UNIVERSITÀ DEGLI STUDI DI TRIESTE

Master's Degree in Process and Materials Engineering

Master's Degree Thesis

Development of Machine-Learning based approaches for coarse-grained simulations

Supervisors	Candidate
Prof. Paola POSOCCO	Cristian GABELLINI
Prof. Zbyšek POSEL	

Academic Year 2019-2020

Abstract

Nel campo della simulazione molecolare al calcolatore, i metodi di *Machine Learning* hanno visto negli ultimi anni un impiego sempre più elevato per la creazione di potenziali inter-atomici o per l'ottimizzazione di potenziali pre-esistenti. I vantaggi derivanti dall'utilizzo di queste tecniche non portano solo ad una riduzione del tempo di macchina necessario per effettuare queste simulazioni, ma anche a sfruttare le caratteristiche fisiche dei sistemi studiati in scale dimensionali differenti definendo così un nuovo modo di effettuare simulazioni multiscala.
I principali campi di utilizzo di queste tecniche, indicate con il termine *data driven*, risiedono nei sistemi *ab-initio* e atomistici; ben inferiore per ora è invece l'utilizzo di questi approcci nelle simulazioni mesoscala. In questa tesi ho sviluppato delle soluzioni che permettono di utilizzare tecniche di *Machine Learning* in simulazioni mesoscala e che sono applicabili a sistemi con una complessità più elevata di quelli fino ad ora analizzati nella letteratura scientifica senza gli svantaggi dei metodi tradizionali. In particolare, l'approccio noto come ottimizzazione bayesiana è stato qui utilizzato per derivare i parametri del potenziale di interazione in simulazioni coarse-grained basate sulla *Dissipative Particle Dynamics*.

Abstract

The last few years, the field of computer aided simulations has seen an increasingly high application of *Machine Learning* techniques for the creation of interatomic potentials or for the optimization of force fields. The advantages from the use of these techniques not only lead to a reduction of the machine time necessary to carry out these simulations, but also in the use of these methods to exploit the physical characteristics of the studied system in different dimensional scales thus defining a new way to multiscale simulations.
These techniques, indicated with the term *data driven*, have been applied in *ab-initio* and atomistic systems; however so far, their use in mesoscale simulations is still quite limited. In this thesis I developed a framework that allows to use *Machine Learning* techniques in mesoscale simulations in systems having a higher complexity than those analyzed so far, overcoming the drawbacks of traditional multiscale methods. I propose here an approach capable of extending these methods to coarse-grained simulations based on *bottom-up* approaches relying on bayesian optimization and focusing on the derivation of the parameters for a *Dissipative Particle Dynamics* potential.

This work is dedicated to Prof. Alessandro De Vita

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AA	All-Atoms
AC	Acquisition Function
ANN	Artificial Neural Network
BO	Bayesian Optimization
BO-GP	Bayesian Optimization with Gaussian Process
COM	Center of Mass
CG	Coarse-Grained
D	Dataset
DPD	Dissipative Particle Dynamics
EI	Expected Improvement
GP	Gaussian Process
IBI	Iterative Boltmann Inversion
ML	Machine Learning
ML-FF	Machine Learning Force Field
NP	Nanoparticle
NN	Neural Network
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RDF	Radial Distribution Function

Chapter 1

Introduction

In recent years, Machine Learning (ML) has seen an exceptional grow of interest in various and different fields. This steep increase of interest is not only restricted to the academic field but affects many areas of our normal life. *Data driven* approaches are becoming a common reality everywhere and lately, even material science has started to use them for their superhuman abilities of utilizing high-dimensional data (Figure 1.1).

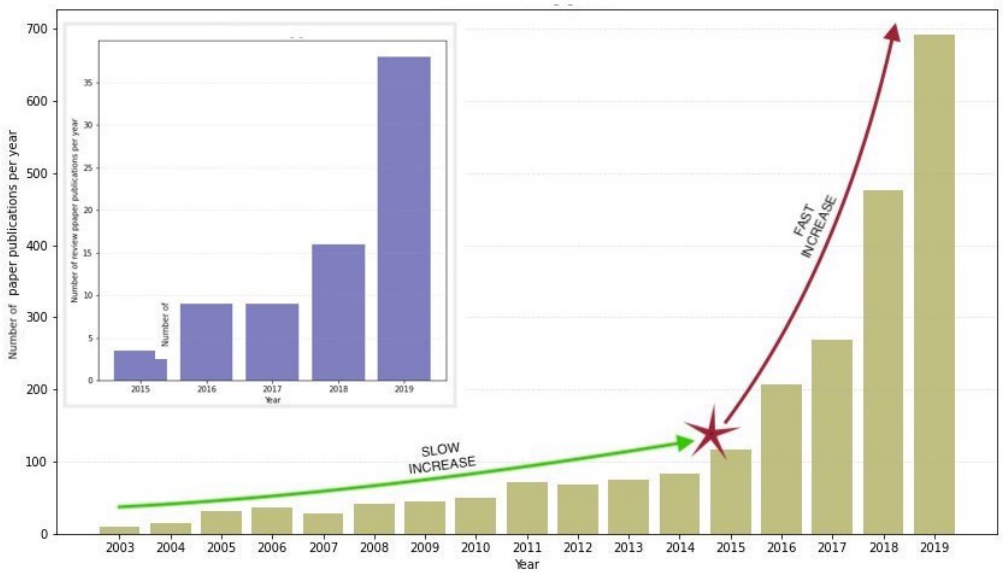


Figure 1.1: Growth of machine learning articles and reviews for material science in the last two decades. In the inlet are shown the number of reviewed publicatios in the last five years.

A pletora of work has been produced by exploiting the capabilities and flexibility of Machine Learning techniques, from speeding up complex and CPU intensive *ab-initio* simulations [1] to the prediction and discovery of new materials [2]. Intelligent model reduction via mathematical, numerical and statistical approaches to reveal the high-dimensional multiscale data in low-dimensional industrial scale has been played a crucial role in efficient design of advanced materials, processes, structures and complex mechanical systems. Exciting developments of intelligent model reduction techniques often go beyond classical theories, incorporate more profound physical mechanisms, machine learning techniques, parallel and high performance computing, and are becoming the exclusive numerical tools in addressing the computational challenges which were difficult or impossible to solve by utilizing the conventional methods.

Among them, one particularly intriguing possibility is the prediction of the hierarchical behavior of auto-organizing nanomaterials. They are usually constituted by thousands of building blocks, self-assembling in the three-dimensional space with structures and properties that are scale and time dependent. These supramolecular structures have been studied experimentally for the past few decades and applications are envisaged in the field of catalysis[3], nanocomposites fabrication[4], sensing[5] and delivery[6]. Being able to known in advance their organization and properties by using computer aided modeling is therefore of paramount importance in designing novel and optimized nanosystems. In this context, polymer-grafted nanoparticles are one of the simplest building block investigated in literature. They are formed by a nanosized inorganic core (typically gold) to which the polymeric chains are physical or covalently linked. Chemical composition and topology of the chains can be designed to grant specific properties both to the nanoparticle and to the overall material. I decided to focus my investigation on this simple, yet versatile system. Predictions of its self-assembling mechanism require the use of coarse-graining (CG) simulations since the common scale lengths span the order of nanometers or micrometers, which are out of the range of the most common atomistic calculations. However, these coarse-grained calculations need the derivation of specific parameters to be carried out. This task is usually time-consuming, and system dependent [7][8][9].

Accordingly, in this thesis I chose to rethink the common approach to the derivation of CG parameters, by implementing a ML based workflow able to exploit structural insights coming from atomistic simulations. Among the several possible approaches I focused my work on the parameterization of a dissipative particle dynamics force field (DPD), as this coarse-graining method is widely used for the simulation of polymer based nanoparticles and for the phenomenon of self-assembly in soft matter systems [8]. I chose to rethink these approaches by utilizing ML to overcome the aforementioned bottlenecks and be able to exploit structural insights from atomistic simulations. In order to achieve this result I have applied a technique

denominated bayesian optimization, and implemented a bottom-up approach based on a structural strategy which has resulted in a workflow to parameterize a complex coarse-graining problem.

Chapter 2

Theoretical Framework

2.1 Molecular Simulations

The aim of this section is to present *state of the art* techniques used in the field of computer simulations with special focus on coarse-grained (CG) techniques.

2.1.1 Atomistic Simulations

Atomistic simulations aim to simplify the real system by considering it composed only by atoms, neglecting the electronic interactions. The so called all-atoms model (or AA) can be defined by the energy E of its Hamiltonian (from the Schrödinger equation):

$$H\Phi(R,r) = E\Phi(R,r) \tag{2.1}$$

where E is an empirical fitting called force field $E(R)(FF)$ that describes the electronic effects lost with the simplification of the system. While this force field (FF) can be found in various ways, be it empirical, data-driven or physical, all FFs can be usually written as a sum of different contributions, as an example:

$$E(R) = E_{val} + E_{nb} + E_{cross} + ... \tag{2.2}$$

with contributions from the bonded interactions, the non bonded interactions, the electrostatic interactions, and a corrective contribute for the chemical species environment. Analitically, these terms depend in general on the FF considered; here we report an example of different types of interactions that can be found in

the CHARMM force field [10]:

$$\begin{cases} E_{val} = E_{bond} + E_{angle} + E_{torsion} \\ E_{bond} = \frac{1}{2}K_{ij}(r_{ij} - r_{ij}^0)^2 \\ E_{angle} = \frac{1}{2}C_{ijk}(cos\theta_{ijk} - cos\theta_{ijk}^0)^2 \\ E_{torsion} = \frac{1}{2}E_n[1 - cos(n\theta_{ijkl})] \end{cases} \quad (2.3)$$

$$\begin{cases} E_{nb} = E_{VdW} + E_{coul} + E_{HB} \\ E_{VdW} = \frac{n\epsilon}{n-m}[\frac{m}{n}(\frac{r_{ij}}{r^*})^{-n} - (\frac{r_{ij}}{r^*})^{-m}] \\ E_{coul} = \frac{q_i q_j}{K r_{ij}} \end{cases}$$

where K_{ij}, C_{ijk} and E_n are the stiffness of the type of interaction, r_{ij} is the bond length, θ_{ijk} is the valance angle and θ_{ijkl} is the dihedral angle. As these FF are empirical, more types of interactions can be added and the force field can be built to focus on computing specific modelling problems like proteins and ligands simulations [10], or in a more general way to simulate a vast range of elements [11] or with different type of preferred interactions [12]. As for classical molecular dynamics, the system must obey the Newton's equation of motion: [13]:

$$M\frac{d}{dt^2}\mathbf{q} = -\nabla_q(E(\mathbf{R})(FF)) \quad (2.4)$$

in which M is the atomic mass and q is the cartesian position vector. This highly non linear system can be solved by various numerical integration techniques such as the Verlet modified method [13] or the Leap-Frog method[14].

2.1.2 Coarse-Grained Simulations

The reason why coarse-graining (CG) is utilized in molecular simulation is simple to understand; even with modern computers, lots of systems can not be simulated with a AA or *ab-initio* techniques as the computational time required for these simulations is beyond capability of today computers. As *ab-initio* are limited to the dimensions of the system and the lengths of its simulation, coarse-graining techniques tries to overcome these limits by reducing the degrees of freedom of the system at the expense of the molecular detail, to achieve a system where multiple atoms or even molecules form the simplest object called *bead*.

While losing chemical and physical pieces of information, we obtain a simplified model that can be simulated with less computational resources and for an increased time length. While there are lots of different CG approaches and force fields, Martini[15] and Voth[16] just to name a few, this work will focus on the Dissipative Particle Dynamics (DPD) force field as largely employed in the scientific community to explore self-assembling[3][8].

Once the CG force field has been selected, the FF parameters have to be assigned. This can be done following two different approaches [17][18]:

- force matching
- structure matching.

Property and force matching will be briefly reviewed after the description of the DPD force field. Structure matching will be described more in details as it is the methodology applied in this thesis.

Dissipative Particle Dynamics

Dissipative particle dynamics is a CG method particularly suited to investigate polymers[4], self-assembling systems [3][8], and more in general soft matter at the nanoscale[19][20]. This technique assumes that the the number of particles (or beads) and total momentum of the system is conserved, while the energy is not [21]. In their formulation the total force on a particle is given by the sum of a conservative force, a dissipation force and a random force:

$$f_{ij} = \sum_{j \neq i} (\mathbf{F_{ij}^C} + \mathbf{F_{ij}^D} + \mathbf{F_{ij}^R}) \tag{2.5}$$

where the dissipative forces $\mathbf{F_{ij}^D}$ are linear to the momentum and $\mathbf{F_{ij}^R}$ are independent from it. The analytical form of these forces was found by solving the Fokker-Planck equation[21][22]:

$$\begin{aligned} \mathbf{F_{ij}^C} &= \begin{cases} a_{ij}(1 - r_{ij})\hat{\mathbf{r_{ij}}} \text{ for } r_{ij} \geqslant 1 \\ 0 \text{ for } r_{ij} < 1 \end{cases} \\ \mathbf{F_{ij}^D} &= -\gamma \omega_D(r_{ij})(\hat{\mathbf{r_{ij}}} \cdot \mathbf{v_{ij}})\hat{\mathbf{r_{ij}}} \\ \mathbf{F_{ij}^R} &= \sigma \omega_R(r_{ij})\hat{\mathbf{r_{ij}}}\ \zeta_{ij} \Delta \mathbf{t}^{-1/2} \end{aligned} \tag{2.6}$$

in which $r_{ij} = |r_i - r_j|$, $\hat{\mathbf{r_{ij}}}$ is the unit vector from j to i, a_ij is the interaction coefficient between j and i, ζ_{ij} is a Gaussian white-noise that ensures the conservation of the total momentum chosen for each pair of beads independently. It was also shown that the friction coefficient γ and the noise amplitude σ, under the Gibbs canonical ensemble, must satisfy of the fluctuation-dissipation theorem [21]:

$$\begin{aligned} \omega_R(r) &= \omega_D^{1/2}(r) \\ \omega &= (2k_B T\gamma)^{1/2} \end{aligned} \tag{2.7}$$

Force Matching

In the force-matching techniques to coarse-graining, the forces that act on preassigned site centers are fitted to the respective average forces computed from AA molecular dynamics simulations. They assume that the forces can be expressed as a sum of short-range interaction and a coloumb long-range interaction [23][24]:

$$f_i^{CG}(r_{ij}) = (f(r_{ij}) + \frac{q_i q_j}{r_{ij}^2})\frac{r_i - r_j}{r_{ij}} \tag{2.8}$$

where $r_{ij} = ||r_i - r_j||$.

Structure Matching

Among many coarse-graining techniques that try to find a potential matching, for instance the pair correlation function $g(r)$ of the AA system (Figure 2.1), typically relying on the potential of mean force and the integral equations, the most interesting one is the Iterative Boltzmann Inversion or IBI [18].

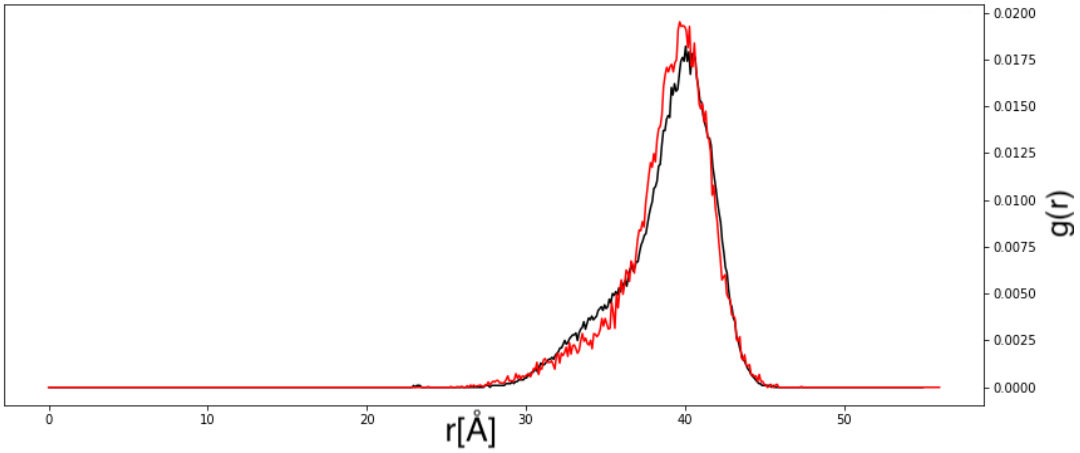


Figure 2.1: Example of matching between the $g(r)$ for the terminal group of a N16 chain for an AA model and CG model. Color legend: AA, black; CG, red.

Iterative Boltzmann Inversion IBI like most of the structure matching coarse-graining approaches aims to match the pair correlation function of the AA model. To achieve that, every iteration improves the CG potential from the difference between the target AA $g(r)$ and the CG one. Analytically it can be expressed as [17][18]:

$$\Phi_{i+1}(R) = \Phi_i(R) + \beta^{-1} ln \frac{g_i(R)}{g_{AA}(R)} \tag{2.9}$$

Because at the starting iteration there are no informations about the potential Φ, the starting potential is approximated to the potential of mean force for the fist iteration:

$$\Phi_{start}(R) = -\beta^{-1} ln(g_{AA}(R)) \tag{2.10}$$

2.2 Machine Learning

With the advance of the computational power and the recent spotlight on the power of the data, Machine Learning (ML) models started to be seen as a viable way to extract pieces of information hidden from the human eyes. This section aims to briefly introduce the common ML notation and explaining the theory behind the ML techniques utilized in this thesis work.

ML is a huge area of study, the techniques are commonly differentiated by the type of data they input and output, and also by the type of task or problem that they are intended to solve. This lead the classification in the following macro areas:

- supervised learning
  - regression
  - classification
- unsupervised learning
- reinforcement learning.

Since the techniques used in this work are part of the *supervised learning* area, the following notations and definitions will cover only the general case of *supervised learning* and will focus on the regression problems. The only difference between regression and classification is the output type, as regression utilizes continuous values as a target while the classification works with categorical classes. A review of the remaining techniques can be found here [25].

2.2.1 Notations

supervised learning is the problem of finding a function that maps an input to an output based on input-output pairs (X,Y) [26]; X and Y are commonly referred to as *feature* and *target*, and together they constitute the *dataset* D. The ML model itself ($\hat{F}$) is the object that predicts the corresponding target value after having received a feature vector X^*:

$$\hat{Y}^* = \hat{F}(X^*) \tag{2.11}$$

This model ($\hat{F}$) is generally defined by its type (gaussian process (GP), neural network (NN), supported vector machine, ridge regression, random forest), a set of model parameters and a set of *hyperparameters* [26]. The difference between these two types of coefficients reside in the methods utilized to learn them and in their properties. The model parameters are defined by the data of the problem while the *hyperparameters* are not determined directly by them. A distinction between the two types of parameters can be also done by their learning techniques

(fitting). While the model parameters are fitted in a training phase by the estimator (minimization of a loss function with the Newton method, backpropagation for NN, inversion of the covariance matrix for GP regression); the *hyperparameters* instead, need to be handled with different techniques such as grid sampling and applied to empirical measures like cross validation or bayesian inference [26] [27].

2.2.2 Workflow

A ML problem starts with the step of data collection and the building of the dataset D. The first generation of data is usually labeled *raw data* as they must be managed before they can be utilized for training a model, this step is called *pre-processing*. In the *pre-processing* the *raw data* is cleaned by removing the null values, the incorrect entries (NaN) or the outliers; in this step, an analysis of the data is done with the objective of finding correlations or trends. Inside this phase the $features\ extraction$ phase aims to reduce the dimensionality of the dataset by embedding the informations of the data and transforming them into machine readable features X and Y that will be used for the model.

As the dataset D is completed, the dataset is unevenly divided in three different categories:

- training set: data for training the model and fitting the parameters
- test set: data for evaluating the trained model
- validation set: data for tuning the model's $hyperparameters$ and for comparing different models.

The model is then trained, tuned and evaluated by using a performance metric (R^2, $RMSE$) on the test set. A validation can also be done with techniques such as cross-validation on the validation to better compare the yield of different models [26][27]. In Figure 2.2 a simplified workflow for a supervised model is showed.

Descriptors The $features\ extraction$ phase aims to embed and transform data into readable features for the ML model. Since molecular modelling employs data obtained from complex physical structures, the final features must satisfy three invariances:

- translational invariance
- rotational invariance
- permutational invariance.

The features utilized in molecular modelling that satisfy these three invariances are commonly called *descriptors*. Various types of descriptors have been developed to better embed the local atomic environments of the systems, for spatial informations the Behler-Parrinello symmetry functions is commonly utilized in neural networks [28], while methods that not only consider the spatial environment but also the chemical one such as the Smooth Overlap of Atomic Orbitals (SOAP) have given great results in GP methods [29].

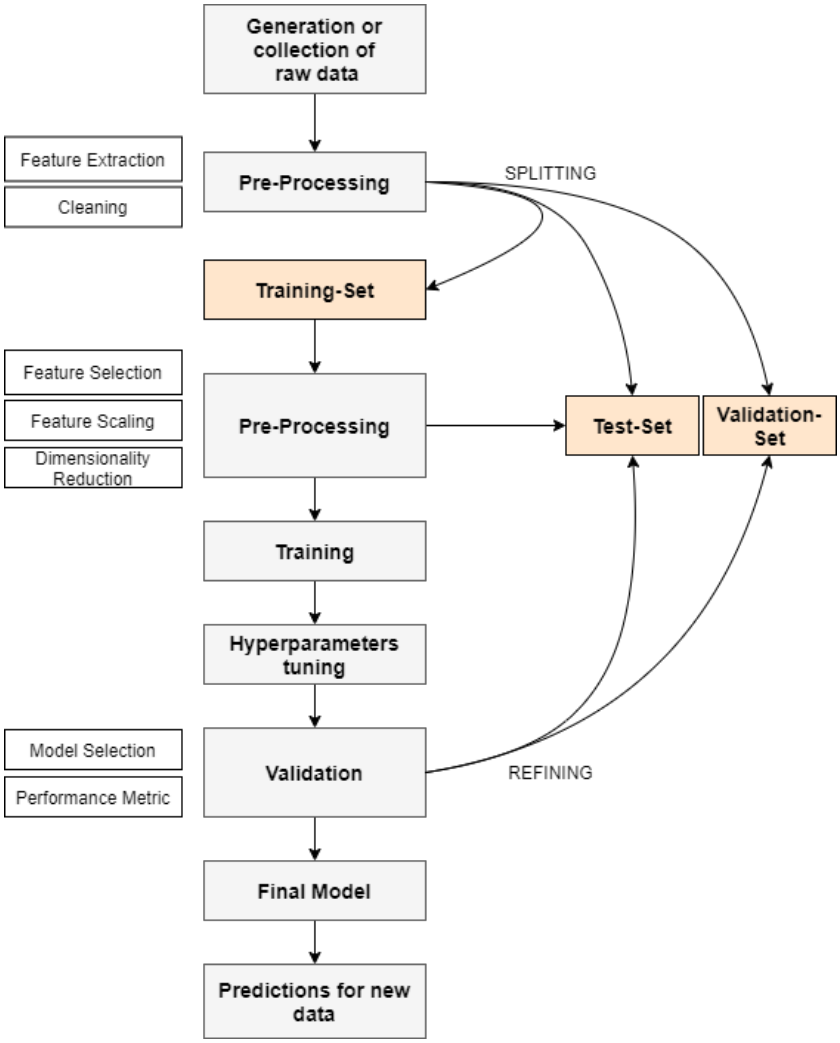


Figure 2.2: Example of workflow for a supervised machine learning model.

2.2.3 Gaussian Processes

Guassian processes are supervised models used for their ability to capture the uncertainty of their prediction and for their flexibility in introducing prior knowledge of the targetted studied system.

Formally, it can be defined as a collection of random variables in which any finite number have a joint gaussian distribution, every GP is specified by its mean function and his covariance function [27]:

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}^{'})) \tag{2.12}$$

The principal assumption of a GP is that the outputs can be sampled from a multivariate gaussian distribution, and so the joint distribution of the target $\mathbf{y}$ and the function values at the sample locations X_* can be written as [27]:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} = \boldsymbol{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{2.13}$$

where σ_n^2 is the variance of a gaussian noise with zero mean and unit variance $\varepsilon = N(0, \sigma_n^2)$ of the observation and $K(X,X)$ is the Graham matrix [26]. By conditioning the joint gaussian prior distribution over the observation X_* we get the analytical equations for the mean of the GP and the variance of the GP:

$$\begin{aligned} \bar{\mathbf{f}_*} &= K(X_*,X)\ [K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y} \\ var(\mathbf{f}_*) &= K(X_*,X_*) - K(X_*,X)\ [K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*) \end{aligned} \tag{2.14}$$

We can write a shorter version of equation 2.3 by applying the symmetric property of the covariance matrix, and thus writing $k_* = K(X_*,X) = K(X,X_*)$ and compacting $[K(X,X) + \sigma_n^2 I]^{-1}$ into C and $K(X_*,X_*)$ into k_{**}:

$$\begin{aligned} \bar{f_*} &= k_*^T\ C\ \mathbf{y} \\ var(f_*) &= k_{**} - k_*^T\ C\ k_* \end{aligned} \tag{2.15}$$

This form explain the power of the bayesian interpretation behind the prediction of a GP: on every sample point x_* we can obtain a prediction value and an uncertainty $var(f_*)$ associated with that prediction.

As a GP is specified by its mean and its covariance, the structure of the latter is one of the core task in the selection of the model. The covariance implies a distribution over functions and so by selecting a specific covariance it is possible to set prior information on this distribution [27], this is done by choosing a specific *kernel*. A kernel can be defined as a function k that maps a pair of input into $\mathbb{R}$ and from this definition, the Graham matrix K can be defined as the matrix whose entries K_{ij} are the kernel product $k(x_i, x_j)$ where x $\in D = \{x_1, x_2, ..., x_n\}$[27]. One

common kernel is the *squared exponential kernel* (or radial basis function RBF) for its properties of being isotropic due to the L_2-norm, their smoothness abilities and by being symmetric and invariant to translations and rotations.

$$k(x, x^{'}) = \sigma_l^2 \ exp(-\frac{\left\|x - x^{'}\right\|^2}{2l^2}) + \sigma_n^2 \delta_{ij} \tag{2.16}$$

Because the application of the kernel is to build the Graham matrix, the Graham matrix is symmetric positive semi-defined matrix; from this property, kernels can be added and multiplied by each other, thus allowing the creation of more sophisticated and complex kernels. For this reasons, kernels allow GPs to be extremely flexible but at the same time require a deep understanding of the process behind the data, otherwise they could be not able to represent the data in an efficient way and lead to errors and wrong predictions. Kernels are usually composed by parameters that allow the tweaking of the interaction between the pairing of the inputs, these parameters are called *hyperparameters*. In the RBF kernel (equation 2.16) there are 3 *hyperparameters* (σ_l, l, σ_n) where σ_n is a *hyperparameters* associated to the noise of the GP.

Varying the *hyperparameters* will change the behaviour of the model and the optimization of these coefficients represent an important part of the training for a GP . While optimizing the *hyperparameters* is important, it is also a difficult task; most of the methods utilized in adjusting these parameters are heuristic procedures like cross-validation or computationally expensive methods as maximizing functions like the *log marginal likelihood*. The latter can be easily defined from the bayes theorem as the integral of the likelihood times the prior [27]:

$$log \ p(y|X, \theta) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}log \left|K + \sigma_n^2 I\right| - \frac{n}{2}log \ 2\pi \tag{2.17}$$

where θ are the *hyperparameters* of the model. From the log marginal likelihood equation it seems that the *hyperparameters* are not present but the covariance matrix inside is ruled by these parameters. Inside the equation we find three terms:

$-\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 I)^{-1}\mathbf{y}$ the data-fit term
$-\frac{1}{2}log \left|K + \sigma_n^2 I\right|$ the complexity penalty
$-\frac{n}{2}log \ 2\pi$ the normalization constant.

To maximize the log marginal likelihood, we must compute at every iteration of the hyperparameters set the inverse of the covariance $(K + \sigma_n^2 I)^{-1}$, an operation that fits the model of the GP (training). In Figure 2.3 is displayed an example of GP.

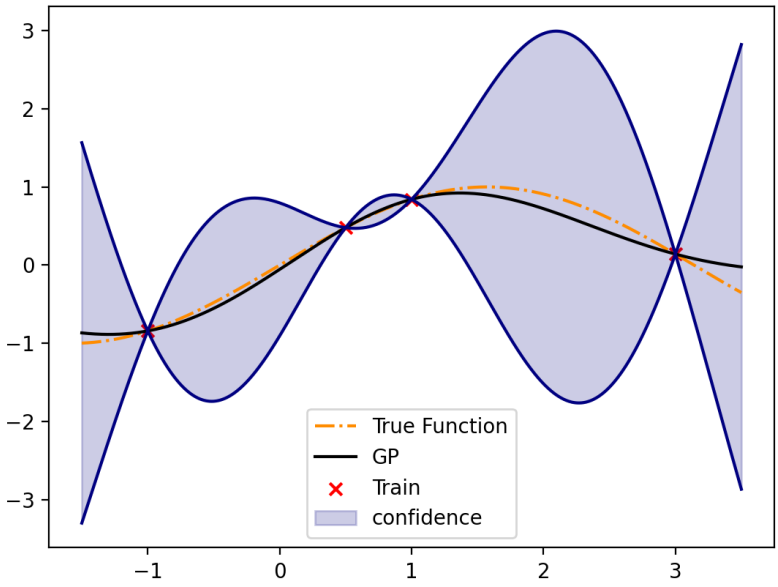


Figure 2.3: One dimensional gaussian process trained on 4 random points sampled from $f(x) = sin(x)$. The gaussian process utilizes a RBF kernel with $(\sigma_l, l, \sigma_n) = (2.0, 2.0, 0.005)$.

2.2.4 Bayesian Optimization

While the GP is a technique used as a data-driven approach to fit a function or to forecast a value; the bayesian optimization (BO) is a tool to search for the maximum (or the minimum) of expensive black box functions. Mathematically, the problem regards finding the maximum (or the minimum) of a black box function f can be written as [30]:

$$\mathbf{x}^* = \underset{x \epsilon \chi}{arg\ max} f(x) \tag{2.18}$$

in which χ is the design space of the optimization. To achieve that, the BO utilizes the bayes theorem to compute the posterior distribution over the various observations $D = \{x_i, f(x_i)\}$ and use that as a surrogate model to estimate the black box function f. While in a BO various type of surrogate models can be utilized (parametric models, linear models, generalized linear models just to name a few)[30], this work focus on the use of GP as a surrogate model. Using a GP as our surrogate model (or in a bayesian setting as our prior) we imply that the black box function (or objective function) is continuous, that the prior is homogeneous and the optimization is independent of the m^{th} differences. Moreover, the variance will tend to converge to zero only if the distance to the nearest observation is zero and absence of a noise [31]. This allows to formally write the previous problem as:

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}^{'})) \tag{2.19}$$

Here, the use of the standard assumption that the mean of the prior is the zero function $\mu(x) = 0$ (equation 2.19).

The main assumption of the BO framework and its convergence is the implied smoothness of the objective function. By utilizing the GP as surrogate model we can control this property by selecting an appropriate kernel (and thus the covariance function), and adjusting it by tweaking the *hyperparameters*. The most commmon choices for kernels are[32]:

- squared exponential kernel or RBF
- matern32
- matern52
- anysotropic squared exponential kernel.

By building this surrogate model we are shaping a prior over our objective function and defining a posterior distribution over the function space by utilizing this prior and the data. We then use this evidence and the knowledge of the prior to maximize the posterior at every iteration, in this way every new sample or evaluation decrease the distance between the global maximum of the objective function and the predicted maximum of the surrogate model. The bayesian optimization samples the new predicted maximum by exploting an utility function (known as acquisition function). BO with GP as surrogate model transforms the beginning problem of optimizing a black box function into a problem of double optimization; the optimization of the GP that can be tackled as explained in the section 2.1 with the maximization of the log marginal likelihood $max\ p(y|X, \theta)$, and tthe optimization of the surrogate model as a maximization of the chosen acquisition function (AC). Mathematically, instead of solving equation 2.18, we are resolving the following problem:

$$\underset{x \epsilon \chi}{arg\ max}\ \alpha(x|D) \tag{2.20}$$

Most common AC in the scientific literatures are [32]:

- Expected improvement (EI)
- Confidence bound criteria (LCB or UCB)
- Probability of improvement (PI).

In Figure 2.4 an iteration of a BO with AC displayed can be seen. The EI can be written as an integral over the normal posterior distribution characterized by the mean and the variance from the normal density function[31] [32]:

$$\alpha_{EI}(\mathbf{x}, \theta, D) = \int_y max(0, y_{best} - y) p(y|\mathbf{x}, \theta, D) dy \tag{2.21}$$

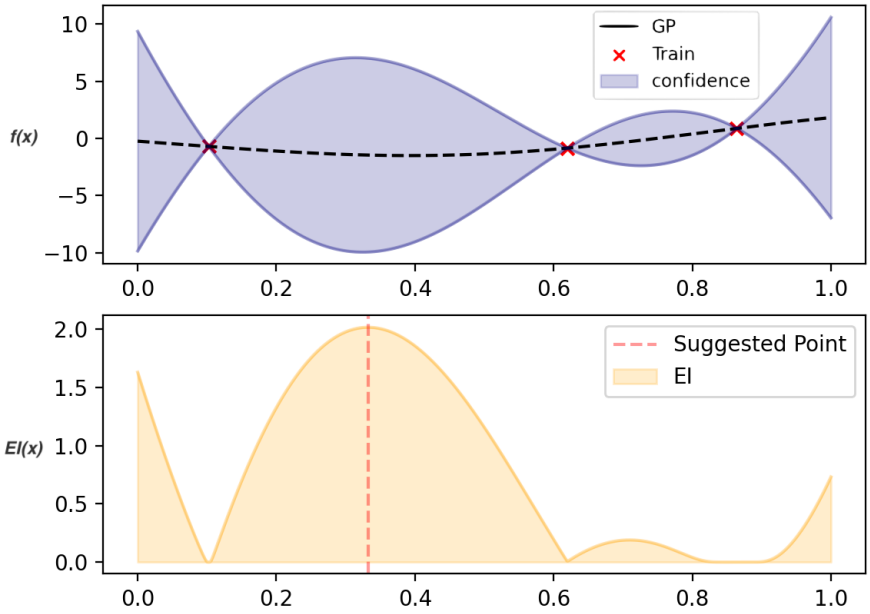


Figure 2.4: Bayesian optimization on a 1D toy model with GP as surrogate model utilizing the EI as acquisition function. The GP surrogate model uses a RBF with hyperparameters (σ_l,l)=(1.0,1.2) and a noise parameters of 0.0005.

and can be written analytically as: [33]:

$$\begin{aligned} \alpha_{EI}(\mathbf{x},\theta,D) &= (\mu(\mathbf{x}) - f(\mathbf{x}^*)\Phi(Z)) + \sigma(\mathbf{x})\phi(Z) \\ Z &= \frac{\mu(\mathbf{x}) - f(\mathbf{x}^*) + \varepsilon}{\sigma(\mathbf{x})} \end{aligned} \tag{2.22}$$

in which the parameter ε denote a coefficient to balance the trade-off between exploring and exploiting and it is usually set at 0.01, Φ is the cumulative distribution function of the standard normal distribution, ϕ is the probability density function of the standard normal distribution and $\mathbf{x}^*$ is the point of maximum observation of the evaluations in D.

2.3 Machine Learning in Molecular Simulation

Machine Learning techniques have seen an incredible spike of interest in the recent years and even more on their cutting-edge applications in the field of molecular

simulations.These techniques are starting to be seen as an interesting alternative to more common empirical or physical methods, as a tool to speed up the long time required to utilize them [34][35].

This is the case of Machine Learning force fields (ML-FF), models learned from data extracted by empirical or physical potentials exploiting them to propose data-driven potentials that requires less computational time [28] and are able to obtain similar accuracy of interatomic potentials[36] or *ab-initio* methods[37].

A different application of these ML techniques is the use of supervised learning models to optimize the parameters of different force fields [38]. This approach allows to achieve better results than the conventional techniques and also allows for the creation of different potential as a combination of parameters.

2.3.1 Machine Learning Force Fields

Because of the long computational time required to simulate a system in quantum or atomistic level, ML has been widely used to formulate and create data driven force fields that are able to utilize informations about the local atoms environment to predict the potential energy or accellerate the computation time of these simulations. While different supervised techniques can be used, this field has seen a large employment of artificial neural networks (ANN) and gaussian process regression (or GP) as the *state of the art* models.

All these supervised methods approach the regression problem by employing a common machine learning workflow of building a dataset, training a model and validation. The work of Behler [28] set the *state of the art* for the topology of the ANN and was used by Patra to scale-up from atomistic to mesoscale a system of liquid water [39]. Their model use Behler-Parrinello symmetric functions as descriptor to embed the informations of the local atoms inside the cut-off of a sphere [28]. Another interesting approach was used by Zeni [40] and showed how a GP can be utilized to create a ML-FF for clusters of metallic nanoparticles by capturing the many body interactions by accurately choosing the model kernel.

2.3.2 Optimization of Force Fields

While most of ML-FF techniques focus on the transition from quantum to atomic level or a speed-up of the calculation by utilizing a ML technique, optimization techniques adress the challenge of scaling up from atomistic to mesoscale. The work of Bejagam showed how a particle swarm optimization assisted by an ANN improves the parametrization of a CG force field [41].

More inspiring for this thesis were the works of Dequidt [42] and McDonagh [38] on the optimization of a DPD force field parameters.
Dequidt utilized a bayesian statistical approach to maximize the likelihood and match the trajectories of an AA simulations. Dequidt showed that by appropriately choosing the analytical interactions form, their method can optimize the parameters of the force field by a bottom-up force-matching approach [42]. McDonagh instead, utilized a 7 dimensional BO-GP with a RBF kernel to minimize a L2 loss function $\sqrt{\sum_{i=1}^{N}(|x_i^{experimental} - x_i^{model}|)}$ and find the best interactions parameters a_{ij} for a simple system composed of molecules made up of alkanes and primary alcohols. This top-down approach exploit the partition coefficient P. The work of McDonagh showed how a BO-GP converges to better values and with less iterations than a random search for this high dimensional spaces and for expensive black box function[38].

While the scaling-up approach from quantum to atomistic level is covered by various type of ML techniques, the scaling-up from atomistic to mesoscale is still an open issue where tradititonal techniques are still mostly used. For adressing this challenge most machine learning methods seeks to optimize *state of the art* physics force fields, while the few ML-FF ones are not able to fully consider the influence of the coarse-graining or the drastic changes of atom environments. Traditional methods instead, empirically extract the needed parameters from macroscopic experiments or calculate these parameters from small and focused AA simulations of the systems. These methods not only require a large time investment, but they are also system dependents.

Because of these reasons and being inspired from the works of Dequidt [42] and McDonagh [38], my technique aims to utilize a BO-GP strategy to optimize the parameters of a DPD force field by utilizing a bottom-up approach and a structure matching strategy. While the workflow presented in this thesis takes inspiration from the previous studies, the objective of this work is not only to employ their proven methodology, but to improve them by utilizing a bottom-up approach and apply it to a more complex case to overcome the drawbacks of the more conventional methods.

Chapter 3

Methodology

This chapter aims to present the systems which focused we used here for developing machine-learning based approaches for coarse-grain simulations. The structure is subdivided into two section, Simulation and Optimization. In the Simulation part, both the system and the simulations task are explained; while in the Optimization part, the main choices for building the framework are covered and explained without diving into the code implementation.

3.1 Simulation

3.1.1 The System

For this work a complex polymer-based system was studied, this system was the simplest one from a family of self-assembling chain-modified nanoparticles. Although this system has been described as simple for its shorter alkyl chain, the charged terminal group and the absence of chain bundling, it is more complex than the reviewed literature. The nanoparticle is composed by a gold core and it is functionalized by 326 alkyl chains (N16) (Figure 3.1) attaining a size of $\phi \sim 4.2$ nm.

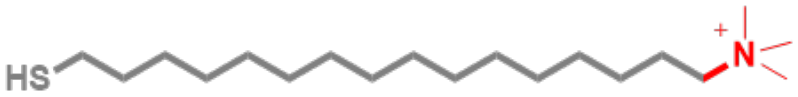


Figure 3.1: Chemical structure of the polymer ligand (N16 chain).

Since for our approach requires both the all-atoms (AA) trajectory and coarse-grained (CG) trajectory, we performed the AA and CG simulations as explained below.

3.1.2 Details of All-Atoms Simulations

The N16 ligand was prepared using antechamber and assigning gaff2[43][44] atom types; partial charges were calculated applying the RESP method provided by RED[45] server.
Au-Au interactions were described with the parameters of INTERFACE[46] force field for metals. An icosahedral gold core was built using OpenMD. A harmonic bond was created between each sulfur atom and a gold atom within 3.3 Å with a spring constant 50.000 kJ/mol*nm^2.[47] Although this interface structure disregards possible goldsulfur binding motifs, it has been shown recently[47] that this simplified treatment yields a description of the structure of self-assembled alkanethiols of various length (n = 3-15) on 2-6 nm size gold core in agreement with experiments.
The system was then solvated with TIP3P water molecules, extending at least 20 Å from each solute atom, and counterions added to neutralize the system. A combination of steepest descent (10000 cycles) and conjugate gradient (10000 cycles), followed by a heating phase of 100 ps in NVT ensemble (integration step = 1 fs), was carried out to reach the production temperature of 300 K. Then, density was brought to its final value with at least 50 ns in NPT conditions (integration step = 2 fs, pressure 1 atm), and pressure was maintained by Berendsen barostat. Finally, we switched to Monte Carlo barostat for production run, of which the first part was discarded until steady-state of ligands RMSD was reached.
Trajectory for final ensemble averages (400 ns) was stored from this point on. Temperature was controlled by Langevin method (damping coefficient of 5 ps^{-1}) throughout all simulations. Electrostatic interactions were computed by means of Particle Mesh Ewald (PME) algorithm, and calculations were carried out using AMBER 18 [48][49].

3.1.3 Details of Coarse-grained Simulations

We assumed the following level of coarse-graining (Figure 3.2)[7]:

- 2:1 mapping for the alkyl chain (8 C bead)
- 1:1 mapping for the methyl group (3 L bead)
- 1:1 mapping for the terminal atom (1 N bead)
- 1:1 mapping for the sulfur group (1 S bead).

The solvent was coarse-grained utilizing a 5 to 1 mapping (5 water molecules = 1 water beads (W)) and the Au core was composed in two types of Au beads:

- AuI: internal gold
- AuE: external gold atoms (shell).

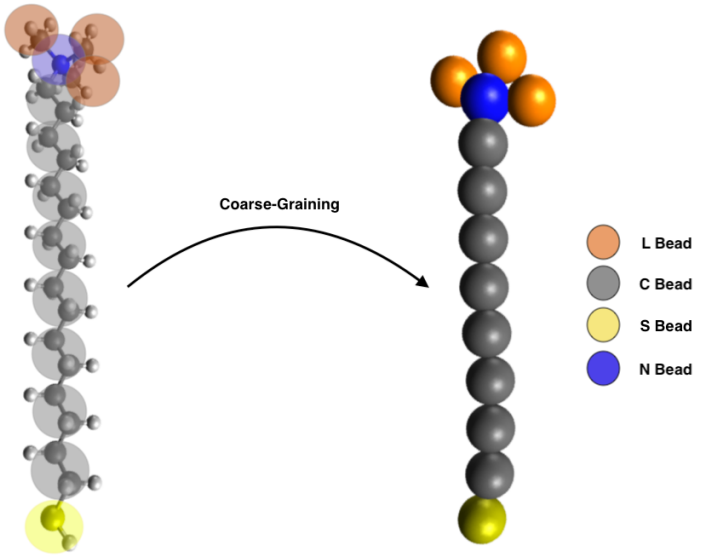


Figure 3.2: Visualization of the mapping utilized for the N16 ligand.

As the terminal bead of every chain is charged, a negative charged bead (Cl) was added in the solvent for every positive charged beads and the full system was prepared as showed below. The mapping of the whole system is reported in Figure 3.3

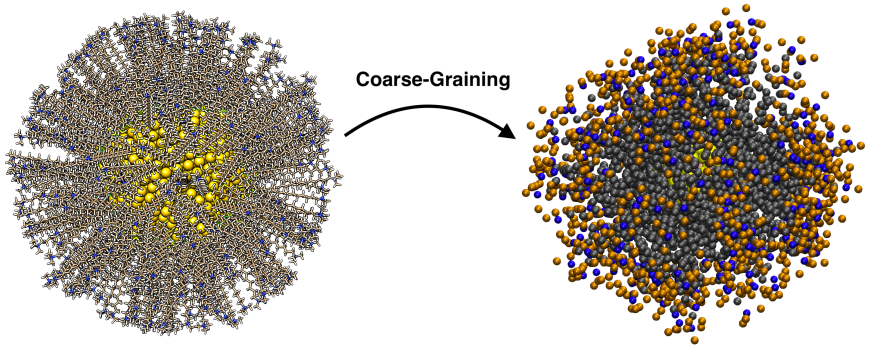


Figure 3.3: All-atoms and coarse-grained structure funtionalized nanoparticle studied in this thesis.

3.1.4 Methods of Coarse-Graining Simulations

The structure of the gold core was constructed by arranging Au DPD beads on an *fcc* lattice into an icosahedron of $\phi \sim 4.2$ nm size using OpenMD. Each ligand was represented by a flexible chain model of beads connected by harmonic springs.
326 ligands was added to the system. Each ligand was placed close to the gold surface and oriented outward with the head-tail vector along the radial direction, ensuring that the corresponding position on the surface did not overlap with any previously positioned ligand using Packmol. Then, the NP was solvated with W and Cl beads by Packmol[50], assuming a bead density of $\rho = 3$ [51].
A $32r_c$ x $32r_c$ x $32r_c$ simulation box was adopted, placing the NP in the middle of a 3D periodic cell. To avoid finite size effect, the distribution of the solvent from the center of mass of the NP was checked in preliminary simulation runs. All Au beads were forced to move as a rigid body during the calculation. At the same time, the sulfur heads can diffuse laterally on the NP surface during the entire simulation time. To help the adhesion and the bonding of the S beads with the gold shell, an artificial wallin potential was also implemented; the only objective of this potential is to set the S beads near the shell in the first steps of equilibration. Although this potential is always present during the simulation, it usually takes values close to 0 rather quickly assuring us that the S beads are near the gold shell and also that the potential is not interfering with the system.
Each configuration was first relaxed for $5\text{x}10^3$ steps and a time step of $\Delta\text{t} = 0.01\ \tau$. Then, at least additional $1\text{x}10^5$ time steps ($\Delta\text{t} = 0.02\tau$) were performed for productive runs. System equilibration was assessed monitoring temperature, pressure, density, and potential energy behaviour. The force cut-off radius r_c ($r_c = 0.56$), the particle mass m_i, and $k_B\text{T}$ (where k_B is the Boltzmann factor and T is the temperature) were taken as units of length, mass and energy. Calculations were carried out using LAMMPS [52].

3.1.5 Analysis of Molecular Trajectory

Every trajectory obtained from the simulations was analyzed to extract various data, namely the radial distribution functions (RDF) of each system components. The thermodynamic informations of the systems were obtained from LAMMPS [52] custom logging and were processed to extract averages and instant values of temperature, pressure, density and energies.
The reference AA trajectory obtained from the simulation for the AA model was analyzed and the RDFs (Figure 3.4) were extracted by pytraj [53] and other python scripts developed in house. CG trajectories were analyzed as well by using mdanalysis [53] and as hoc python scripts. The RDFs were normalized utilizing the number of particles instead of the spherical shell volume.
VMD [54] was used for the visual representation of the system and visual analysis.

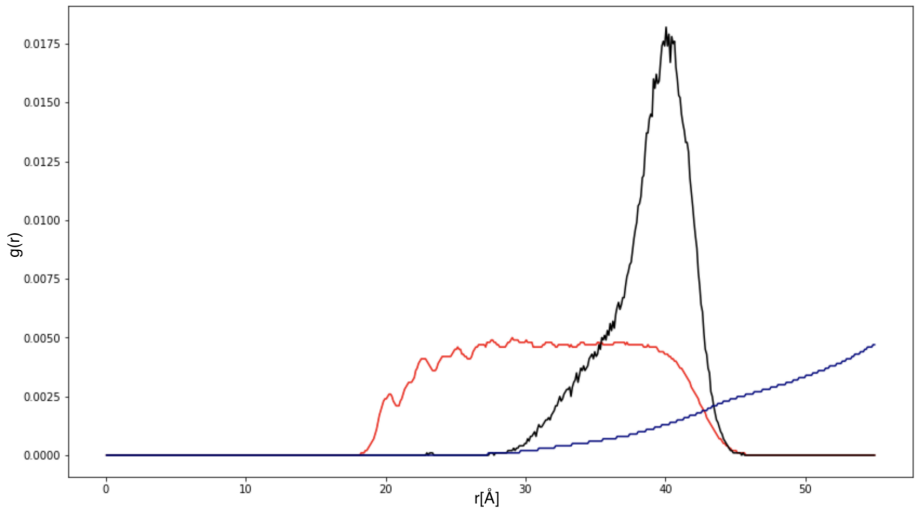


Figure 3.4: Reference RDF calculated from all-atoms simulation and normalized by the number of atoms. Color legend: red, chains; black, charged terminal group; blue, water.

3.2 Optimization

To optimize the DPD force field we choose to use a bayesian optimization (BO) on a gaussian process (GP) as a surrogate model, the implementation of both was made originally for this work by me.

3.2.1 Loss Function

The loss function was chosen upon the work of McDonagh[38]. Even if his work was focused on a top-down approach, the L_2 loss function is commonly used for its symmetric property and it was used by Florian for his structural approach [18]. There is not a clear way to choose a good scalar coefficient to embed the AA informations, we decided to use a structural approach and the RDFs from the AA simulation of the system.
The following RDFs were analyzed at each iteration:

- Chains (N16)
- Terminal group (N)
- Water (W).

By choosing these three RDFs as our targets, we obtain the following loss function to minimize:

$$L_2 = \sigma_1 L_2^{N16} + \sigma_2 L_2^{N} + \sigma_3 L_2^{W} \tag{3.1}$$

where $L_2^{i-j} = \sqrt{\sum^{bins} |RDF^{CG}(i,j) - RDF^{AA}(i,j)|^2}$ is the loss function for the selected pairs and $(\sigma_1, \sigma_2, \sigma_3) = (130, 100, 375)$ are weights to consider the difference of the values of these radial distribution functions.

3.2.2 Interactions Space

A consistent part of the complexity of the optimization was due to the interactions space of the system.
As the system used for this work was made up by 8 different bead types, the number of interaction parameters can be easily calculated:

$$n\# = \frac{(n+k-1)!}{k!(n-1)!} = 36 \tag{3.2}$$

as the pair interaction means that k=2, we get that our systems is composed by 36 different interactions and consequently, 36 different parameters.
While the complete set of different interactions is showed in table 3.1, to reduce the large interaction space it was decided to constrain the interaction space to simplify the system without altering the chemical information associated with each parameter.

$\mathbf{a}_{S-x}$ The S beads are extremely important to achieve a good adhesion to the external shell of the gold core, we chose to neglect the optimization of these interactions beside the one of the S-S pair. For the other S bead interactions, values from similar system were utilized[7][8].

$\mathbf{a}_{L-x}$ We assumed they have the same interactions as the bead C

$\mathbf{a}_{AuI-x}$ The interactions between the internal Au beads against other beads (beside C) were considered the same as the external Au beads (AuE). We believe that these interactions do not alter our system as the internal shell doesn't interact with most part of the system.

$\mathbf{a}_{AuI-AuE}$ The interaction within the core beads (AuI and AuE) were not optimized but instead fixed at a standard value. This choice is also supported by the assumption of the rigid behaviour of the core.

Boundaries The boundaries of the optimization problem were fixed:

$$a_{ij} \in [-10{,}140] \tag{3.3}$$

We decided for a negative value as the lower bound to search solutions both the attractive and repulsive. Because of this choice, the solution was searched on the hypercube $x_i = (-10{,}140)$ for all 13 interactions coefficients.
With all these constrains our interactions space was reduced to 13 coefficients to optimize, 13 fixed interaction coefficients and 10 linear dependent coefficients as it is summarized in Table 3.1.
Even after adding these constrains to the optimization space, the system was more complex than the ones found the scientific literature, not only for the number of dimensions of the optimization (n=13) but also for the boundaries of the system [38][55].

Bead 1	Bead 2	Optimize	Lower Bound	Upper Bound	Value
AuE	C	YES	-10	140	
AuE	N	YES	-10	140	
AuE	W	YES	-10	140	
AuE	Cl	YES	-10	140	
AuI	C	YES	-10	140	
C	C	YES	-10	140	
C	N	YES	-10	140	
C	W	YES	-10	140	
C	Cl	YES	-10	140	
N	N	YES	-10	140	
N	W	YES	-10	140	
N	Cl	YES	-10	140	
S	S	YES	-10	140	
AuE	L	As C	-10	140	
AuI	N	As AuE			
AuI	L	As AuE			
AuI	W	As AuE			
AuI	Cl	As AuE			
C	L	As C			
N	L	As C			
L	L	As C			
L	W	As C			
L	Cl	As C			
AuE	AuE	NO			51.6
AuE	AuI	NO			51.6
AuE	S	NO			-10.0
AuI	AuI	NO			51.6
AuI	S	NO			40.0
C	S	NO			72.0
N	S	NO			68.9
L	S	NO			72.0
S	W	NO			80.0
S	Cl	NO			80.0
W	W	NO			51.6
W	Cl	NO			51.6
Cl	Cl	NO			51.6

Table 3.1: The table shows the DPD interaction parameters, the boundaries of the optimization, the values of the fixed parameters and the constrained parameters.

3.2.3 Sampling of the Interactions Space

One of the biggest problem of high dimension BO is the efficient sampling of the n dimensional space to probe the acquisition function in order to search for the minima. This problem is usually referred as *The curse of dimensionality* by Bellman[56] and can be easily seen by the exponential increase of pairs with the increase of the dimensions for a fixed number of points and is represented by the formula $p = n^d$.
Different techniques were tested to improve the sampling of the space [26]:

- Grid search
- Random sampling
- Sobol sequence
- Latin hypercube sampling (LHS).

The technique utilized by McDonagh [38] of random sampling was also utilized to generate a randomized grid at each iteration. While with few dimensions grid sampling can also be efficient to create a dense grid, the more we increase the problem dimension the more our points will be distant from each others. Because of this, grid search start to fall short and there is a need of a new sampling technique to tackle this issue.
From all the sampling techniques the most promising ones are the LHS and the uniform sampling with the cartesian product. As the Figure 3.5 shows, the quasi-random LHS covers the space more evenly while being randomized, whereas the uniform sampling creates more local dense spots of points.
More progress can be made in the sampling as both of them have the limitation of not retaining the previous informations of the sampling space.

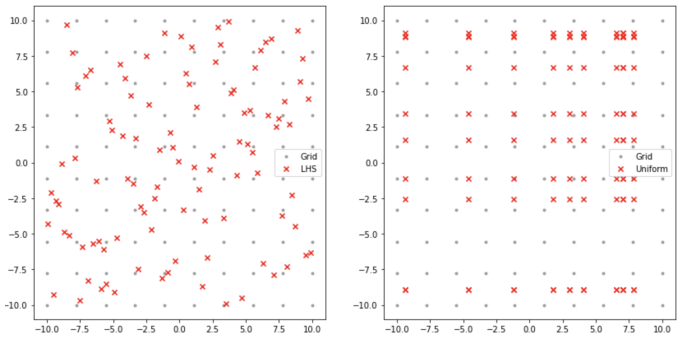


Figure 3.5: Example of the different sampling techniques: LHS, grid sampling and uniform sampling.

3.2.4 Optimization of the Acquisition Function

The main optimization routine, as already described in the BO section, requires to find the maxima of the acquisition function.
The framework I developed can utilize three different techniques:

- Random sampling
- Direct optimization
- L-BFGS-S.

The random sampling technique utilizes the cartesian product method used by McDonagh [38] and it quickly build a randomized sample grid to test the AC on n-dimension. This grid is utilized to sample the AC and to search for the minima. The Direct and L-BFGS-S techniques are implemented as numerical solvers for the AC. An example of an optimization is showed in 3.6

3.2.5 Programs and Frameworks

The implementation of the framework for this project was done by using the following:

- Python 3 (>3.6)
- Numpy [57]
- Matplotlib [58]
- Scipy [59].

The main routines for the optimization were done by using bash scripts; more informations about the framework code can be found in Appendix A.

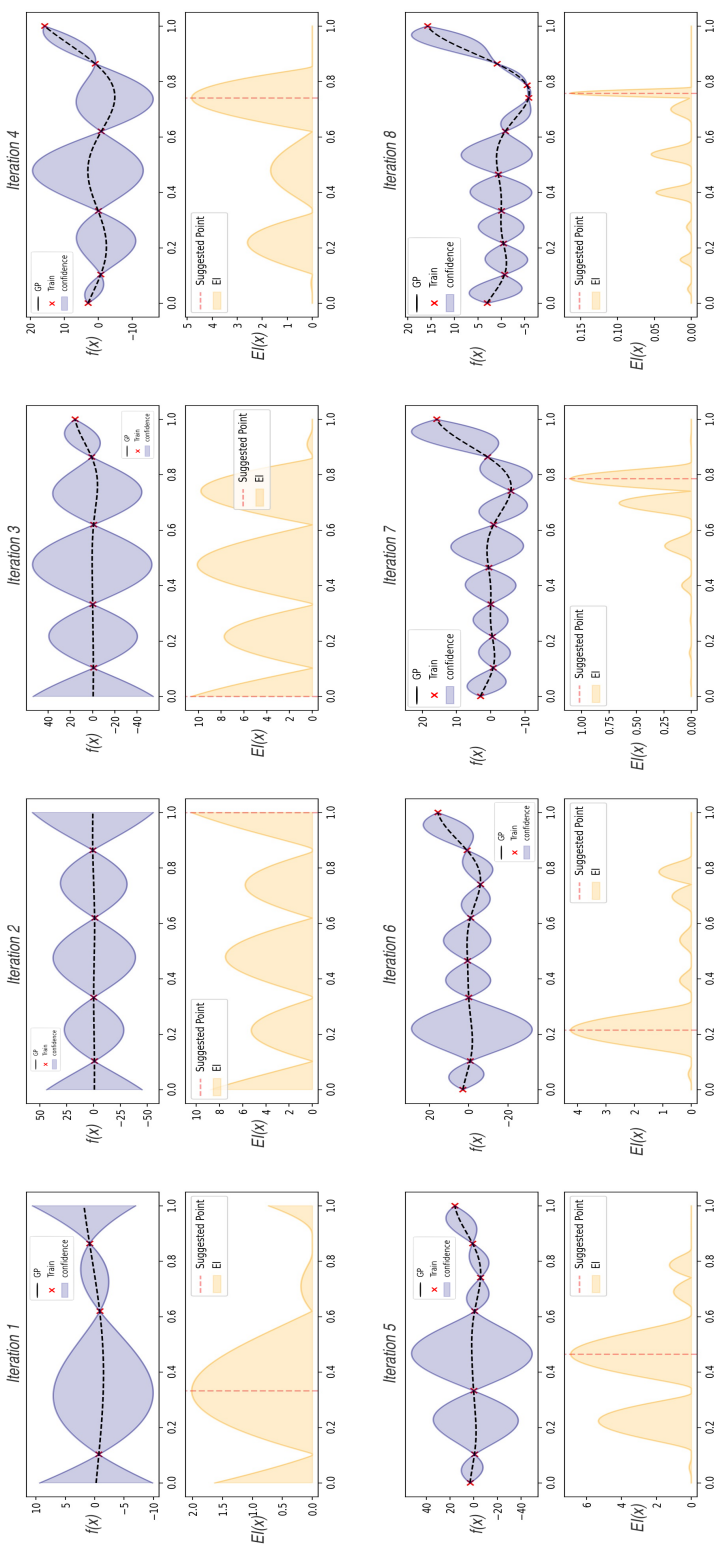


Figure 3.6: Bayesian optimization with expected improvement as acquisition function of a 1D toy model. The AC function is sampled using the random sampling technique.

Chapter 4

Results

The structure of the chapter is divided in two section, Framework and Project. In the Framework the bayesian optimization (BO) framework built for this thesis is explained, while in the Project section the results of the optimization routines are illustrated and explained.

4.1 Framework

In this section more informations about the practical implementation and the testing of the framework will be covered.
While the code of the programs is not showed, some snippets and more about the code can be found in A, the main routines are illustrated as pseudo-code.

4.1.1 Implementation

While the description of the framework theory in Section 2.2.3-2.2.4, here we mention only those details that are important for further discussion of results.
As previously discussed, the GP is the surrogate model chosen for this BO technique. A standard procedure to train a GP is to calculate the inverse of the covariance matrix and rely the prediction phase on the formula 2.14. The main drawback for this procedure is the main drawback of the GP model: the computational complexity of the inversion of a matrix is $\mathcal{O}(n^3)$ where n is the number of training points; this computational complexity is one of the reason of the absence of GP in the field of *big data*.
Another numerical issue stems from the inversion of the covariance matrix that can cause numerical instability when the covariance matrix is a singular matrix. This problem is usually solved by adding a small noise hyperparameter even in the case of a noise-free prediction.

Instead of relying on the normal inversion, this framework is implemented with the routines suggest on [27] utilizing the Cholesky decomposition on the Graham matrix as it can be seen in the in algorithm 1.

Algorithm 1 GP Training and Prediction

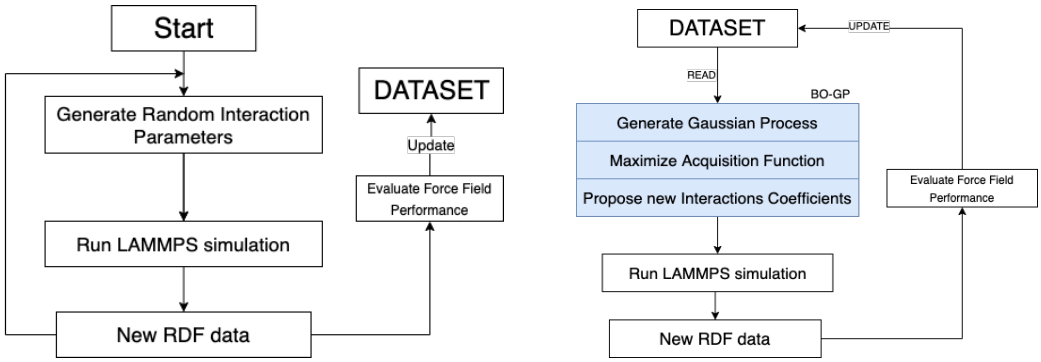
Input : X (inputs), Y (targets), k (kernel), σ_n^2 (noise), X_* (Test)
Compute the Graham Matrix: $K = k(X, X)$
$L = cholesky(K + \sigma_n^2 I)$
$\alpha = L^T/(L/Y)$
$\mathbf{v} = L/k_*$
Compute GP mean: $\mu = \mathbf{k_*}^T \alpha$
Compute GP var: $var = k(X_*, X_*) - \mathbf{v}^T\mathbf{v}$
Output : (μ, var)

Nevertheless, numerical instabilities can still be present in the Cholesky inversion, and the same procedure of adding a small noise σ_n for noise-free cases is still applied.
Furthermore, BO standard routine is shown as pseudo-code in algorithm 2.

Algorithm 2 Bayesian Optimization

for t = 1,2... **do**
    Generate surrogate model GP over the dataset$D_{1:t} = \{(x_i, y_i)\}$
    Find $\mathbf{x_{t+1}}$ by maximizing the AC
    Sample the objective function $y_{t+1} = f(x_{t+1})$
    Augment dataset with (x_{t+1}, y_{t+1})
end for

Before proposing a new point to sample, the framework checks that it is distant enough from the previous proposals by calculating $|x_{i:t} - x_{t+1}| > err$. This operation is usually not needed for most BO, but as we are using a GP as a surrogate model, points that are too close can eventually break the GP and disrupt the optimization run. This behaviour is unwanted, especially when the model is trapped in a local minima and start sampling the points around this location.
The general optimization routine consist in two parts: the inizialization and the optimization and both can be seen in Figure 4.1.
The initialization is used to generate the 100 points that compose the training dataset D for the interaction parameters optimization; every set of points is generated by respecting the constrains explained on section 3.2.2. At the end of each run, for each training point, the RDFs are calculated and the loss function evaluated before starting a new iteration (Figure 4.1(a)).



(a) Dataset Generation **(b)** Optimization Routine

Figure 4.1: Flow chart of the inizialization of the dataset and the main routine of optimization.

After building the dataset, the optimization routine started. At every iteration the framework optimize the GP by maximizing the log marginal likelihood (equation 2.17), and the BO-GP propose a new point to sample. The framework then generates the input files for the dissipative particle dynamics (DPD) simulation and calculate the radial distribution functions (RDF) and corresponding loss functions to evaluate the performance of the force field before updating the dataset and running next iterations (Figure 4.1(b)).

4.1.2 Convergence tests

To test the framework, I choose to run the optimization of some well described systems.
The Branin two dimensional function and the Hartmann six dimensional function are deemed fit for this task as they are usually chosen to demonstrate the optimization performance of other systems [60]. The Branin function is run on the framework for 20 iterations while the Harmann function is tested for 50 iterations, both of them starting from one training points. In both cases the expected improvement is used as acquisition function.

Branin 2D Function The Branin 2D function has 3 global minima. This optimization problem is usually resolved for $x_1 \in [-5,10],\ x_2 \in [0,15]$.

$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\ cos(x_1) + 10 \tag{4.1}$$

The three global minima with value $f(x_1, x_2) = 0.397887$ are located at the points $(-\pi,12.275), (\pi,2.275)$ and $(9.42478,2.475)$.

Hartmann 6D Function The Hartmann 6D Function has 6 local minima. This optimization problem is usually solved for the hypercube where $x_i \in [0,1]$ for $i = 1,2,...,6$.
In the figure 4.2 it can be seen the result of the test.

$$f(x) = -\sum_{i=1}^{4} \alpha_i \; exp(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2) \tag{4.2}$$

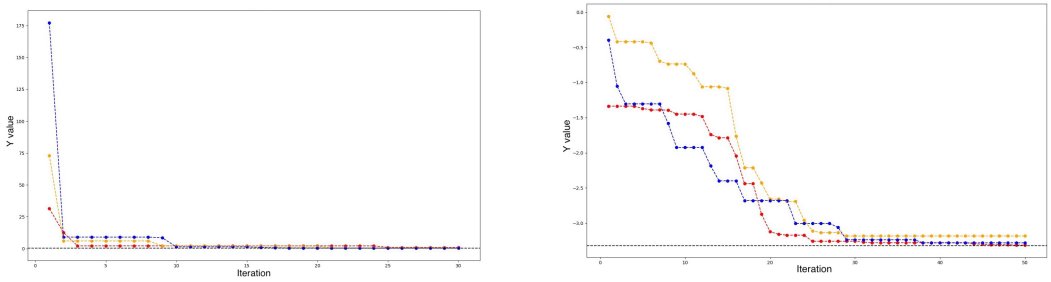
and

$$\alpha = (1.0,1.2,3.0,3.2)^T$$

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 6 & 0.06 & 10 & 0.1 & 14 \end{pmatrix} \tag{4.3}$$

$$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}$$

The global minima for this function is $f(x_1,...,x_6) = -3.32237$ located at the point (0.20169,0.150011,0.476874,0.27532,0.311652,0.6573).



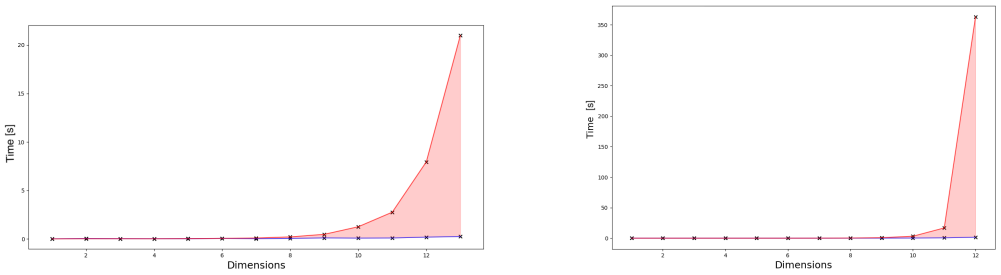
(a) Branin 2D function **(b)** Hartmann 6D function

Figure 4.2: Convergence rate for the Branin and Hartmann function. Color legend: red, L-BFGS-B solver; yellow, DIRECT solver; blue, random sampling.

All approaches appears reliable enough for a real optimization task but the DIRECT technique is more proned being stuck in local minima. Because of this, the NAIVE approach and the BFGS solver technique were chosen to optimize our CG model where the vast space of interactions will be searched.

In addition, we propose another test for computational time requirements for an optimization task for a model with different dimensions.
In this case the bottleneck of the framework lies in the prediction ($\mathcal{O}(n)$), as the

number of sample points is large.
The function chosen for this test is the multivariate gaussian distribution; which can be motivated by the straight forward implementation in higher dimensions. At each number of dimensions d, $p = n^d$ numbers of pairs are generated by the cartesian product from the n samples drawn from an uniform distribution. After generating the p input pairs, the single run of optimization starts and the time is evaluated; this procedure is done for 2,3,4 samples as it is shown in Figure 4.3.



(a) Color legend: red, 3 points; blue, 2 points

(b) Color legend: red, 4 points; blue, 3 points

Figure 4.3: Computation time for a single iteration of bayesian optimization on increasing number of dimensions d. The number of prediction is calculated with n^d where n is the number of points for a dimension.

The same routine was run on the 3 samples case to see the behaviour at even higher dimension (Figure 4.4).

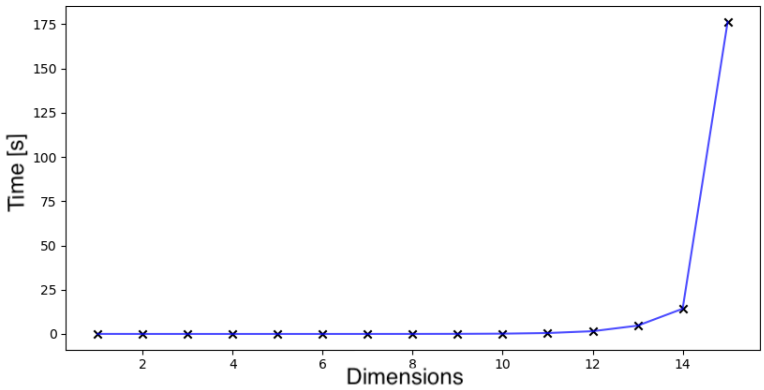


Figure 4.4: Computational time for a single iteration of bayesian optimization on increasing number of dimensions d on 3^d sample points for each dimensionality d.

From the figures, we extrapolated a large increment of time on each increase of

dimension; this was somehow expected as the number of testing points increases with an exponential behaviour meanwhile the complexity time of the sampling of the AC is $\mathcal{O}(n)$, as our surrogate model is a gaussian process.

We showed that the sampling of the space for the optimization of the acquisition function (AC) is a rather challenging task. As a consequence, the problem is addressed by different types of sampling to achieve uniform distribution of points in the space, but also we implemented various types of numerical methods to optimize high dimensional functions. By following the work of McDonagh [38], the first optimization routine of the interaction parameters is done with the his sampling method while the successive routines are completed by utilizing a bounded BFGS method as it brings better results in terms of finding the minima of the AC.

4.2 Project

As this chapter focus on the project evolution and results; as with every optimization a new challenge arose, the proposed workflow is derived directly from the choices made to tackle each challenge.

4.2.1 Evolution

The proposed optimization model is utilized to perform not only the optimization of the interaction parameters of the *Dissipative Particle Dynamics* (DPD) force field but also the optimization of bonds and angles parameters for the N16 polymer chain. The entire phase space of parameters to optimize would be, if optimized, of 54 dimensions. The parameters that I optimize are described in Sections 3.1.3-3.2.2 and take the analytical form of $\mathbf{F}_{ij}^{\mathbf{C}} = a_{ij}(1 - r_{ij})\hat{\mathbf{r}}_{\mathbf{ij}}$ (Section 2.1.2).
The sections below explains the phase space as a combination of various type of the different parameters used in the optimizations:

- a_{ij} : interaction parameters
- (K_i,θ_i) : angles parameters
- (K_j,r_j) : bonds parameters
- Free : parameters optimized
- Fixed : parameters with their value fixed
- Constrained : parameters whose value is assigned.

The Figure 4.5 shows the approach derived from the optimization of the DPD force field:

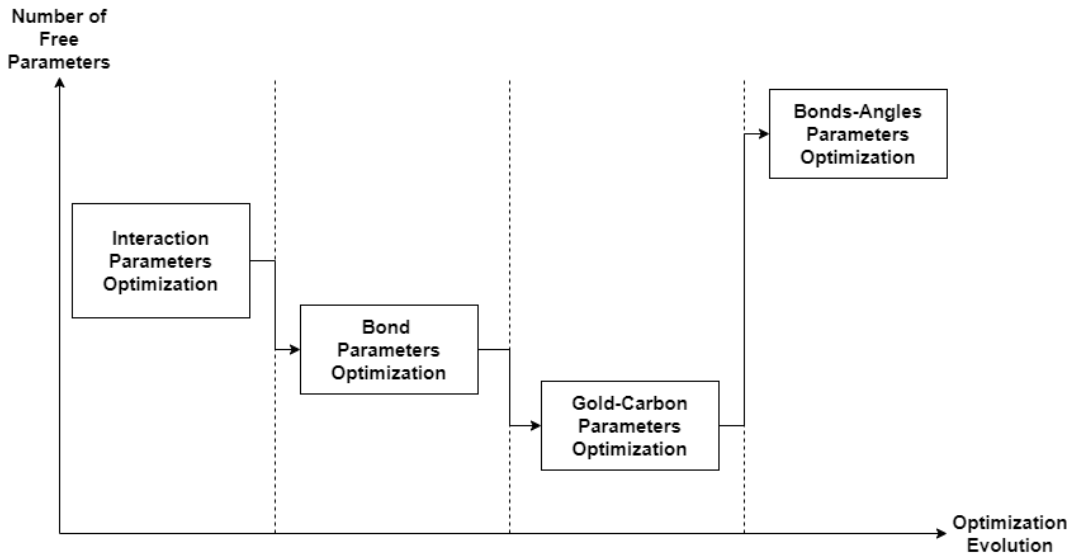


Figure 4.5: Evolution diagram of the bayesian optimization approach utilized in this work.

Interaction Parameters Optimization

The interaction parameters optimization is done in two step:

- full 13 interaction parameters optimization
- 8 interaction parameters optimization.

13 Parameters Optimization The first optimization is the optimization of the 13 interaction parameters with the constrains explained in the Section 3.3.2 and using the methodology utilized by McDonagh. The summary of the interactions parameters is provided in Table 4.1.

Type	**Free**	**Fixed**	**Constrained**
a_{ij}	13	13	10
K_i	0	5	0
ϕ_i	0	5	0
K_j	0	4	0
r_j	0	4	0

Table 4.1: Summary table for the interaction parameters optimization. Bond and angles interactions are not optimized and 13 parameters are improved.

From Figure 4.6 it is clear that the choice of different variables from the radial

distribution function (RDF), such as average temperature, average pressure or average total energy, would not have been optimal. In fact, for wrong physical configurations of the system, determinated by a high loss function value, the DPD force field doesn't show substantial deviations of the thermodynamic average values to distinguish these conformations from the more correct ones.

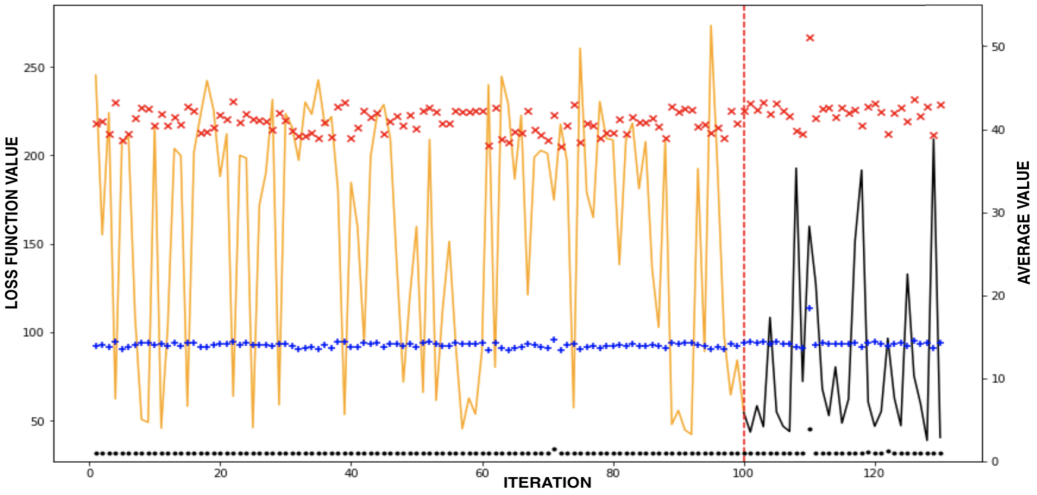


Figure 4.6: Loss function value and thermodynamics averaged values for 130 iterations. Color legend: yellow line, randomized run; red line, start of the optimization; black line, optimization run; black point: average temperature; blue point, average pressure; red point, average total energy.

At the end of the fixed number of steps, I verify the optimization performance by plotting the best value of the loss function at every iteration (Figure 4.7).

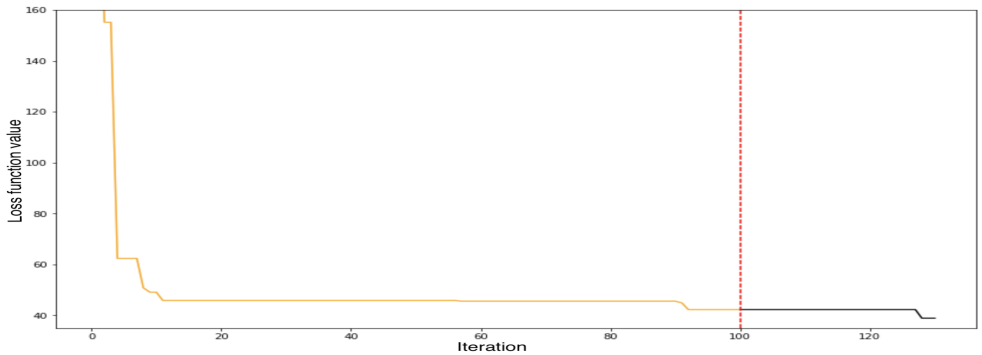


Figure 4.7: Convergence rate plot for the 130 iteraction parameters optimization run; color legend: yellow line, randomized run; red line, start of the optimization; black line, optimization run.

The sharpest improvement occurs during the dataset creation by the random sampling routine illustrated in Figure 4.1 (a), while in the optimization phase only obtain a marginal improvement of the loss function values. To ensure that the optimizer is not stuck near a local minima, I calculate the Euclidean distances between two consecutive observations; in this way I can be sure that the observations are sufficiently far apart (Figure 4.8).

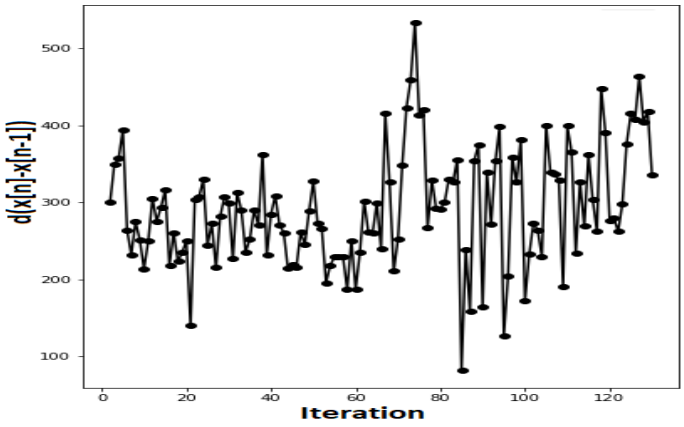


Figure 4.8: Euclidean distance between two consecutive observations on the interaction parameters optimization.

Since the observations are sufficiently apart from each other, the poor efficiency of the optimization is probably due to the large dimensionality of the problem. The method used here (Section 3.2.3) and by McDonagh[38] does not guarantee a grid dense enough to effectively maximize the acquisition function for such a large and complicated optimization space.
Figure 4.9 shows the system with the lowest loss function value for the set of iterations. Even if the value of the loss function for this system is high, the distribution of the terminal groups is similar to that one obtained from the atomistic simulation. To the contrary, the distributions of the chain and the solvent deviate from the atomistic reference. Since the distribution of the terminal group is correct, to improve the distribution of the chains and the water, I fix the 5 interaction parameters of the terminal group and run an optimization on the remaining parameters. This allows me to reduce the dimensionality of the system to 8 free parameters and to increase the sampling the AC.

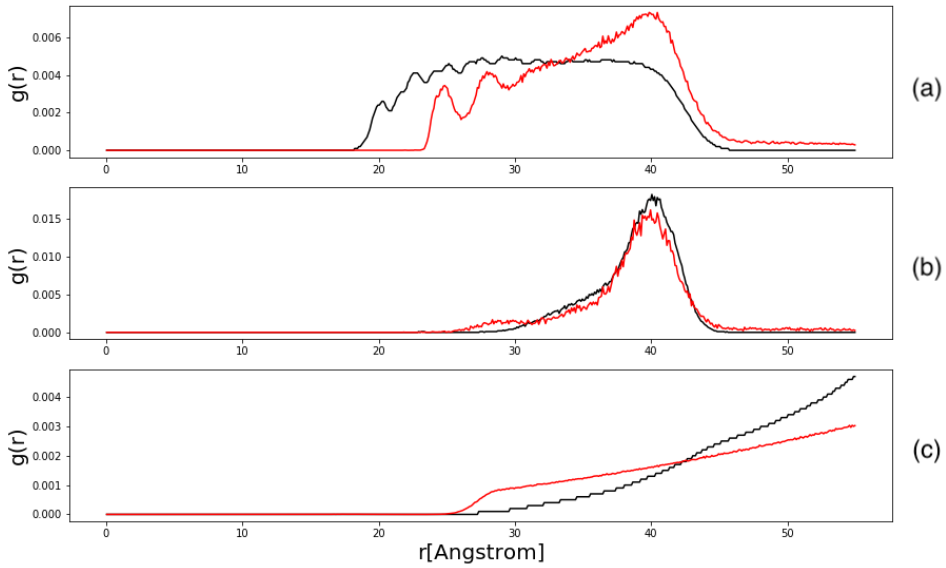


Figure 4.9: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the best system in the optimization of the 13 parameters from AA and CG simulations. Color legend: AA, black; CG, red. (a) N16 chains; (b) terminal group; (c) water. Parameters: AuE-C=128.9, AuE-N=127.1, AuE-W =90.5 ,AuE-Cl=-8.2, AuI-C=134.7, C-C=113.2, C-N= 2.52, C-W=138.1, C-Cl=24.2, N-N=-4.3, N-W=124.4, N-Cl=4.5, S-S=118.6.

8 Parameters Optimization Fixed the 5 parameters of the terminal group and with only 8 free parameters to optimize, I perform an optimization of 10 iterations to refine the behavior of the chains, the water and at the same time, to not modify the correct distribution of the terminal groups. The summary of the interactions parameters is showed in Table 4.2.

Type	**Free**	**Fixed**	**Constrained**
a_{ij}	8	18	10
K_i	0	5	0
θ_i	0	5	0
K_j	0	4	0
r_j	0	4	0

Table 4.2: Summary table of the interaction parameters optimization after fixing the terminal beads interactions.

In this optimization, from the visualization of the RDFs, I notice that no improvements have been found in the distribution of chains and water, moreover

the distribution of the terminal groups results modified despite the fixed parameters. Since the loss function from this optimization does not improve, I decide to continue the work using the best system found up in the 13 parameters optimization and already showed above in Figure 4.9.

Bonds Parameters Optimization

Since the RDF of the chains begins too far away from the center of mass of the nanoparticle and in addition, the chains are more compressed than the one obtained from the all-atoms simulation, I optimize bonds parameters of the N16 chain. Because of the number of dimensions in this optimization and the efficiency found in the previous routine, I decide to change approach and start to use a numerical solver (L-BFGS-B) to maximize the AC. In addition, I also decide to not include the RDF of the water molecules into the evaluation of the loss function from now on, after running some test systems to observe the behaviour of the water distribution (Section 4.2.2). In fact, despite the ability of the system to respond to variations in water related interaction parameters, a good matching between the AA and CG could not be achieved. We believe this is due to the coarser description of the water interface with the respect to AA calculation. However, this point deems further investigation.
Since I use the best system found so far, the interaction parameters are fixed and I optimize only the 8 bond parameters as showed in Table 4.3

Type	**Free**	**Fixed**	**Constrained**
a_{ij}	0	36	0
K_i	0	5	0
θ_i	0	5	0
K_j	4	0	0
r_j	4	0	0

Table 4.3: Summary table for the bonds parameters optimization. As the interaction parameters are fixed, only the stiffness K_j and the equilibration length r_j are optimized.

Since the BO works iteratively, I run the optimization for 50 iterations and build the dataset *on the fly*; for this problem I choose the following boundary conditions which span the common range of value for these parameters in DPD:

- $K_j \in$ (0.2,1.0)
- $r_j \in$ (40,180).

After the 50 iterations, Figure 4.10 shows not only that the optimization has improved the loss function associated with the starting system, but also that

the convergence rate and number of improvements are much more evident. By comparing it with previous optimization, the solver has improved the abilities of sampling the AC. In addition, the figure also shows the sampling of the solver and assure that it is not trapped in a plateau.

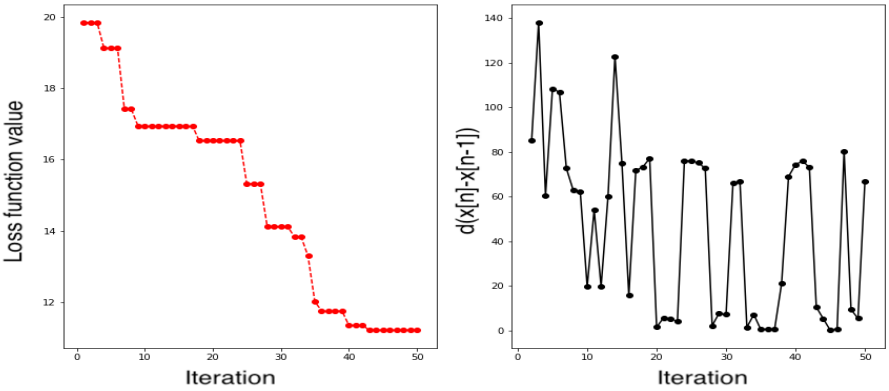
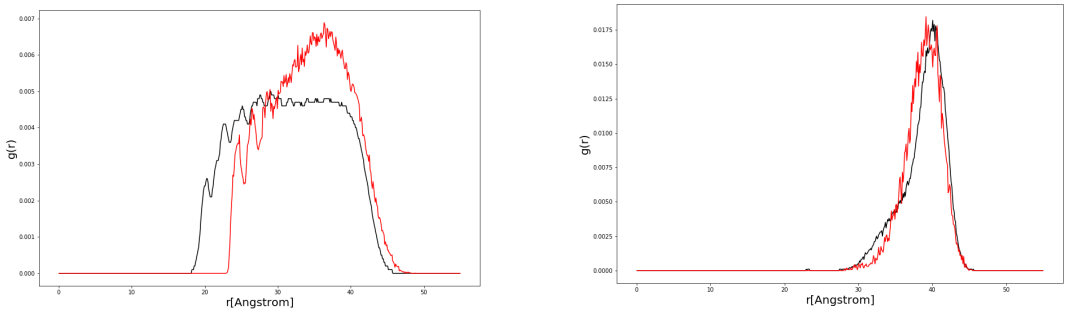


Figure 4.10: Convergence rate plot and Euclidean distance between two consecutive observations (right) on the 50 bond parameters optimization.

The best system in Figure 4.11 shows how there are no substantial improvements to the chain distributions even for the stiff bond parameters.



(a) N16 chains distribution **(b)** Terminal group N distribution

Figure 4.11: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the best system in the optimization of the bonds parameters from AA and CG simulations. Color legend: AA, black; CG, red. Parameters: K_1^a=95, K_2^a =124, K_3^a=125, K_4^a=125, r_1=0.2, r_2=0.22, r_3=0.85, r_4=0.83.

Gold-Carbon Parameters Optimization

I optimize the gold-carbon parameters as the key interactions between the gold nanoparticle and the carbon chains seems to not be captured yet. Since the nanoparticle interactions are prevalently between the S bead (fixed as explained in Section 3.2.2) and the C bead, constraining the optimization dimensions to only these two parameters allow to search a space of only 2 dimensions. The gold-carbon optimization seeks to optimize only the AuE-C and AuI-C parameters and is done in two step:

- global optimization
- local optimization.

The systems I utilize is the best one found in the bond optimization routine; as the bond parameters are fixed and the other interaction parameters are fixed too, beside AuE-C and AuI-C, the optimization space is less complex (Table 4.4):

Type	**Free**	**Fixed**	**Constrained**
a_{ij}	2	24	10
K_i	0	5	0
θ_i	0	5	0
K_j	0	4	0
r_j	0	4	0

Table 4.4: Summary table for the optimization of the gold-carbon interaction parameters.

Global Optimization The global optimization is constrained on the same boundaries of the interaction parameters optimization:

- AuE-C $\in$ (-10,140)
- AuI-C $\in$ (-10,140).

I run the optimization for 50 iterations and Figure 4.12 shows the convergence rate.

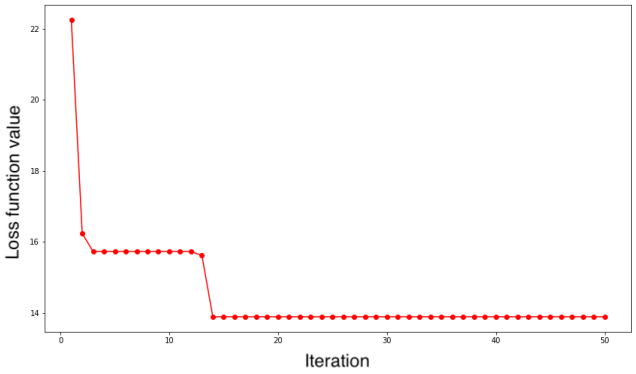


Figure 4.12: Convergence plot for the global Au-C optimization.

Since the loss function doesn't improve after the 14th iteration, I check the behaviour of the solver; as result of the the fewer dimensions on this optimization (n=2), instead of the previously euclidean distance plot to check for the solver behaviour, I utilize a contour plot to show the loss function 4.13.

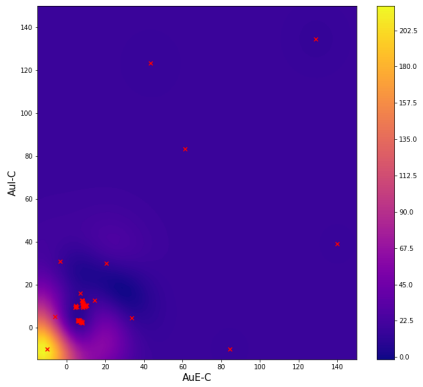
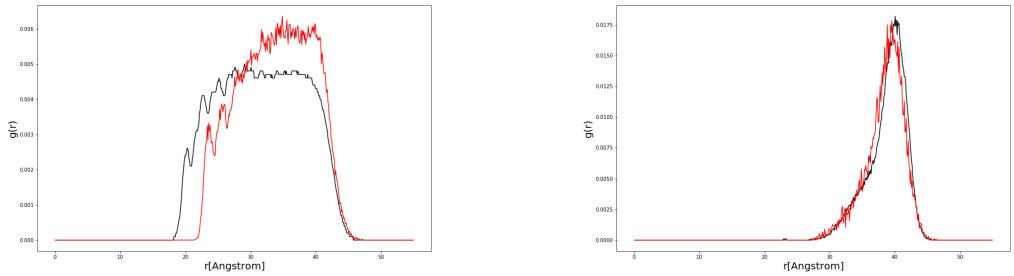


Figure 4.13: Gaussian process mean plot of the loss function evaluations for the AuE-C, AuI-C optimization on the (-10,140) boundaries.

Even by focusing only on 2 parameters the BO is not able to capture the starting point of the polymer chain distribution does not improve. The terminal group distribution and also the terminal point of the chains now, seems to follow the AA reference (Figure 4.14).



(a) N16 chains distribution

(b) Terminal group N distribution

Figure 4.14: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the best system in the global optimization of the AuE-C and AuE-C parameters from AA and CG simulations. Color legend: AA, black; CG, red. a_{AuE-C}=9.5, $a_{AuI-C} = 9.5$.

By visualizing the RDFs of the various systems, I noticed the occurrence of a shift of the initial peak to the left for attractive values of the AuE-C interaction parameters. To investigate this relationship I prepare different systems by changing the interaction parameter a_{AuE-C} to represent attractive interactions between the gold core and the alkyl chain.
Figure 4.15 shows how strong attractive values of AuE-C increase the presence of the peak at the correct distance, but also have a negative influence on the terminal group distribution as it increases the presence of the terminal group near the core surface.

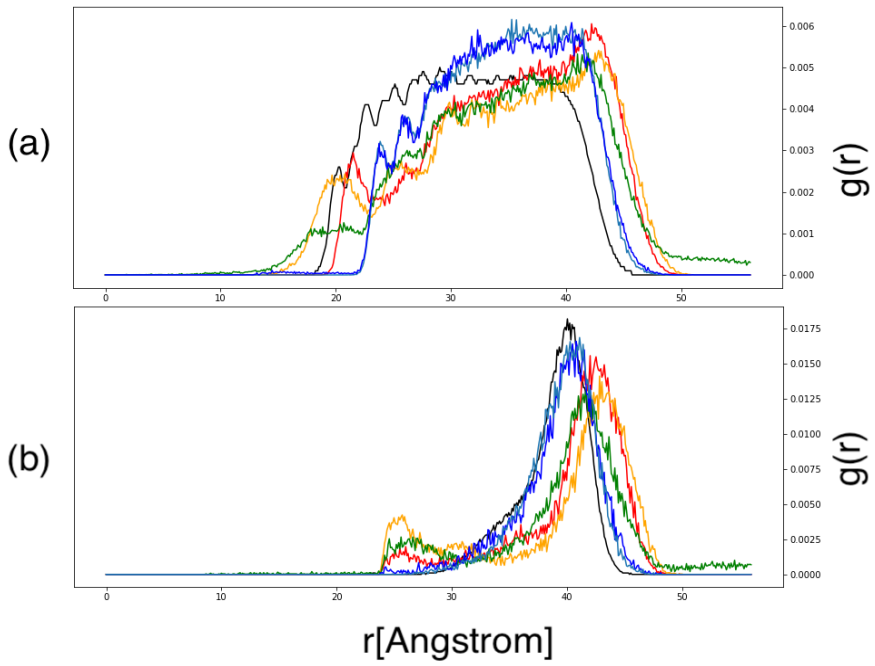


Figure 4.15: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for different test systems by changing the AuE-C parameters. (a) N16 chains distribution; (B) terminal group N distribution; Color legend: AA, black; CG (a_{AuE-C}=0,a_{AuI-C}=10), red; CG (a_{AuE-C}=-1,a_{AuI-C}=10), orange; CG (a_{AuE-C}=2,a_{AuI-C}=10), green; CG (a_{AuE-C}=6,a_{AuI-C}=10), cyan; CG (a_{AuE-C}=10,a_{AuI-C}=10), navy.

Local Optimization After testing the AuE-C behaviour, I decide to run an optimization of the same gold-carbon parameters on a limited space, this local optimization is done on a portion of the phase space of strong attractive interaction parameters:

- AuE-C $\in$ (-2,6)
- AuI-C $\in$ (10,30).

I run the optimization task for 50 iterations and Figure 4.16 shows an improvement of the loss function value.

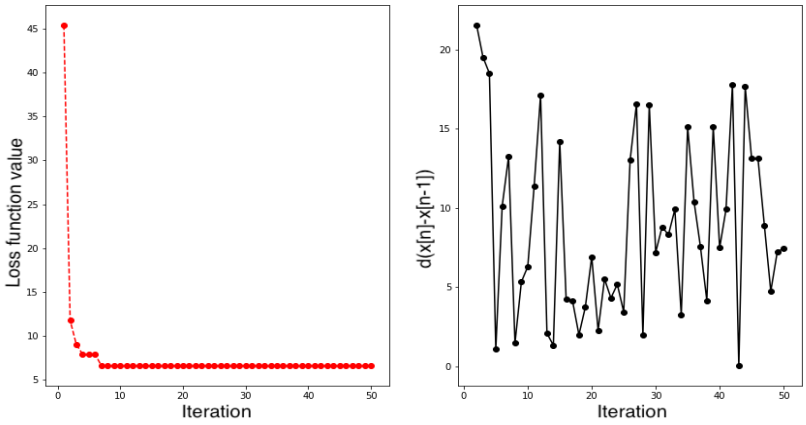
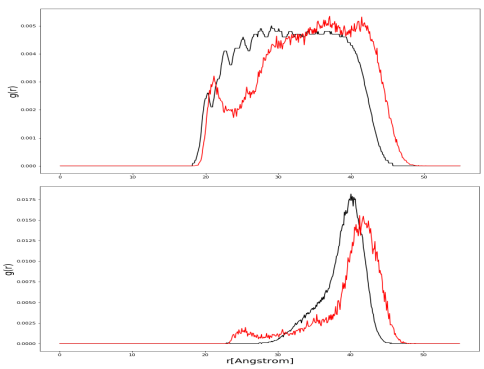
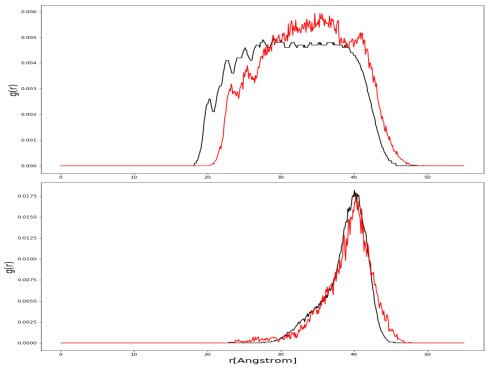


Figure 4.16: Convergence and Distance plot for the restricted Gold-Carbon Optimization

While the loss function and system are improved in the chain distribution, at a visual analysis of the RDFs for the iterations I notice that the system with the worst loss value for this optimization (figure 4.17(a)) is the overall the better system as the chains distribution of the system are less compressed and the chains starting point is correct.



(a) Worst loss value system; a_{AuE-C}=-2,a_{AuI-C}=6



(b) Best loss value system; a_{AuE-C}=-2,a_{AuI-C}=30

Figure 4.17: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the best and worst system in the optimization of the local gold parameters from AA and CG simulations. Color legend: AA, black; CG, red.

In contrast, the best loss value RDFs (figure 4.17(b)) while having a better ending

point and a better configuration of terminal beads, is not able to capture the starting point of the polymer chain distribution. Following this reason, I choose to continue the work with the system (Figure 4.17(a)) as I consider it the best one as it captures better the behaviour of the polymer chains.
The optimizations of gold-carbon parameters show how in this complex system a strong correlation is present, these interactions seem to affect not only the distance of the starting point in the polymer chains, but also the behaviour of the terminal beads (Figure 4.17(a)). Increasing the parameters lead to the presence of a valley near the first peak.

Bonds Angles Parameters Optimization

Since the new interaction parameters change the starting point in the polymer chains but also in the terminal group distribution I decide to run another optimization. To fully account for the polymer chain structure I also include the angle parameters between the chain beads (namely the spring constant and the equilibrium angle) in the search; since less stiff parameters could lead to smooth the valley after the peak of the chain distribution and smooth the increase near the gold core surface for the terminal group distribution (Figure 4.17(a)). This optimization adds to the 8 free bond parameters from the bond optimization, 10 other free parameters from the angle one; Because of this assumption, we obtained a space of 18 parameters, which is the largest dimensionality analyzed in this thesis (Table 4.5).

Type	**Free**	**Fixed**	**Constrained**
a_{ij}	0	36	0
K_i	5	0	0
θ_i	5	0	0
K_j	4	0	0
r_j	4	0	0

Table 4.5: Summary table of the bond and angle parameters optimization. As the interaction parameters are fixed only the chain bonds (K_j, r_j) and angles (K_i, θ_i) are optimized.

For this problem I choose the following conditions which span the common range of value for these parameters in DPD and they include both rigid and flexible cases:

- $K_j \in$ (0.2,1.0)
- $r_j \in$ (40,180)
- $K_i \in$ (0.2,10)

- $\theta_i \in (30,180)$.

The optimization task is run for 30 steps and I find that the loss function value for the system improved (Figure 4.18)

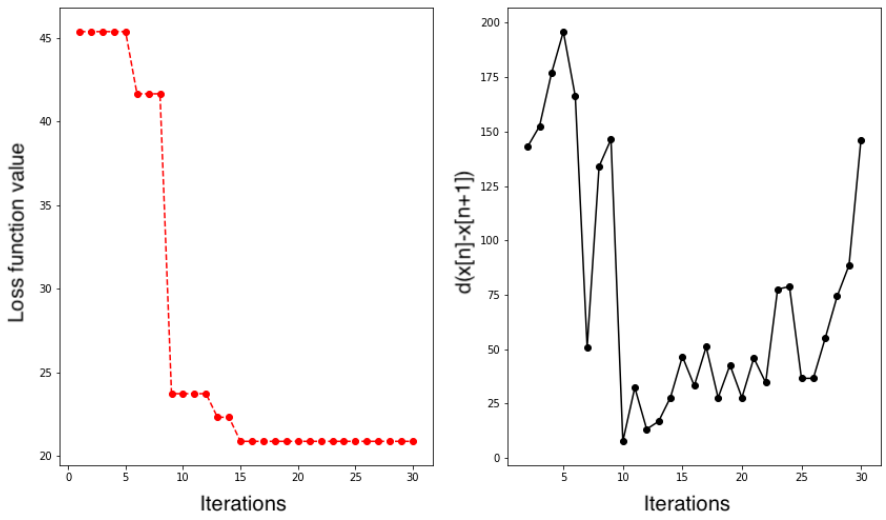
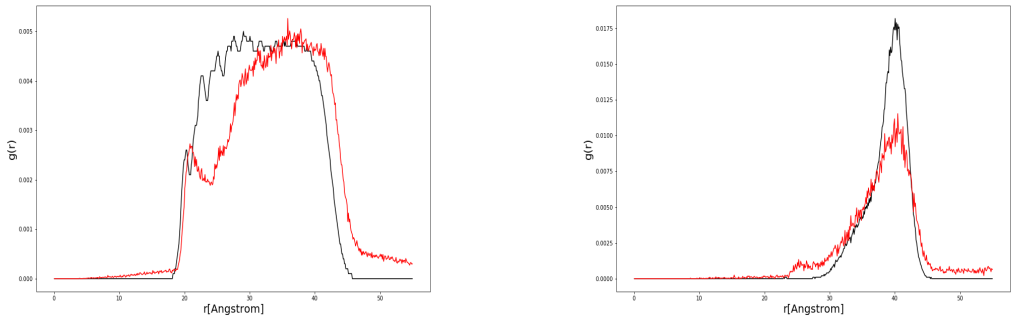


Figure 4.18: Convergence plot and Euclidean distance plot for the bond and angle parameters optimization.

The system from the best iteration in Figure 4.19 shows how the optimization of these parameters improved significantly the distribution of the polymer chains and the terminal group .While the RDF of the terminal group doesn't match entirely the AA reference, improving these parameters has led to the disappearance of the beads near the core surface in a better agreement with the AA simulation, in addition the correct distance of the peak seems to be matched. The chains distribution shows how the BO is able to correctly match the starting peak and to relax the chains in agreement with the reference, but it is not able to fully capture and match the complex behaviour of the polymer.



(a) Distribution of terminal group N16 **(b)** Distribution of terminal group N

Figure 4.19: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the best system obtained from the bond and angle stiffness and equilibrium parameters optimization of the local gold from AA and CG simulations. Color legend: AA, black; CG, red. Parameters: K_1^b=92,K_2^b=84.5,K_3^b= 60,K_4^b=67,r_1=0.2,r_2= 0.2,r_3=1.0,r_4 = 0.2,K^a = 10,K_2^a=4.7,K_3^a = 3.2,K_4^a= 10.0,K_5^a = 0.2,θ_1 = 89, θ_2=78,θ_3 =77,θ_4 = 72,θ_5=68.

4.2.2 Loss Function

In this section the issues and challenges connected to the L_2 loss function will be described and discussed.

Statistic uncertainty

While the L_2 loss function minima is mathematically defined as zero, the statistical nature of the RDF makes it practically impossible for this minima to be actually reachable in our case.
Fluctuations on the RDF are common as the bin values are averaged but more importantly, the degrees of freedom lost by mapping from AA to CG hampers retaining the "perfect" structural behaviour of the atomistic phenomena.
More troublesome it is the complexity and smoothness of our phase space and also how many local minima are present in our systems.
It can be speculated that, as the DPD is a soft potential, the phase space surface could have many plateau and local minima leading to similar incorrect physically configurations.

Sampling

The increased complexity added by the high dimensionality of the phase spaces faced in this work represent not only a challenge on the computational side, but also on the visualization side.
As the dimensionality of the interaction space was large, the risk of incorrect and not uniform sampling was always a concern. The common ways to address this challenge is by visualizing the phase space with different types of plots and to see how the sampling is proceding. Scatter plot matrix were tried to achieve a good visualization but lacking on simplicity therefore, for that reason a box plot was chosen to convey the informations visually.
Figure 4.20 shows that the interaction parameters space was sampled at least once for all parameters in the entire range, beside for the C-C interaction. Since it is not possible to extrapolate the sampling across all dimensions, this figure at least assure us that for most combinations of parameters, no zone were forbidden.

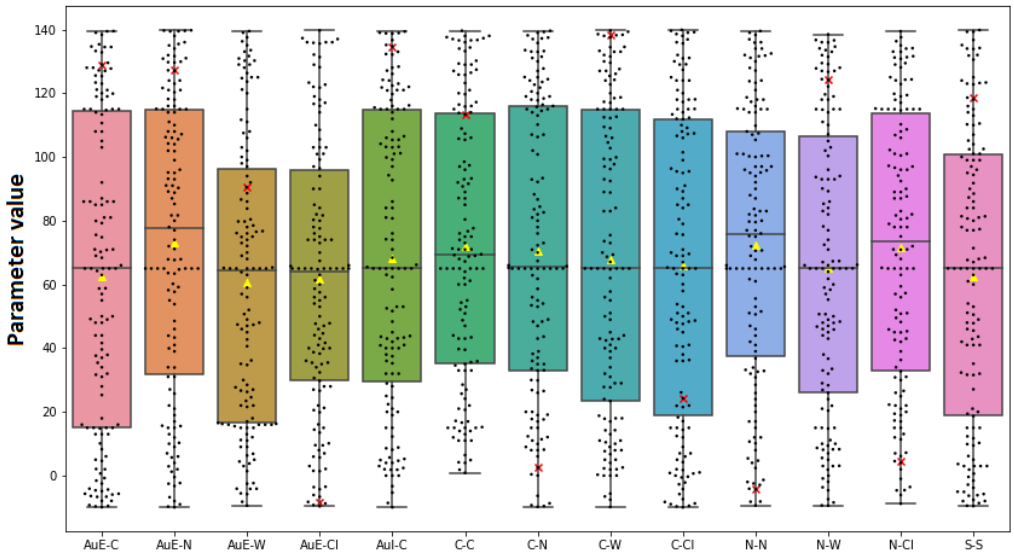
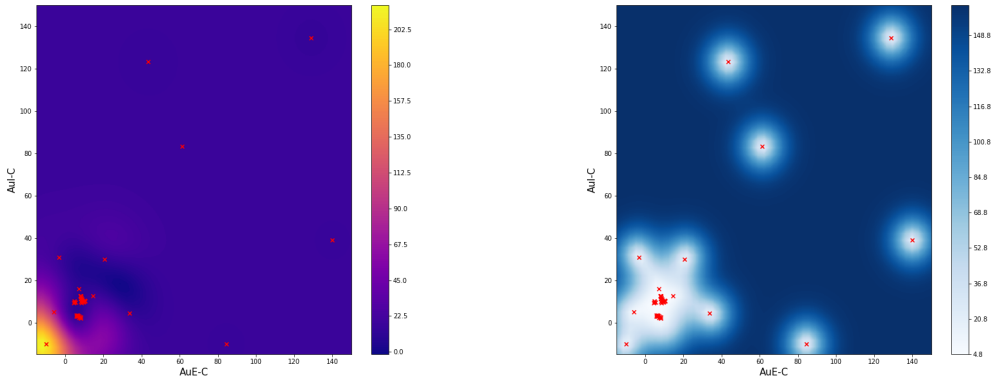


Figure 4.20: Interaction coefficients sampled by the BO framework in the 130 iterations performed in this work. Legend: mean, yellow triangle; best system interation parameters, red x.

The absence of sampling in the lower boundary of the C-C interaction is caused by unexpected behaviour of the system where unphysical behaviour of the nanoparticle leads to an impossibility to conclude the CG simulations.
A more detailed sampling analysis was done for the optimization of the gold-carbon interaction as the visualization of a lower dimensional problem is more trivial and

direct.

Gold-Carbon Loss function Since the gold-carbon optimization utilized two free parameters, it was possible to directly plot the loss function predicted surface by utilizing the GP package of the framework made for this work.
The Figure 4.21 shows how most of the phase space is represented by a plaetau and most of the sampling was done near the region where the optimizer found lower values of the loss function.

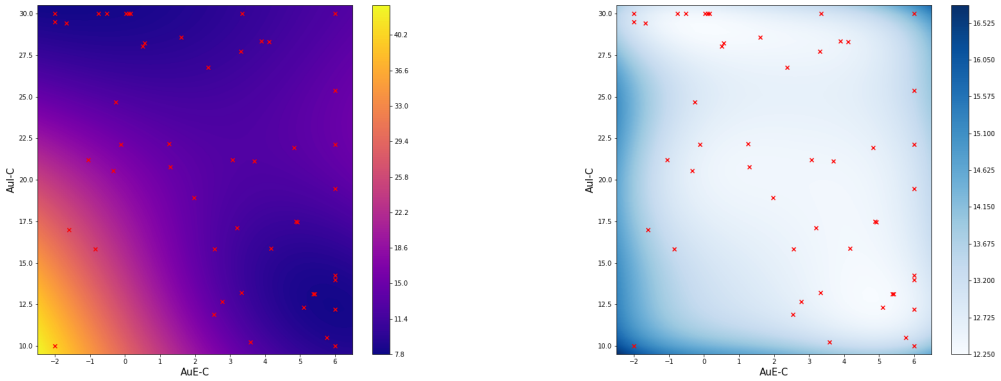


(a) Mean **(b)** Variance

Figure 4.21: Loss function surface plot from GP prediction for the global Au-C optimization task with the associated uncertainty.

Figure 4.22 shows us how the sampling of the local optimization don't cover entirely the strong attractive interactions. The strong attractive interactions that capture the beginning point of the polymer chain are unfavorable points for the loss function, that after making sure of the presence of the local maximum, it prefers not to test more in proximity of the configuration that is favorable for us; the Figure 4.22(a) also shows the presence of two local minimas at the opposite sides of the graph. The uncertainty associated with the loss function shows how in this case, the RBF kernel used by us could be unsuitable for this portion of space. The strong attractive interactions that capture the beginning point of the polymer chain are unfavorable points for the loss function, that after making sure of the presence of the local maximum, it prefers not to test more in proximity of the configuration that is favorable for us; the Figure 4.22(b) also shows the presence of two local minimas at the opposite sides of the graph. The uncertainty associated with the

loss function shows how in this case, the RBF kernel used by us could be unsuitable for this portion of space.



(a) Mean **(b)** Variance

Figure 4.22: Loss function surface plot from GP prediction for the local Au-C optimization task with the associated uncertainty.

Water RDF

Since during the first two optimization routines the BO was not able to capture the AA behaviour of the water, it was decided to test the behavior and response of the system on different sets of interaction parameters. To determine whether the system was reacting adequately to solvent interaction and to observe a change in the RDF of the water, it was decided to set the interaction parameter between the solvent (W) and the alkyl chain (C,L,N) to obtain opposite behaviors of the system. To obtain a hydrophobic behaviour the interaction parameters were chosen to be strongly repulsive (a_{W-C}=106.6,a_{W-L}=106.6,a_{W-N}=106.6), instead to capture the hydrohilic system these parameters were chosen as highly attractive (a_{W-C}=51.6,a_{W-L}=51.6,a_{W-N}=51.6). In one case at a high value to obtain repulsive interactions to lead to a hydrophobic behaviour and in the opposite case at a low value to have attractive interactions to obtain a hydrophilic behaviour. Both systems were simulated with the same routine explained in Section 3.1.4.
Figure 4.23 shows how both systems react to the change of parameters by demonstrating an influence of the solvent in both the distribution of the polymer chains and in the terminal group. The water distribution instead remains almost unchanged.

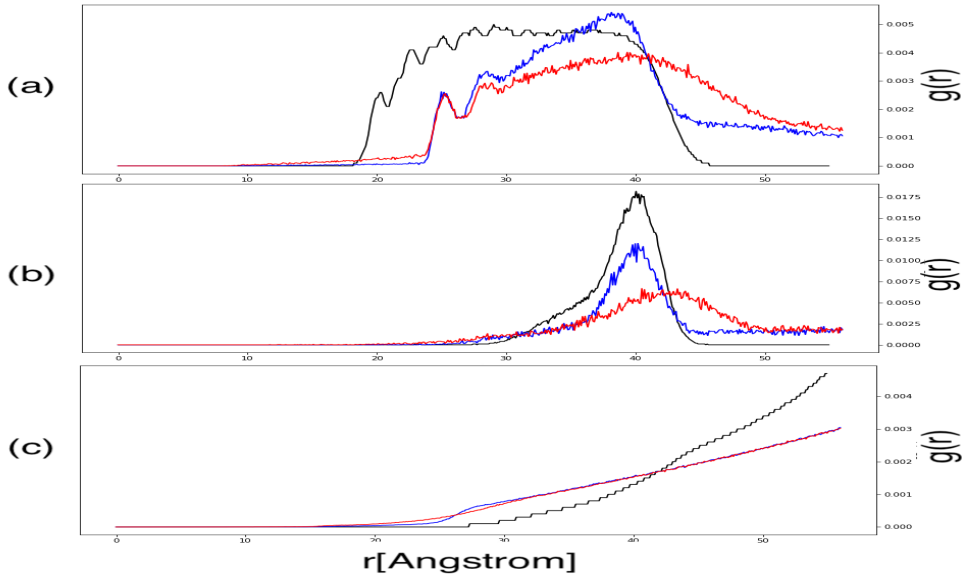


Figure 4.23: RDFs as a function of the distance r from the center of mass of the nanoparticle calculated for the the two configurations cases. Color legend: AA, black; CG hydrophilic, red; CG hydrophobic, blue. (a) N16 chains; (b) terminal N group; (c) water.

Although not achieving changes in the water distribution the two remaining RDFs demonstrate to be influenced by the solvent and displaying the correct attractive and repulsive behaviors. Since this behaviour could lead to higher values of loss function and also to wrong sampling of the search space it was decided to neglet the contribute to the loss function of the water RDF.
The causes of this behaviour were not further investigated and they are still an open question, even if from this section we have found that the inability to capture the water distribution is not due to the choice of the loss function but to the system itself, as the 5:1 mapping used for this model could be excessive. Even without the contribution of the water distribution in the loss function, the influence of the solvent on the nanoparticle is still captured in the remaining RDFs.

4.3 Results

As a final section, the results for every step of the evolution diagram in Figure 4.5 is be provided and discussed. As the RDFs for each of the following values were already showed in the previous section, only the parameters tables will be displayed.

Interaction Parameters Optimization

By looking at the value displayed in Table 4.8 and found from the interaction parameter optimization, we can observe that the physical repulsive interaction between the beads of the same species is properly captured as the C-C and S-S parameters are near the upper boundary. The same repulsive nature was not found instead on the interaction between two terminal beads even with a good agreement of the RDF.

Bead 1	Bead 2	Lower Bound	Upper Bound	Value
AuE	C	-10	140	128.9
AuE	N	-10	140	127.1
AuE	W	-10	140	90.5
AuE	Cl	-10	140	-8.2
AuI	C	-10	140	134.7
C	C	-10	140	113.2
C	N	-10	140	2.52
C	W	-10	140	138.1
C	Cl	-10	140	24.2
N	N	-10	140	-4.3
N	W	-10	140	124.4
N	Cl	-10	140	4.5
S	S	-10	140	118.6

Table 4.6: Summary table for the interactions a_{ij} of the best system found in the optimization of the interaction parameters.

Bond Optimization

By looking at the parameters in Table 4.9 and found by the bond optimization, the system is highly rigid.

Parameter	Lower Bound	Upper Bound	Value
K_1^b	40	180	95
K_2^b	40	180	124
K_3^b	40	180	125
K_4^b	40	180	125
r_1	0.2	1.0	0.2
r_2	0.2	1.0	0.22
r_3	0.2	1.0	0.85
r_4	0.2	1.0	0.83

Table 4.7: Summary table for the stiffness K_j and the equilibratio length r_j of the best system found in the optimization of the bonds parameters.

Gold-Carbon Optimization

Bead 1	Bead 2	Lower Bound	Upper Bound	Value
AuE	C	-2	6	-2
AuI	C	10	30	10

Table 4.8: Summary table for the interactions a_{AuE-C}, a_{AuI-C} of the best system found in the optimization of the gold-carbon interaction parameters.

Bonds and Angles Optimization

The RDFs show that the presence of the first peak is actually ruled by the Au-C interaction. As other RDFs displayed an absence of the peak, we supposed that there is not only an important relation between the Au and C parameters but also from the difference of values between them when the external shell become highly attractive.

Parameter	Lower Bound	Upper Bound	Value
K_1^b	40	180	92
K_2^b	40	180	84.5
K_3^b	40	180	60
K_4^b	40	180	67
r_1	0.2	1.0	0.2
r_2	0.2	1.0	0.2
r_3	0.2	1.0	1.0
r_4	0.2	1.0	0.2
K_1^a	0.2	10	10
K_2^a	0.2	180	4.7
K_3^a	0.2	10	3.2
K_4^a	0.2	10	10.0
K_5^a	0.2	10	0.2
θ_1	30	180	89
θ_2	30	180	78
θ_3	30	180	77
θ_4	30	180	72
θ_5	30	180	68

Table 4.9: Summary table for the bond stiffness and equilibrium length (K^b,r_j), and angle stiffness and equilibrium angle (K^a,θ_j) parameters of the best system found in the optimization of the bond and angle optimization.

4.3.1 Validation

To validate the previous supposition of a loss function with multiple local minima, I run another optimization of the parameters interaction for 50 iterations under the constrains on the interaction space of Section 3.2.2.

In this case I didn't use the sampling method employed by McDonagh and in the first optimization, instead I chosen to use the L-BFGS-B numerical solver. The figure 4.24 shows how on the same problem of optimizing 13 a_{ij} of the DPD potential, the numerical solver provides a better convergence and values of loss function even with a small database D of only random points.

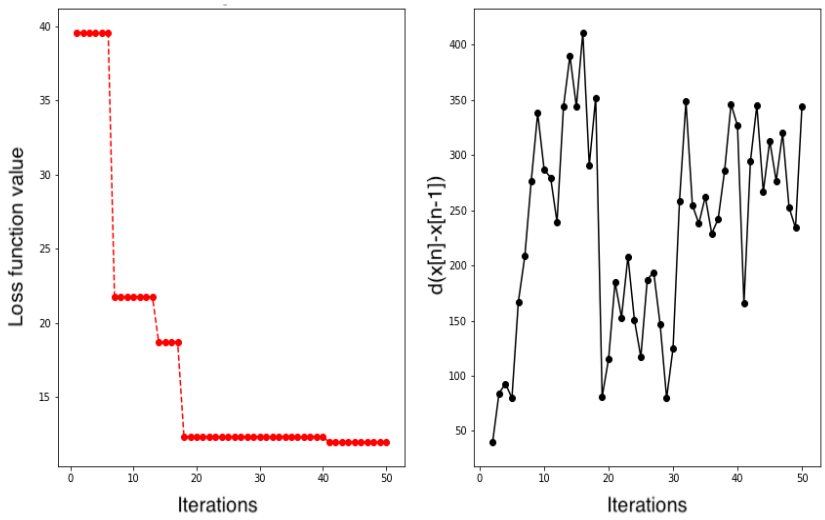
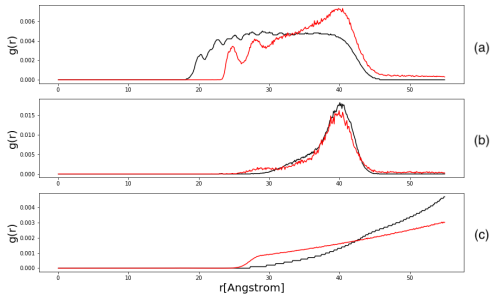
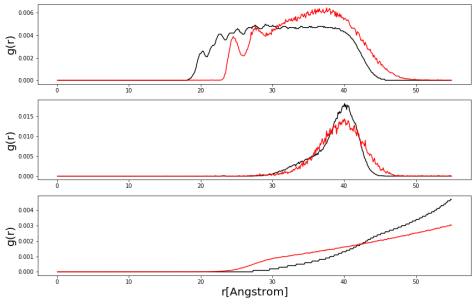


Figure 4.24: Convergence plot and Euclidean distance plot for the validation optimization.

Figure 4.25 shows how the best system of this optimization run compare with the result of the first parameters optimization; although the interaction parameters found are wildly different between the two systems, the chain distribution and distribution of the terminal groups display the same incorrect physical behaviour of the previous systems.
As previously supposed, the presence of multiple similar configurations and local minima could be a consequence of the soft interaction of the DPD potential as even with different set of interaction parameters the configurations for this system assumes rather similar distributions and show similar physical behaviours.



(a) Random sampling best system



(b) L-BFGS-B solver best system

Figure 4.25: RDFs of the best system from the optimization of the interaction parameters using random sampling compared to the ones from the L-BFGS-B solver run. Colour legend: AA, black; CG, red.

Table 4.10 shows the interaction parameters of both the systems found from the first optimization and from this one.

Bead 1	**Bead 2**	**Random Sampling**	**L-BFGS-B**
AuE	C	128.9	45.0
AuE	N	127.1	38.0
AuE	W	90.5	54.0
AuE	Cl	-8.2	107.0
AuI	C	134.7	91.0
C	C	113.2	74.0
C	N	2.5	90.0
C	W	138.1	77.0
C	Cl	24.2	117.0
N	N	-4.3	66.0
N	W	124.4	100.0
N	Cl	4.5	38.0
S	S	118.6	88.0

Table 4.10: Summary table for the interactions a_{ij} of the two best systems found from the optimization with the random sampling and the optimization with the L-BFGS-B solver.

Chapter 5

Conclusions

In this thesis work I investigated the possibility of extending the use of machine learning techniques to the derivation of potentials for coarse-graining simulations. I focused on a system of polymer based nanoparticles as they are materials with considerable application in the field of catalysis, nanocomposites fabrication, sensing and delivery with their abilities of self-assembling in complex three-dimensional structures. I selected a promising machine learning technique, bayesian optimization, and through a bottom-up strategy I used it to optimize the parameters of one of the most common force field (dissipative particle dynamics) for these system by developing a novel workflow.
The results obtained show how the approach I used is able to partially capture the complex behavior of these systems and it is able to parameterize the force field through fast and interchangeable optimization routines. Although this technique is not able to guarantee a perfect structural matching, it is able to capture the distances of the peaks of the various distributions from the all-atoms simulation. Through this approach I also identified the correlation and influence that some parameters have on this system.
While it is possible to use bayesian optimization for complex systems, some critical points need to be addressed. The chosen loss function seems to be too naive for this type of problem and led to the issue of finding unfavorable physical systems with a low loss function value. Furthermore the approach was not able to capture the behaviour of the water or provided us with information related to this limitation. Therefore, further research is needed to determine an improved loss function that penalizes more the unfavorable physical systems, to investigate the scalability of this approach by cross correlation, and to investigate the behaviour of the coarse-graining level on the system.

Appendix A

Other Details

Code and Framework

The framework can be found on the github page:
https://github.com/FNTwin/GPGO

Bibliography

[1] Florian Häse, Ignacio Fdez Galván, Alán Aspuru-Guzik, Roland Lindh, and Morgane Vacher. *Chemical Science* 10 (2019), pp. 2298–2307.

[2] Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A. Persson, Gerbrand Ceder, and Anubhav Jain. *Nature* 571 (2019), pp. 95–98.

[3] Andrew J. Chancellor, Bryan T. Seymour, and Bin Zhao. *Analytical Chemistry* 91 (2019), pp. 6391–6402.

[4] Shengchao Chai, Xiao Cao, Fengrui Xu, Liang Zhai, Hu-Jun Qian, Quan Chen, Lixin Wu, and Haolong Li. *ACS nano* 13.6 (2019), pp. 7135–7145.

[5] Radislav A. Potyrailo. *Chemical Society Reviews* 46 (2017), pp. 5311–5346.

[6] Marek Grzelczak, Luis M. Liz-Marzán, and Rafal Klajn. *Chemical Society Reviews* 48 (2019), pp. 1342–1361.

[7] Paola Posocco, Cristina Gentilini, Silvia Bidoggia, Alice Pace, Paola Franchi, Marco Lucarini, Maurizio Fermeglia, Sabrina Pricl, and Lucia Pasquato. *ACS Nano* 6 (2012), pp. 7243–7253.

[8] Maria Şologan, Domenico Marson, Stefano Polizzi, Paolo Pengo, Silvia Boccardo, Sabrina Pricl, Paola Posocco, and Lucia Pasquato. *ACS Nano* 10 (2016), pp. 9316–9325.

[9] Domenico Marson et al. *Small* 15 (2019), p. 1900323.

[10] Jing Huang, Sarah Rauscher, Grzegorz Nawrocki, Ting Ran, Michael Feig, Bert L. de Groot, Helmut Grubmüller, and Alexander D. MacKerell. *Nature methods* 14 (2017), pp. 71–73.

[11] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff. *Journal of the American Chemical Society* 114 (1992), pp. 10024–10035.

[12] Mary J. Van Vleet, Alston J. Misquitta, and J. R. Schmidt. *Journal of Chemical Theory and Computation* 14 (2018), pp. 739–758.

[13] Benedict J. Leimkuhler, Sebastian Reich, and Robert D. Skeel. «Integration Methods for Molecular Dynamics». Springer, 1996, pp. 161–185.

[14] Jundong Chen. *IOP Conference Series: Earth and Environmental Science* 128 (2018), p. 012110.

[15] Siewert J. Marrink, H. Jelger Risselada, Serge Yefimov, D. Peter Tieleman, and Alex H. de Vries. *The Journal of Physical Chemistry B* 111 (2007), pp. 7812–7824.

[16] Gary S. Ayton and Gregory A. Voth. *The journal of physical chemistry. B* 113 (2009), pp. 4413–4424.

[17] Sergei Izvekov and Gregory A. Voth. *The Journal of Physical Chemistry. B* 109 (2005), pp. 2469–2473.

[18] Florian Müller-Plathe. *ChemPhysChem* 3 (2002), pp. 754–769.

[19] Rastko Sknepnek, Joshua A. Anderson, Monica H. Lamm, Jörg Schmalian, and Alex Travesset. *ACS Nano* 2 (2008), pp. 1259–1265.

[20] Cangyi Chen, Tiancai Zhang, Lei Zhu, Bin Zhao, Ping Tang, and Feng Qiu. *ACS Macro Letters* 5 (2016), pp. 718–723.

[21] P. Español and P. Warren. *Europhysics Letters (EPL)* 30 (1995), pp. 191–196.

[22] Robert D. Groot and Patrick B. Warren. *The Journal of Chemical Physics* 107 (1997), pp. 4423–4435.

[23] Zhen Cao and Gregory A. Voth. *The Journal of Chemical Physics* 143 (2015), p. 243116.

[24] Aram Davtyan, Gregory A. Voth, and Hans C. Andersen. *The Journal of Chemical Physics* 145 (2016), p. 224107.

[25] Frank Noé, Alexandre Tkatchenko, Klaus-Robert Müller, and Cecilia Clementi. *Annual Review of Physical Chemistry* 71 (2020), pp. 361–390.

[26] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[28] Jörg Behler. *The Journal of Chemical Physics* 134 (2011), p. 074106.

[29] Albert P. Bartók, Risi Kondor, and Gábor Csányi. *Physical Review B* 87 (2013), p. 184115.

[30] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. *Proceedings of the IEEE* 104 (2016), pp. 148–175.

[31] J. Mockus. en. 1982, pp. 473–481.

[32] Peter I. Frazier. *arXiv:1807.02811 [cs, math, stat]* (2018).

[33] Donald R. Jones, Matthias Schonlau, and William J. Welch. *Journal of Global Optimization* 13 (1998), pp. 455–492.

[34] Randy Jalem, Kenta Kanamori, Ichiro Takeuchi, Masanobu Nakayama, Hisatsugu Yamasaki, and Toshiya Saito. *Scientific Reports* 8 (2018), p. 5845.

[35] J. S. Smith, O. Isayev, and A. E. Roitberg. *Chemical Science* 8 (2017), pp. 3192–3203.

[36] Jacob R. Boes, Mitchell C. Groenenboom, John A. Keith, and John R. Kitchin. *International Journal of Quantum Chemistry* 116 (2016), pp. 979–987.

[37] Albert P. Bartók, Mike C. Payne, Risi Kondor, and Gábor Csányi. *Physical Review Letters* 104 (2010), p. 136403.

[38] James McDonagh, Ardita Shkurti, David Bray, Richard Anderson, and Edward Pyzer-Knapp. *Journal of Chemical Information and Modeling* 2019 (2019).

[39] Tarak K Patra, Troy D. Loeffler, Henry Chan, Mathew J. Cherukara, Badri Narayanan, and Subramanian K. R. S. Sankaranarayanan. *Applied Physics Letters* 115 (2019), p. 193101.

[40] Claudio Zeni, Kevin Rossi, Aldo Glielmo, and Francesca Baletto. *Advances in Physics: X* 4 (2019), p. 1654919.

[41] Karteek K. Bejagam, Samrendra Singh, Yaxin An, and Sanket A. Deshmukh. *The Journal of Physical Chemistry Letters* 9 (2018), pp. 4667–4672.

[42] Alain Dequidt and Jose G. Solano Canchaya. *The Journal of Chemical Physics* 143 (2015), p. 084122.

[43] Junmei Wang, Romain M. Wolf, James W. Caldwell, Peter A. Kollman, and David A. Case. *Journal of Computational Chemistry* 25 (2004), pp. 1157–1174.

[44] Junmei Wang, Wei Wang, Peter A. Kollman, and David A. Case. *Journal of Molecular Graphics & Modelling* 25 (2006), pp. 247–260.

[45] Enguerran Vanquelef, Sabrina Simon, Gaelle Marquant, Elodie Garcia, Geoffroy Klimerak, Jean Charles Delepine, Piotr Cieplak, and François-Yves Dupradeau. *Nucleic Acids Research* 39 (2011), W511–517.

[46] Hendrik Heinz, Tzu-Jen Lin, Ratan Kishore Mishra, and Fateme S. Emami. *Langmuir* 29 (2013), pp. 1754–1765.

[47] Alex K. Chew and Reid C. Van Lehn. *The Journal of Physical Chemistry C* 122 (2018), pp. 26288–26297.

[48] Romelia Salomon-Ferrer, David A. Case, and Ross C. Walker. *WIREs Computational Molecular Science* 3 (2013), pp. 198–210.

[49] Romelia Salomon-Ferrer, Andreas W. Götz, Duncan Poole, Scott Le Grand, and Ross C. Walker. *Journal of Chemical Theory and Computation* 9 (2013), pp. 3878–3888.

[50] L. Martínez, R. Andrade, E. G. Birgin, and J. M. Martínez. *Journal of Computational Chemistry* 30 (2009), pp. 2157–2164.

[51] Vincenzo Barone, Alessandro Bencini, Maurizio Cossi, Andrea Di Matteo, Maurizio Mattesini, and Federico Totti. *Journal of the American Chemical Society* 120 (1998), pp. 7069–7078.

[52] Steve Plimpton. *Journal of Computational Physics* 117 (1995), pp. 1–19.

[53] Daniel R. Roe and Thomas E. Cheatham. *Journal of Chemical Theory and Computation* 9 (2013), pp. 3084–3095.

[54] William Humphrey, Andrew Dalke, and Klaus Schulten. *Journal of Molecular Graphics* 14 (1996), pp. 33–38.

[55] Yanming Wang, Tian Xie, Arthur France-Lanord, Arthur Berkley, Jeremiah A. Johnson, Yang Shao-Horn, and Jeffrey C. Grossman. *Chemistry of Materials* 32 (2020), pp. 4144–4151.

[56] L. Bittner. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 42.7-8 (1962), pp. 364–365.

[57] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. *Computing in Science & Engineering* 13 (2011), p. 22.

[58] J. D. Hunter. *Computing in Science & Engineering* 9 (2007), pp. 90–95.

[59] Pauli Virtanen et al. *Nature Methods* 17 (2020), pp. 261–272.

[60] Umberto Noè and Dirk Husmeier. *On a New Improvement-Based Acquisition Function for Bayesian Optimization*. 2018.