . {0, 0, 1}

(*Calculates the azimuthal angle of the field, beta, w.r.t the BRF*)
getBeta[phi_, theta_, psi_] :=
  If[field2 < 0, 2*Pi - VectorAngle[{field1, field2, 0} , {1, 0, 0}] ,
     VectorAngle[{field1, field2, 0} , {1, 0, 0}]] /. {field1 ->
     toBRF[phi, theta, psi, {fx, fy, fz}][[1]],
    field2 -> toBRF[phi, theta, psi, {fx, fy, fz}][[2]],
    field3 -> toBRF[phi, theta, psi, {fx, fy, fz}][[3]]};

(*Calculates the polar angle of the field, gamma, w.r.t the BRF*)
getGamma[phi_, theta_, psi_] :=
  VectorAngle[{fx, fy, fz}, e3[phi, theta, psi]];

(*Returns the total dipole with polarizability: *)
getDipole[phi_, theta_,
   psi_] := {dipole1[beta, gamma], dipole2[beta, gamma],
    dipole3[beta, gamma]} /. {beta -> getBeta[phi, theta, psi],
    gamma -> getGamma[phi, theta, psi]};

(*Returns the intrinsic dipole (without polarizability):
getDipole[phi_, theta_, psi_] := {dipole1b[beta,gamma], \
dipole2b[beta,gamma], dipole3b[beta,gamma]} /. {beta-> getBeta[phi, \
theta, psi],gamma-> getGamma[phi, theta, psi]};
*)

(*a function to convert some vector v given in BRF to FRF for given \
EA*)
toFRF[phi_, theta_, psi_, v_] :=
 e1[phi, theta, psi] v[[1]] + e2[phi, theta, psi] v[[2]] +
  e3[phi, theta, psi] v[[3]]
```

```
(*a function to convert some vector v given in FRF to BRF for given \
EA*)
toBRF[phi_, theta_, psi_, v_] := {e1[phi, theta, psi] . v,
  e2[phi, theta, psi] . v, e3[phi, theta, psi] . v}

(*angular velocity in terms of EA*)
w1[t] = phi'[t] Sin[theta[t]] Sin[psi[t]] + theta'[t] Cos[psi[t]];
w2[t] = phi'[t] Sin[theta[t]] Cos[psi[t]] - theta'[t] Sin[psi[t]];
w3[t] = phi'[t] Cos[theta[t]] + psi'[t];

(* torque in BRF
calculated by f x d, (field cross dipole)
remove semicolon to see the generated rules.
modify this to simulate time dependent or more complex fields/dipoles \
*)
torque = Inner[Rule, {M1[t], M2[t], M3[t]} ,
    Cross[{d1, d2, d3},
     toBRF[phi[t], theta[t], psi[t], {fx, fy, fz}]],
    List ] /. {d1 -> getDipole[phi[t], theta[t], psi[t]] [[1]],
    d2 -> getDipole[phi[t], theta[t], psi[t]] [[2]],
    d3 -> getDipole[phi[t], theta[t], psi[t]] [[3]]};

(*inital conditions given as symbols.
later we apply rules to substitute with specific values/ranges *)
init = {phi[0] == phi0, theta[0] == theta0, psi[0] == psi0,
   theta'[0] == thetaD0, phi'[0] == phiD0, psi'[0] == psiD0};

(*The Euler equations in BRF with the dipole/field interaction \
included as a substituion rule.
remove the semicolon to see the actual equations*)
eq = {
    I1 D[w1[t], t] + (I3 - I2) w1[t] w2[t] == M1[t] ,
    I2 D[w2[t], t] + (I1 - I3) w3[t] w1[t] == M2[t] ,
    I3 D[w3[t], t] + (I2 - I1) w2[t] w1[t] == M3[t]
    } /. torque;

(*the system to solve. Euler equations plus initial conditions
remove semicolon to see the actual system*)
sys = {eq, init} // Flatten;

(*function to solve the 3D rotor for given inertia, dipole, initial \
conditions and simulation time
*)
solve[inertia_, field_, init_, T_] :=
 NDSolve[sys /. ({inertia, field} // Flatten) /. init, { phi[t],
   theta[t], psi[t]}, {t, 0, T}, SolveDelayed -> True]

(*Solve the 3D rotor
We solve for a specific setup and get substitution rules as the result *)
```

```
sol = solve[
  inertia = {I1 -> 8701.1, I2 -> 7250.6, I3 -> 6310.1},
  field = {fx -> 1, fy -> 0, fz -> 0},
  init = {phi0 -> 4 Pi/5, theta0 -> Pi/4, psi0 -> Pi/2,
    thetaD0 -> 0, phiD0 -> 0, psiD0 -> 0}

  , T = 2000]

(*We plot the results...*)

(*plot BRF in FRF*)
ParametricPlot3D[{Evaluate[e1[phi[t], theta[t], psi[t]] /. sol],
  Evaluate[e2[phi[t], theta[t], psi[t]] /. sol],
  Evaluate[e3[phi[t], theta[t], psi[t]] /. sol]}, {t, 0, T},
 AxesLabel -> {x, y, z},
 PlotLegends -> LineLegend [{"e1", "e2", "e3"}]]

(*plot dipole in FRF*)
ParametricPlot3D[
 Evaluate[toFRF[phi[t], theta[t], psi[t],
    getDipole[phi[t], theta[t], psi[t]]] /. field /. sol], {t, 0, T},
  AxesLabel -> {x, y, z}, PlotLegends -> LineLegend[{"\[Mu]"}],
 PlotRange -> {{-30, 120}, {-90, 90}, {-90, 90}}]

(*plot beta and gamma*)
 Plot[{Evaluate[getBeta[phi[t], theta[t], psi[t]] /. field /. sol],
  Evaluate[getGamma[phi[t], theta[t], psi[t]] /. field /. sol]}, {t,
  0, T}, AxesLabel -> {"time", "[rad]", z},
 PlotLegends -> LineLegend[{"beta", "gamma"}],
 PlotLabel -> "Field angles"]

(*plot angle between dipole and field*)
Plot[Evaluate[
  VectorAngle[{fx, fy, fz} /. field // Flatten,
   toFRF[phi[t], theta[t], psi[t],
      getDipole[phi[t], theta[t], psi[t]]] /. field /. sol //
    Flatten]], {t, 0, T}, AxesLabel -> {"time", "[rad]", z},
 PlotRange -> {0, 3.14},
 PlotLegends -> LineLegend[{"Angle \[Mu] v E"}]]

(*EA*)
Plot[{Evaluate[phi[t] /. sol], Evaluate[theta[t] /. sol],
  Evaluate[psi[t] /. sol]}, {t, 0, T},
 AxesLabel -> {"time", "[rad]", z},
 PlotLegends -> LineLegend[{"phi", "theta", "psi"}],
 PlotLabel -> "Euler angles"]

(*first principal axes in FRF*)
```

```
Plot[Evaluate[e1[phi[t], theta[t], psi[t]] /. sol], {t, 0, T},
 AxesLabel -> {x, y, z},
 PlotLegends -> LineLegend[{"I1x", "I1y", "I1z"}],
 PlotLabel -> "first principal axes"]

(*plot magnitude of dipole*)
Plot[Norm[
  Evaluate[getDipole[phi[t], theta[t], psi[t]]] /. field /. sol], {t,
  0, T}, AxesLabel -> {"time", "[Debye]", z},
 PlotLegends -> LineLegend[{"|\[Mu]|"}] , PlotRange -> {50, 115}]

(*3D visualization
select time or let it play. hit '+' button for more options and slow down
playback speed with arrows down (needs to be slowed down several times). *)

disp = {0.05, 0.05, 0.05};

m = Manipulate[
  {
   Graphics3D[
    {
     (*inertia ellipsoid in normalized form just to see the shape*)
     {Opacity[0.8],
      GeometricTransformation[
       Ellipsoid[{0, 0, 0}, Normalize[{I1, I2, I3}] /. inertia],
       EulerMatrix[{phi[t], theta[t], psi[t]}, {3, 1, 3}] /. sol /.
        t -> tt]},

     (*BRF*)
     Arrow[{{0, 0, 0},
       e1[phi[t], theta[t], psi[t]] /. sol /. t -> tt // Flatten}],
     Arrow[{{0, 0, 0},
       e2[phi[t], theta[t], psi[t]] /. sol /. t -> tt // Flatten}],
     Arrow[{{0, 0, 0},
       e3[phi[t], theta[t], psi[t]] /. sol /. t -> tt // Flatten}],

     Text["e1",
      disp + e1[phi[t], theta[t], psi[t]] /. sol /. t -> tt //
       Flatten],
     Text["e2",
      disp + e2[phi[t], theta[t], psi[t]] /. sol /. t -> tt //
       Flatten],
     Text["e3",
      disp + e3[phi[t], theta[t], psi[t]] /. sol /. t -> tt //
       Flatten],

     (*dipole*)
     {Red,
      Arrow[{{0, 0, 0},
        Evaluate[
             toFRF[phi[t], theta[t], psi[t],
              Normalize[getDipole[phi[t], theta[t], psi[t]]]]] /.
            field /. sol /. t -> tt // Flatten]]},
     {Red,
      Text["p", {disp +
             Evaluate[
              toFRF[phi[t], theta[t], psi[t],
               Normalize[getDipole[phi[t], theta[t], psi[t]]]]] /.
            field /. sol /. t -> tt // Flatten]]},

     (*field*)
     Arrow[{{1.8, -1, 0}, {1.8, -1, 0} + {fx, fy, fz} /. field /.
        t -> tt }],
     Text["F",
      disp + {1.8, -1, 0} + {fx, fy, fz} /. field /. t -> tt ],

     (*
     {Red, Arrow[{{0,0,0},Cross[toFRF[phi[t], theta[t], psi[t], {d1,
     d2, d3}/.dipole]/.sol/.t\[Rule]tt //Flatten, {fx, fy,
     fz}/.field /.t\[Rule]tt ]}]
     },
     {Red, Text["M",2 disp + Cross[toFRF[phi[t], theta[t], psi[
     t], {d1, d2, d3}/.dipole]/.sol/.t\[Rule]tt //Flatten, {fx, fy,
     fz}/.field /.t\[Rule]tt ]]}
     *)
     }
    , AxesLabel -> {"x", "y", "z"},
    PlotRange -> {{-3, 3}, {-2, 2}, {-2, 2}}, Axes -> True,
    ImageSize -> Medium,
    PlotLabel -> StringForm["Reference frame\nt=``", tt],
    ViewPoint -> {2, 1.4, 1.2}]

   }
  ,
  {tt, 0, T, 0.05}]

(*To check the unit of the time-step:*)

fq = Quantity[1, "V/nm"]
iq = Quantity[1, "Debyes"] // UnitConvert[#, "nm * ps * A"] &
muq = Quantity[1, "Daltons * Angstroms * Angstroms"] //
  UnitConvert[#, "g *nm *nm"] &
(*define desired unit of timestep*)
ttarget = Quantity[1, "fs"];
1/(fq*iq/muq) // Sqrt
1/(fq*iq/muq ) // Sqrt // UnitConvert[#, "ps"] &
tscale = k -> (% // UnitConvert [#, QuantityUnit[ttarget]] & //
    QuantityMagnitude )
```