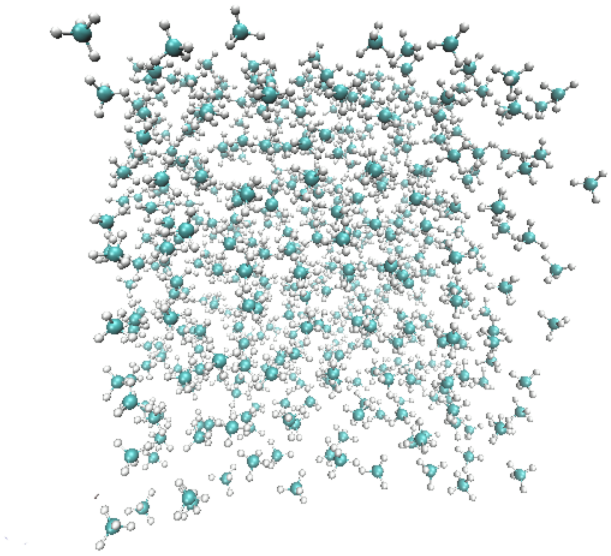
PARAMETERIZING MOLECULAR MODELS VIA GAUSSIAN APPROXIMATION METHODS

Vasiliki Kyrieri



MASTER THESIS

DEPARTMENT OF MATHEMATICS AND APPLIED MATHEMATICS

UNIVERSITY OF CRETE

Supervisors:

Prof. Vagelis Harmandaris, Department of Mathematics and Applied Mathematics, UoC

Dr. Evangelia Kalligiannaki, Institute of Applied and Computational Mathematics, FORTH

Members of committee:

Prof. Vagelis Harmandaris, Department of Mathematics and Applied Mathematics, UoC

Prof. George N. Makrakis, Department of Mathematics and Applied Mathematics, UoC

Dr. Yannis Pantazis, Institute of Applied and Computational Mathematics, FORTH

Abstract

In this thesis, we study Molecular Dynamics simulations, a scientific field that focuses on the physical motion of atoms on a very small order of magnitude. Today, many chemical and biological processes, such as protein interactions, can be modeled through MD simulations with aim of analyzing their properties. The challenging part of studying MD simulations of complex systems refers to an as-accurate-as-possible prediction of the structure-property relationship at the microscopic level and the expensive calculations of the dynamic quantities due to the wide range of length and time scales. By decreasing the number of degrees of freedom, the new system can be used with fewer variables. This method, known as Coarse-Graining, maps the atomistic particles into mesoscopic particles such as "superatoms".

There exists a variety of methods to obtain the total force of the mesoscopic system, either parametric or non-parametric models. In this thesis, the Gaussian process regression model, a flexible non-parametric family of models capable of approximating functions using relatively small data sets, is applied to a simple system, a methane system, and its results are compared with two more straightforward approximations. One with a parametric pair potential, the Lennard-Jones potential, and another with the Linear B-splines representation. We learn the approximate force fields, with the Force Matching criterion of loss, which minimizes the average distance between the atomistic forces and the approximate CG forces.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Vangelis Harmandaris, for showing me trust, being so willing to help and also to encourage me. He inspired me the love of the field of Machine Learning.

I would also like to say a special thanks to Dr. Evangelia Kalligiannaki for her continuing support, her guidance throughout this project and the time she has devoted into my work.

Furthermore, I would like to thank Antonis Hatzirakis for sharing his knowledge and for helping my endeavor to complete my thesis.

Finally, I owe my deepest gratitude to my family who supported me and believed in me, and to my friends who helped me through all these years to communicate my feelings and my dreams for a brighter future.

Contents

List of Figures

List of Tables

Notation

Atomistic Description

$\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N)$ Atomistic particles' position

N # of microscopic particles

$\mathbf{q} \in \mathbb{R}^{3N}, \mathbf{q}_i \in \mathbb{R}^3$

Coarse-grained description

$\mathcal{Q} = (\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_M)$ Coarse-grained particles' position

M # of CG-particles

$\mathcal{Q} \in \mathbb{R}^{3M}, \mathbf{Q}_i \in \mathbb{R}^3$

$\Pi(\mathbf{q}) = \mathcal{Q}, \Pi : \mathbb{R}^{3N} \to \mathbb{R}^{3M}$ Mapping function

Minimization Parameters

$\mathbf{a} = (\acute{\mathbf{a}}, \hat{\mathbf{a}}, \mathbf{a}^*)$ Minimization parameters

$\acute{\mathbf{a}} = (\acute{a}_1, \acute{a}_2, \cdots)$ parameters for the W_{mono}

$\hat{\mathbf{a}} = (\hat{a}_1, \cdots, \hat{a}_{n_p})$ parameters for the W_{dimer}

n_p # of minimization parameters (dimer)

$\mathbf{a}^* = (a_1^*, \cdots, a_{\mathbf{n_p}^*}^*)$ Parameters for the W_{trimer}

$\mathbf{n_p}^*$ # of minimization parameters (trimer)

$\acute{a}_i, \hat{a}_i, a_i^* \in \mathbb{R}$

$\mathbf{x} = (x_1, x_2, ..., x_{n_p})$ Basis points

n_s # of samples(configurations)

1 Introduction

Complex molecular materials, from colloids and hybrid nanocomposites to biomolecular systems, are used in a variety of biological and chemical applications. The theoretical and computational modeling of such systems is a very intense research area. The main challenge in these models is finding a direct quantitative relationship between microscopic and mesoscopic levels. Our goal is to approximate as accurately as possible the chemical structure and the measurable quantities over a wide range of length and time scales.

At the most detailed level, the atomistic level, the scientific region that describes a molecular system and atoms' movement, is known as Molecular Dynamics (MD), ref. [11, 34]. In a chemical system, the Potential of Mean Force (PMF) is the most significant quantity to estimate, and it needs complicated and costly computations to approximate it. MD simulations numerically integrate Newton's equations of motion, providing valuable information about the physical properties of molecular systems.

A way to decrease the computational cost of simulations, and so to explore a larger range of space and time regions, is by applying Coarse-Graining (CG) methods. The idea of CG relies on reducing the dimensions of the molecular system by averaging out the details of the atomistic system at the molecular level by representing groups of atoms with a single CG particle. There are many ways to define CG "mappings". For example, define a CG particle as part of a molecule e.g., a number of monomers or long molecules represented as one CG particle. The choice of the degree of coarse-graining is based on the complexity of the current system. In scientific computing and applied statistics, the subject of CG methods is a very active research topic. Ref. [2, 3, 1, 32, 35, 36]

The systematic CG modeling for a given system, based on detailed microscopic data, is shortly described through the following stages:
(a) execution of microscopic simulations on small model systems,
(b) selection of a CG map (transformation from the atomistic to the CG description),
(c) development of a CG effective interaction potential,
(d) execution of the CG MD simulations.

The main issue we need to handle concerns the choice of technique to describe all CG particle forces. There are several methods to approximate a parametric or non-parametric model under different minimization principles. In this thesis, our goal is to solve the Force Matching (FM) problem that refers to the minimization problem between two quantities, the target forces and the approximated CG forces.

A promising new tool to describe the energy of complex systems is via the adoption of Machine Learning tools, such as artificial Neural Networks [10, 3, 32, 33], Gaussian kernels [3, 19, 20, 7, 8], and Gaussian processes. Here, we focus on Gaussian Process Regression (GPR) models, a nonlinear, non-parametric Bayesian probability method, able to approximate an unknown function $f(x)$ as a sum of basis functions multiplied by the minimization parameters and based on a common kernel function. The advantage of GPR relies on its representation flexibility and inherent uncertainty measures over predictions.

The results of this method are compared to two other methods, the Linear Basis method and the Lennard-Jones (LJ) method. In all three methods, a linear problem arises. We solve the minimization problem by applying the normal equations to the final linear system.

This thesis consists of four main chapters. In chapter 2, we introduce Molecular Dy-

namics. This groundbreaking method allows us to study classical many-body systems, and we affiliate it with the Coarse-Graining method to reduce the computational cost. The next chapter presents the field of Machine Learning and the theory of Gaussian Process Regression, which is capable of approximating functions using observed data. In order to understand how the Gaussian Process Regression is connected with the Kernel Ridge Regression that we applied in our methane system, we present a benchmark toy problem. The next step is to define the Force Matching problem by applying the above techniques. In chapter 4, we construct a cost (loss) function that corresponds to the Force Matching problem and is a minimization expression between the atomistic target forces and the approximate Coarse-grained particles' forces of a molecular system. In the last chapter 5, we describe the methane system and according to this system, we learn the Coarse-Grained model with the Gaussian Process Regression method, and present the results of our study.

2 Multi-scale Dynamics

2.1 Introduction

The studying of physical phenomena on a molecular level is necessary for the investigation of scientific challenges and technological applications. From a theoretical point of view, the main question in complex molecular systems is the linking of the chemical structure of a microscopic (atomistic) level and that of a mesoscopic (molecular) level over a wide range of time and length scales. The solution to such challenges can contribute to a wide range of potential applications in nanotechnology, the pharmaceutical industry biotechnology, cosmetics, etc., and generally in the designing of new materials, [1, 4].

In both microscopic and mesoscopic systems, we can study the behavior of the constituent species of a system. However, the main difference is that the former refers to atoms and the mesoscopic to molecules. Methods that study such systems are called "molecular dynamics methods" and are represented by Monte Carlo (MC) and molecular dynamics (MD) methods.

MD is a deterministic simulation method responsible for the interaction and motion of particles in a many-body system. These microscopic particles follow Newton's law and have defined trajectories and material structures. In contrast, MC, known as random sampling or statistical test method, is a stochastic simulation method used to estimate the possible outcomes of an uncertain event and is consistent in approximating the actual process of the physics. MC methods show a strong ability to analyze thermodynamic equilibrium but are unsuitable for investigating dynamical phenomena. Thus, the advantage of MD is its ability to study a system in both thermodynamic equilibrium and non-equilibrium states. For the above reasons, we will examine the MD technique in the next section.

2.2 Molecular Dynamics

As we mentioned, MD simulation is a technique for studying the movement of atoms in a classical many-body system. The word "classical" means that the particle's motion obeys the laws of classical mechanics, Newton's law. It is necessary to know the trajectory (position and velocity) along with the energy for all particles and the system's density in each step. Compared to real experiments, the advantage of MD lies in its ability to repeat, stop or restart the process with the required initial conditions at a much lower cost. Additionally, mock experiments are more beneficial if they involve dangerous substances than real ones where there is a potential for physical harm.

MD simulations are approximations of the actual experiments at the level of atomic or molecular detail. First, we choose a data set of particles with initial positions and velocities and solve Newton's equations of motion until the system reaches equilibrium. After that, we do the actual measurement. During the process, we aim to avoid mistakes such as not having enough samples or preparing an incorrect data set.

In simulated systems, the MD method studies a model consisting of N particles that interact with each other based on the present forces. The problem of predicting the trajectory of each interacting atom and/or molecule is known as the N-body problem, where N is the total number of particles in the simulated system. To calculate the desired quantities of each particle, i.e. its position and velocity, we numerically solve the classical equations of motion

applied to each particle, considering all the interactions between them.

2.2.1 Equations of Motion

In order to solve the N-body problem, we numerically solve the equations of motion. Consider a system of N interacting atoms that obey Newton's second law

$$\mathbf{F} = \mathbf{M}\frac{d^2\mathbf{q}}{dt^2} \tag{1}$$

where $\mathbf{M}$ is the mass matrix of particles given by $\mathbf{M} = diag[m_1, \cdots, m_N]$ and $\mathbf{F} \in \mathbb{R}^{3N}$ is the total force applied to each atom's position $\mathbf{q}(t) \in \mathbb{R}^{3N}$ at time t. Thus the N-body equations of motion arise,

$$\mathbf{M}\frac{d^2\mathbf{q}}{dt^2} = -\frac{\partial U(\mathbf{q})}{\partial \mathbf{q}} \tag{2}$$

where U is the potential energy of the system.

In a molecular system, the trajectory is given by the generalized positions $\mathbf{q}(t)$ and momenta $\mathbf{p}(t) \in \mathbb{R}^{3N}$ of the particles and are used in the calculation of the Hamiltonian equations of motion

$$\dot{\mathbf{q}} = H_{\mathbf{p}}, \quad \dot{\mathbf{p}} = -H_{\mathbf{q}} \tag{3}$$

$$\frac{dq_i}{dt} = \frac{\partial H(\mathbf{p},\mathbf{q})}{\partial p_i}, \quad \frac{dp_i}{dt} = -\frac{\partial H(\mathbf{p},\mathbf{q})}{\partial q_i}, \quad i = 1, ...N \tag{4}$$

where $H = H(\mathbf{q},\mathbf{p}) = K(\mathbf{p}) + U(\mathbf{q})$ is the Hamiltonian where the kinetic energy K depends on the momentum $\mathbf{p}$ and the potential energy U on the position $\mathbf{q}$ Ref. [11]. The kinetic energy is given by the form $K(\mathbf{p}) = \frac{1}{2}\mathbf{p}\mathbf{M}^{-1}\mathbf{p}$ and the momentum vector is defined as $\mathbf{p} = \mathbf{M}\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t}$.

Thus we have the following equations to solve,

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} = \frac{\partial K}{\partial \mathbf{p}} = \frac{\mathbf{p}}{m} \tag{5}$$

$$\dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}} = -\frac{\partial U}{\partial \mathbf{q}}. \tag{6}$$

An important property of the equations of motion is that the Hamiltonian quantity H is conservative, meaning it does not depend on time $\dot{H} = \frac{\mathrm{d}H}{\mathrm{d}t} = 0$. Another property is that Hamilton's equations are time-reversible, which means that by changing the signs of the velocities, the molecules change their trajectory and move backward. The above properties must also be taken into account by computer-generated molecular systems.

For an N-body system, where N can be a large number, the Hamiltonian equations of motion are $6N$ differential equations. To study the system evolution, we need to integrate the equations, through a computational procedure. The computational cost is huge, thus a solution is to reduce the number of atoms $M < N$ and, by extension, reduce the computational cost. A useful theory to achieve our goal is statistical mechanics, where the microscopic properties of atomistic particles can be related to the macroscopic bulk properties of materials.

2.2.2 Numerical Solutions of the Equations of Motion

There are many methods for solving the classical equations of motion, especially Newton's equations, so that information about the positions and velocities of the atoms can be obtained. We will present the two most common methods Ref. [9, 11].

Verlet Algorithm

The simplest and usually preferred method to integrate Newton's equations of motion is the Verlet Algorithm. First, we apply a Taylor Expansion of the coordinate of a particle, around time t,

$$\mathbf{q}(t+\Delta t) = \mathbf{q}(t) + \dot{\mathbf{q}}(t)\Delta t + \frac{\ddot{\mathbf{q}}(t)}{2}\Delta t^2 + \frac{\dddot{\mathbf{q}}(t)}{3!}\Delta t^3 + O(\Delta t^4) \tag{7}$$

and

$$\mathbf{q}(t-\Delta t) = \mathbf{q}(t) - \dot{\mathbf{q}}(t)\Delta t + \frac{\ddot{\mathbf{q}}(t)}{2}\Delta t^2 - \frac{\dddot{\mathbf{q}}(t)}{3!}\Delta t^3 + O(\Delta t^4) \tag{8}$$

where $\dot{\mathbf{q}}(t) = \mathbf{v}(t), \mathbf{v}(t) \in \mathbb{R}^{3N}$ is the particles' velocity and $\ddot{\mathbf{q}}(t) = \mathbf{a}(t), \mathbf{a}(t) \in \mathbb{R}^{3N}$ acceleration. After summing the above equations we have the form

$$\mathbf{q}(t+\Delta t) + \mathbf{q}(t-\Delta t) = 2\mathbf{q}(t) + \mathbf{a}(t)\Delta t^2 + O(\Delta t^4). \tag{9}$$

When evaluating the new positions, the error is of order Δt^4, where Δt is the time step.

As we noticed in eq. 9 we do not use the velocities, so we can derive them by subtracting the eq.8 from eq.7.

$$\begin{aligned} \mathbf{q}(t+\Delta t) - \mathbf{q}(t-\Delta t) &= 2\mathbf{v}(t)\Delta t + O(\Delta t^3) \\ \mathbf{v}(t) &= \frac{\mathbf{q}(t+\Delta t) - \mathbf{q}(t-\Delta t)}{2\Delta t} + O(\Delta t^2) \end{aligned} \tag{10}$$

Velocity-Verlet Algorithm

Here we will present an alternative to the Verlet algorithm, a scheme that uses positions and velocities computed at equal times. For the particles' positions, we apply the usual Taylor expansion

$$\mathbf{q}(t+\Delta t) = \mathbf{q}(t) + \dot{\mathbf{q}}(t)\Delta t + \frac{\ddot{\mathbf{q}}(t)}{2}\Delta t^2, \tag{11}$$

whereas velocities are calculated through

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{dt}{2}[\ddot{\mathbf{q}}(t) + \ddot{\mathbf{q}}(t+\Delta t)]. \tag{12}$$

2.2.3 MD Algorithm

We present a flow diagram of the MD algorithm. The procedure we follow for constructing an MD algorithm has several steps. The first step of the molecular model construction is to initialize each particle's position and velocities. After estimating the total force (intermolecular and intramolecular) on each particle, we proceed to the integration of the motion equations. We repeat N_k times the integration of the equations until the system reaches equilibrium. Also, it is needed to store the accurate positions, velocities, and forces at every step. As we can see in figure 1, the MD algorithm is terminated by calculating the averages of the measured quantities and the desired properties.

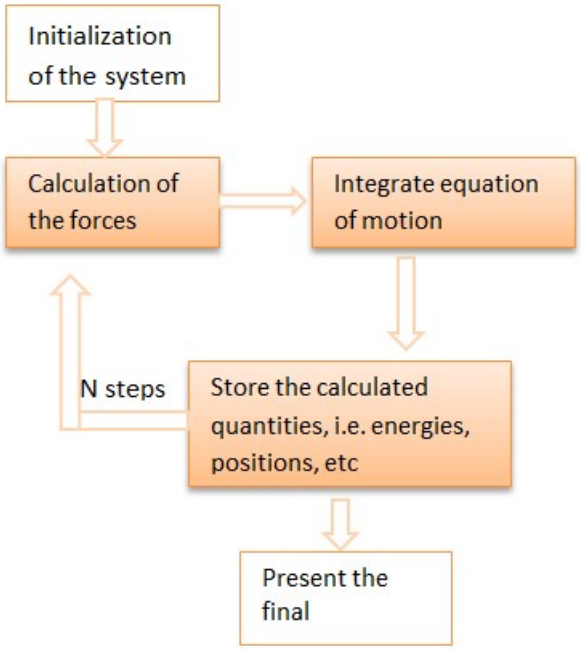


Figure 1: A Flow diagram of MD algorithm

Also, the pseudocode of the Velocity-Verlet integrator is shown in Algorithm 2.1.

Algorithm 2.1 A sample of Velocity-Verlet Algorithm

Initialize:
    $\mathbf{q}, \mathbf{v}, \mathbf{f}$
for $i = 1, \cdots, N_k$ **do**
    $q(i) = q(i) + dt \cdot v(i) + dt \cdot dt/2 \cdot f(i)$ ▷ Update positions and velocities at t+dt
    $v(i) = v(i) + dt/2 \cdot f(i)$ ▷ using velocities and forces at t
end for
function GET FORCES(f)
    Newton's law
end function
for $i = i, N_k$ **do**
    $v(i) = v(i) + dt/2 \cdot f(i)$ ▷ update velocities at t+dt using forces at t+dt
end for

2.3 Molecular Force Field

As we mentioned, according to Newton's law the total force is calculated by the form:

$$\mathbf{F}_I = -\nabla_{\mathbf{q}_I} U(\mathbf{q}), \quad I = 1, 2, \cdots, N \tag{13}$$

where $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N)$ is the positions of the atoms. In most cases, the potential energy consists of potentials with 2-body, 3-body, or/and 4-body interactions

$$U(\mathbf{q}) = \sum_{ij} U_{ij}(\mathbf{q}_i, \mathbf{q}_j) + \sum_{ijk} U_{ijk}(\mathbf{q}_i, \mathbf{q}_j, \mathbf{q}_k) + \sum_{ijkl} U_{ijkl}(\mathbf{q}_i, \mathbf{q}_j, \mathbf{q}_k, \mathbf{q}_l). \tag{14}$$

In a molecular system there are different types of interactions between atoms. Thus, another way of expressing the potential energy is by the sum of the bonded and non-bonded interactions.

$$U(\mathbf{q}) = U_{bonded}(\mathbf{q}) + U_{non-bonded}(\mathbf{q}) \tag{15}$$

According to ref [13], bond stretching, bond rotation, and angle bending, appear in bonded forces (see figure 2). These forces involve intramolecular interactions between atoms bound by covalent bonds, and the computational cost of the bonded forces is O(N), whereas, at non-bonded forces, there are intermolecular interactions between neighboring atoms that are not connected. The electrostatic potential and Lennard-Jones potential due to van der Waals forces, which is the combination of dispersive and repulsive forces, appear at non-bonded forces and can be computed in O(N^2) time. To reduce the computational cost, we can apply several techniques, such as cut-off and tail corrections, periodic boundary conditions, and minimum image convention, ref. [12, 13].

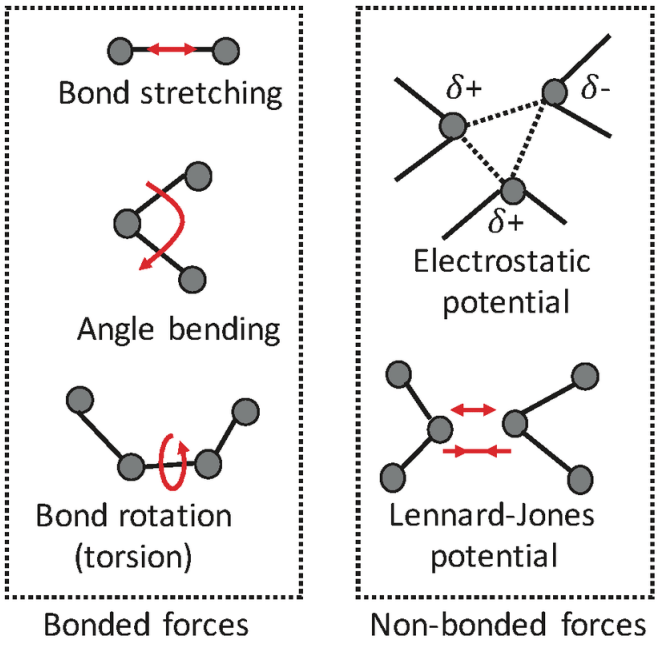


Figure 2: Bonded and non-bonded forces consider in MD simulations.Ref. [13]

Lennard-Jones Potential

The Lennard-Jones potential is a more common choice in molecular simulations and it can accurately model weak van der Waals bonds. It is given by

$$U_{LJ}(r_{ij}) = 4\epsilon\Big[\Big(\frac{\sigma}{r_{ij}}\Big)^{12} - \Big(\frac{\sigma}{r_{ij}}\Big)^{6}\Big] \tag{16}$$

where $r_{ij} = |\mathbf{q}_i - \mathbf{q}_j|$ is the Euclidean distance between two particles,

$$r_{ij} = \sqrt{(q_{i,x} - q_{j,x})^2 + (q_{i,y} - q_{j,y})^2 + (q_{i,z} - q_{j,z})^2}, \tag{17}$$

ϵ is the depth of the potential well and σ is the distance at which the intermolecular potential between the two particles is zero.

The term $(\frac{\sigma}{r_{ij}})^6$ describes the attraction between two atoms while the term $(\frac{\sigma}{r_{ij}})^{12}$ describes the repulsion force as two atoms cannot approach each other when they come closer than one of the sum of their atomic radii σ.

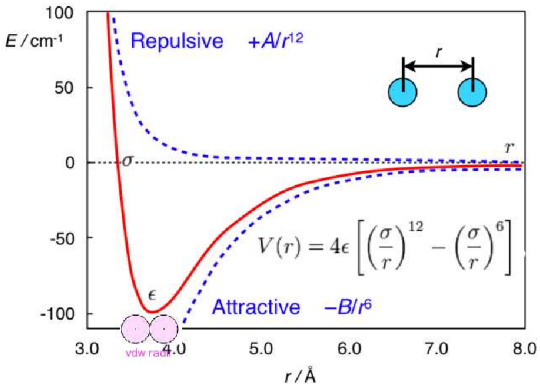


Figure 3: Lennard-Jones potential Ref. [14]

2.3.1 Cut-off and tail corrections

The most time-consuming procedure in an MD simulation is the calculation of all particles' interactions. For a particular configuration, we take each pair of particles and estimate the energies or/and the forces. Therefore, it is necessary to restrict the number of particle pairs, to decrease the computation time. An important characteristic in molecular systems is that the potential energy decreases as the distance between two particles increases. This 'maximum' distance is known as the cut-off distance r_c, ref. [4].

Let us assume a system of N particles in a box that interact via a pair interaction Lennard-Jones potential. The energies depend on the local chemical environment of the particles, according to a cut-off radius r_c of choice. This means that each particle's contribution is dependent on a number of neighboring particles that interact with the central particle examined. The outside-of-the-cut-off-radius particles do not interact with the central molecule. Otherwise, their contribution is minor, as shown in figure 4. This means that the calculation of the interactions is done by

$$U_{LJ}(r) = \begin{cases} 4\epsilon\Big[\big(\frac{\sigma}{r}\big)^{12} - \big(\frac{\sigma}{r}\big)^{6}\Big] & \text{if } r <= r_{cut} \\ 0 & \text{if } r > r_{cut} \end{cases} \tag{18}$$

2.3.2 Periodic Boundary Conditions

The number of particles in systems can be very large. So, in order to overcome this problem, we need to specify the simulation box and the periodic boundary conditions. Figure 4 illustrates the use of periodic boundary conditions in two dimensions. The central simulation box is surrounded by its periodic images. When a moving particle leaves the simulation box at one boundary, one of its images simultaneously enters the simulation box at the opposite boundary. Therefore, the total number of particles in the system is conserved. In order to prevent incorporating spurious interactions between the periodic images of the particles into

the force calculation, the interaction range must be less than the half length of the simulation box — the so-called minimum-image convention Ref. [16].

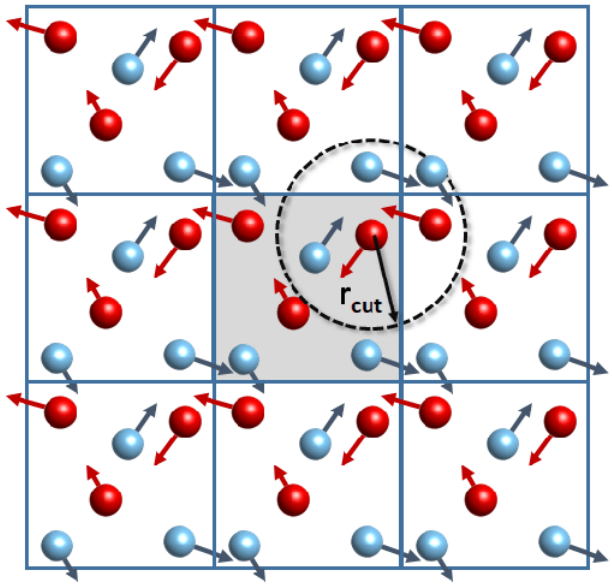


Figure 4: A two-dimensional periodic system with the minimum image convention. Ref. [15]

2.3.3 Minimum Image Convention

The minimum-image convention is a common form of Periodic Boundary Conditions, where each particle in the simulation box interacts with the closest image of the remaining particles in the system. For example, according to the central atom i, we want to compute the minimum distance with respect to a neighboring atom j. So, in every direction of the three-dimensional space, the image of the neighboring atom is obtained according to:

$$r_{ij,x} \leq \frac{L}{2}, \quad r_{ij,y} \leq \frac{L}{2} \quad r_{ij,z} \leq \frac{L}{2} \tag{19}$$

where r_{ij} is the distance between the i-atom and the j-atom, $r_{ij} = \sqrt{r_{ij,x}^2 + r_{ij,y}^2 + r_{ij,z}^2}$, whereas the parameter L stands for the length of the simulation box.

2.4 Coarse Graining

At the microscopic (atomistic) level, simulating molecular systems can be a time-consuming process due to the detailed calculations between all the atoms. Therefore, it is not feasible to apply methods, such as Molecular dynamics or Monte Carlo, to large realistic systems or molecules of complex structure. A way to handle these challenges is to model the systems on the mesoscopic level, which means that we study coarse-grained particles instead of atoms, to decrease the length and time scales accessible by simulations. Ref. [2, 3]

At the mesoscopic level, the CG particles are atoms mapped into small batches, also known as superatoms. Through the procedure of mapping from the atomistic level to the

CG level, the degrees of freedom of the N-atoms are reduced as the new system is consisted of M-particles ($M < N$). During the mapping process, we need semi-empirical functional forms taken from prior knowledge and with a lot of physical intuition to describe the particle motion.

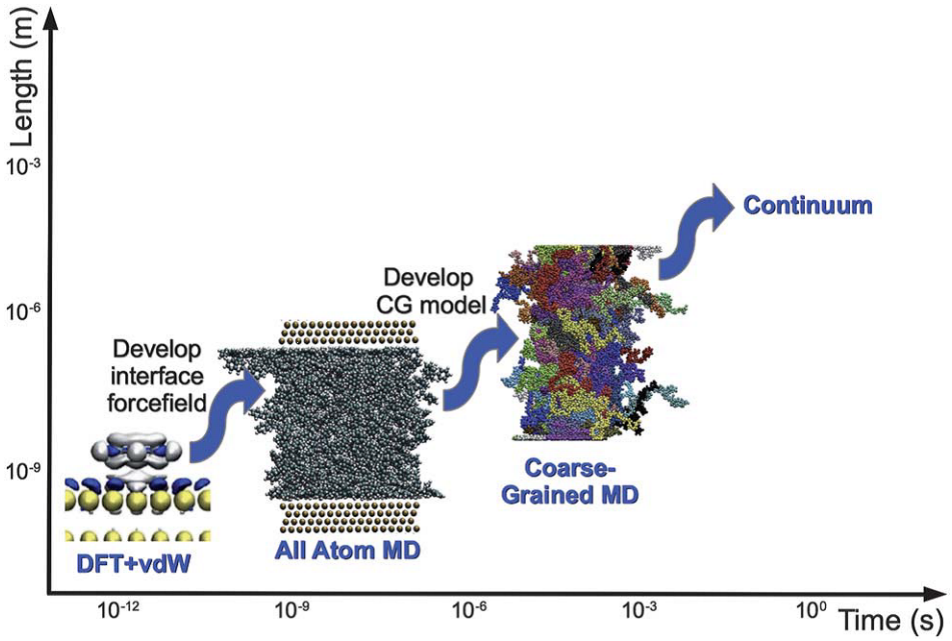


Figure 5: Levels of multiscale dynamics.
Multiscale modeling can be defined as the technology by which problems, such as the calculation of material properties or system behavior, are solved at one level using information from different levels of different time and space scales. There are four distinguished levels: level of quantum mechanical models, level of molecular dynamics models (atomistic), coarse-grained models (mesoscopic), level of continuum models, and level of materials.

In coarse-grained modeling, there are several steps to build a molecular simulation. First, we consider a small model system of rather low molecular weight atoms. After choosing a CG map, representing a function capable of transforming the atomistic system into a CG model, we proceed to estimate the potential energy or CG force field. Then, we run the simulation taking into account the missing atomistic degrees of freedom in the CG structures.

2.4.1 Atomistic Level

Let us assume a system of N atoms in a box of volume V and at temperature T. These N particles, with position vector $q_i, \quad i = 1, ..., N$, obey Newton's Second Law, that is

$$\mathbf{F} = \mathbf{M}\frac{d^2\mathbf{q}}{dt^2}.$$

Obtaining the N-body equations of motion, we have

$$\mathbf{F}(\mathbf{q}) = -\nabla_{\mathbf{q}} U(\mathbf{q}) = -(\nabla \mathbf{q}_1 U(\mathbf{q}), \cdots, \nabla \mathbf{q}_N U(\mathbf{q})) \tag{20}$$

$$\mathbf{M}\frac{d^2\mathbf{q}}{dt^2} = -\nabla_{\mathbf{q}} U(\mathbf{q}), \tag{21}$$

where $F(\mathbf{q}) = (F_1(\mathbf{q}), .., F_N(\mathbf{q})), \quad F \in \mathbb{R}^{3N}$ is the total potential energy and $F_1(\mathbf{q}) = -\nabla_{q_1} U(\mathbf{q}), \quad F_1(\mathbf{q}) \in \mathbb{R}^3$. According to statistical mechanics, the Gibbs canonical measure gives the probability of a state $\mathbf{q}$ as:

$$\mu(d\mathbf{q}) = Z^{-1} e^{-\beta U(\mathbf{q})} d\mathbf{q} \tag{22}$$

where $Z = \int\limits_{\mathbb{R}^{3N}} e^{-\beta U(\mathbf{q})} d\mathbf{q}$ is the partition function and $\beta = \frac{1}{K_B T}$, K_B the Boltzmann constant.

Next, we will apply the principles of the field of statistical mechanics, a tool that is used to relate the microscopic properties of individual atoms to the macroscopic bulk properties of materials.

2.4.2 Coarse-Grained Level and CG Mapping

The new CG coordinates are calculated through a mapping function $\mathbf{\Pi}$

$$\begin{aligned} &\mathbf{\Pi} : \mathbb{R}^{3N} \to \mathbb{R}^{3M} \\ &\mathbf{q} \mapsto \mathbf{\Pi}(\mathbf{q}) = \mathcal{Q}, \end{aligned} \tag{23}$$

where $\mathcal{Q} = (\mathbf{Q}_1, \mathbf{Q}_2, .., \mathbf{Q}_M), \quad \mathcal{Q} \in \mathbb{R}^{3M} \quad (M < N)$ the new CG coordinates. Commonly, the mapping function has a linear form

$$\mathbf{Q}_i = \mathbf{\Pi}_i(q_1, q_2, ..., q_N) = \sum_{j=1}^{N} w_{ij} q_j, \quad i = 1, 2, .., M. \tag{24}$$

The parameters w_{ij} are non-negative weights that determine whether the j-atom will contribute to the i-molecule.

The probability of a state $\mathcal{Q}$ is as well given by the Gibbs measure

$$\bar{\mu}(\mathcal{Q}) = \int_{A(\mathcal{Q})} \mu(\mathbf{q}) d\mathbf{q}. \tag{25}$$

Thus,

$$\bar{\mu}(\mathcal{Q}) = Z^{-1} \int_{A(\mathcal{Q})} e^{-\beta U(\mathcal{Q})} d\mathcal{Q}, \tag{26}$$

where $A(\mathcal{Q}) = \{\mathbf{q} \in \mathbb{R}^{3N} : \mathbf{\Pi}(\mathbf{q}) = \mathcal{Q}\}$.

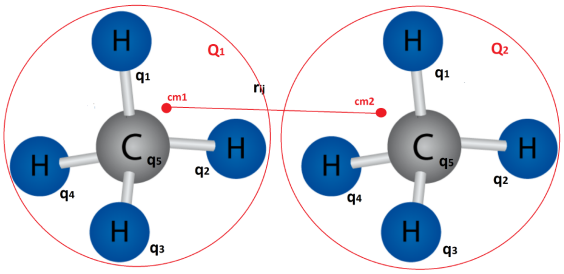


Figure 6: System of the two-body bulk methane.

2.4.3 Potential of Mean Force

The corresponding free energy of eq.26 defines the M-body potential of the mean force (PMF),

$$\bar{U}^{PMF}(\mathcal{Q}) = -\frac{1}{\beta}\ln\int_{A(\mathcal{Q})} e^{-\beta U(\mathcal{Q})}d\mathcal{Q}. \tag{27}$$

Accordingly, the mean force $\bar{F}^{PMF} : \mathbb{R}^{3M} \to \mathbb{R}^{3M}$ is

$$\bar{F}_i^{PMF}(\mathcal{Q}) = -\nabla_{\mathcal{Q}_i} U^{PMF}(\mathcal{Q}), \quad i = 1, 2, .., M. \tag{28}$$

The calculation of the function (27) is computationally costly due to the high dimensionality of the integral. Therefore, we seek to approximate the PMF in the most efficient manner and calculate approximate interaction potentials $\bar{U}_{eff}(\mathcal{Q};\theta)$. These may be parametrized by $\theta \in \Theta$. Now, we can write the probability given by the canonical Gibbs measure of the CG state $\mathcal{Q}$ using the effective potential $\bar{U}_{eff}$

$$\bar{\mu}(d\mathcal{Q};\theta) = \bar{Z}^{-1}\int_{A(\mathcal{Q})} e^{-\beta\bar{U}_{eff}(\mathcal{Q};\theta)}d\mathcal{Q} \quad . \tag{29}$$

In principle, the M-body potential energy of mean force consists of the contributions of two-body, three-body e.t.c interactions. Let the term $r_{ij} = \|\mathbf{Q}_j - \mathbf{Q}_i\|, \quad i, j = 1, 2, ..., M$ denote the distance between the particle i and j. Thus,

$$\bar{U}^{PMF}(\mathcal{Q}) = \sum_{i,j} u_2(r_{ij}) + \sum_{i,j,k} u_3(r_{ij}, r_{ik}, r_{jk}) + ... \quad . \tag{30}$$

It is common to estimate the PMF with the two-body contributions.

$$\bar{U}^{PMF}(\mathcal{Q}) \approx \sum_{i,j} U(r_{ij}) = \bar{U}_{eff}(\mathcal{Q};\theta) \quad . \tag{31}$$

The next step is to present the numerical methods that can approximate the CG potential. These methods are:

- the Boltzmann inversion, (BI), Iterative BI (IBI), and Inverse Monte Carlo (IMC) Ref. [21, 22, 29],
- the force matching (FM) Ref. [2, 23, 17, 24, 25] and
- the relative entropy (RE) Ref. [2, 26, 27, 28].

Boltzmann Inversion

The BI is a family of numerical methods, such as the direct inverse Boltzmann, the iterative inverse Boltzmann, and the inverse Monte Carlo. In all these cases, the n-body PMF is defined by

$$\bar{U}^{(n),PMF}(\mathcal{Q}^{(n)}) = -\frac{1}{\beta}\log g^{(n)}(\mathcal{Q}^{(n)}) \quad ,$$

and the n-body distribution (correlation) function

$$g^{(n)}(\mathcal{Q}^{(n)}) = \frac{M!}{(M-n)!\rho^n}\int_{q\in\mathbb{R}^{3N}:\Pi q=\mathcal{Q}} \mu(q)dq \quad ,$$

where $\mathcal{Q}^{(n)} = (\mathbf{Q}_1, \mathbf{Q}_2, ..., \mathbf{Q}_n)$, of $n(< M)$ particles. All methods use the pair correlation function $g^{(2)}(\mathcal{Q}^{(2)}) =: \bar{g}(R)$, where R is the pairwise distance between two CG particles, thus the CG effective interaction is

$$\bar{U}_{eff}(R) = -\frac{1}{\beta}\log \bar{g}(R) \quad .$$

Specifically, in iterative inverse Boltzmann (IBI) methods, the CG potential is refined at the iteration $(i+1)$ according to

$$\bar{U}_{eff}^{(i+1)}(R) = \bar{U}_{eff}^{(i)}(R) + ck_BT\log\frac{\bar{g}^{(i)}(R)}{\bar{g}^{(ref)}(R)} \quad ,$$

where c is a constant to ensure the stability of the iterative process. In each iteration, convergence is examined to determine whether the CG non-bonded distribution function matches the atomistic one, within the numerical accuracy. The scheme is analogous for both bonded and non-bonded pair potentials, and it can be shown that the two-body pair potential of mean force converges to the reference PMF.

Force Matching

The FM method is applied to more complex and non-linear CG maps and matches the average force on pseudoatoms in the CG system to the expected force from the all-atom system. It determines a CG effective force $\bar{F}(\mathcal{Q};\theta)$ as the solution of the mean-least square minimization:

$$\min_{\theta\in\Theta}\mathbb{E}_\mu[\|h(\mathbf{q}) - \bar{F}(\Pi(\mathbf{q});\theta)\|^2] \quad ,$$

where $\mathbb{E}_\mu[\cdot]$ averages with the respect to $d\mu(\mathbf{q})$ and $h(\mathbf{q}) \in \mathbb{R}^{3M}$ is the local mean force.

Relative Entropy

The RE method's goal is to minimize the

$$\mathcal{R}(\mu|\mu^\theta) = \mathbb{E}_\mu[\log\frac{\mu(\mathbf{q})}{\mu^\theta(\mathbf{q})}] \quad ,$$

where $\mu(\mathbf{q})$ is the microscopic Gibbs measure and $\mu^\theta(\mathbf{q})$ a back-mapping of the approximate CG measure $\bar{\mu}(\mathcal{Q})$. This method can approximate the PMF due to the properties of RE and the definition of the PMF $\bar{\mu}(\mathcal{Q})$. Therefore, the minimization has the following form:

$$\arg\max_{\theta\in\Theta}\{\beta\mathbb{E}_\mu[\bar{U}_{eff}(\Pi(\mathbf{q});\theta) - U(\mathbf{q})] - [\log Z^\theta - \log Z]\},$$

where $Z^\theta = \int\limits_{\mathbb{R}^3} e^{-\beta\bar{U}_{eff}(\mathcal{Q};\theta)}d\mathcal{Q}$. The RE minimization problem can be defined either directly on the path or on the equilibrium ensemble.

3 Introduction to Machine Learning and Gaussian Approximation Methods

Over the past years, the field of machine learning has been increasingly developing. In this section, we will be acquainted with terms, such as machine learning and its categories. Gaussian process regression models have been used in machine learning applications because of their representation flexibility and inherent uncertainty measures over predictions. In this thesis, they are used in order to approximate the forces of a CG system.

3.1 Machine Learning Theory

Machine learning is a branch of artificial intelligence. Machine learning algorithms are able to learn from previously gained data in order to predict or decide without being programmed to do so. There are many ways for the algorithms to work with the training data and also for the algorithms to be constructed. There are three major categories of machine learning [30]:

- **Supervised learning** algorithms try to identify and model the connection between the output and the input. This category includes regression and classification problems. Cases such as the digit recognition example, in which the aim is to assign each input vector to one of a finite number of discrete categories, are called classification problems. If the desired output consists of one or more continuous variables, then the task is called regression.

- **Unsupervised learning** algorithms are using the input data to detect patterns and group them. The main algorithm types of this category are the clustering algorithms, which aim to discover groups of similar examples within the data. Other algorithms are the density estimation algorithms, where the goal is to determine the distribution of data within the input space, and the algorithms that refer to projecting the data from a high-dimensional space down to two or three dimensions for the purpose of visualization. During recent years, unsupervised learning (a.k.a. self-supervised learning) has been used along with deep neural networks for representation learning.

- **Reinforcement learning** methods use trials and errors so as to discover the "best" output by continuously learning from the experiences of the environment. Applications that we can find in this category are computer chess games or self-driving cars.

3.2 Gaussian Process Regression Theory

Gaussian Process Regression (GPR) is a nonlinear, non-parametric regression that helps interpolate between data points scattered in high-dimensional input space ref. [7]. The method is based on Bayesian probability theory and is related to other regression techniques, such as kernel ridge regression (KRR) and linear regression with radial basis functions. It is based on a large amount of data and a flexible function that fits the data and then makes predictions.

Kernel is a measure of similarity between two data points, x, x', usually denoted $k(x, x')$. A detailed discussion will be given in the next section.

Let y denote an unknown function which maps inputs $x \in \mathbb{R}^d$ to outputs, $y : \mathbb{R}^d \to \mathbb{R}$. And let $\mathbf{x} = (x_1, \ldots, x_{n_s})^{tr}$, $x_i \in \mathbb{R}^d$ denote the known n_s independent input observations and output $y_i = y(x_i), i = 1, ..n_s$ of value x_i ref. [19]. The goal is to use the data values x_i to create an estimator and to predict the continuous function $y(x)$ as accurately as possible, considering the expected error of the prediction. Two approaches will be presented: the weight-space and the function-space views of the GPR.

3.2.1 Introduction to Stochastic Processes

Stochastic processes describe a dynamical system that does not behave deterministically in time. Ref. [5] Let T be an ordered set, $(\Omega, \mathcal{F}, \mathcal{P})$ a probability space and $(\mathcal{E}, \mathcal{C})$ a measurable space. A stochastic process is a collection of random variables $X = X_t \, , t \in T$ such that for each fixed $t \in T, X_t$ is a random variable form $(\Omega, \mathcal{F}, \mathcal{P})$ to $(\mathcal{E}, \mathcal{C})$. The set Ω is known as the sample space, where $\mathcal{E}$ is the state space of the stochastic process X_t [5].

A one-dimensional continuous-time Gaussian process $X_t \sim \mathcal{GP}(m(x), k(x, x'))$ is a stochastic process for which $\mathcal{E} = \mathbb{R}$ and all the finite-dimensional distributions are Gaussian, i.e., every finite-dimensional vector $(X_{t_1}, X_{t_2}, \cdots, X_{t_k})$, for any choice of distinct values $t_1, \cdots, t_k \in T$ has a multivariate normal distribution with mean vector $\mathbf{m}_k := \mathbb{E}\mathbf{X}$ and covariance matrix $\mathbf{K}_k = \mathrm{Cov}\,(\mathbf{X}, \mathbf{X})$, symmetric non-negative definite matrix. That is, the random vector $\mathbf{X}$ follows the Gaussian distribution $N(\mathbf{m}, \mathbf{K})$ and has a Gaussian probability distribution function given by

$$f_{\mathbf{X}}(x_k) = (2\pi)^{-n/2} (\det \mathbf{K})^{-1/2} \exp\left[-\frac{1}{2} < \mathbf{K}^{-1}(\mathbf{x} - \mathbf{m}), \mathbf{x} - \mathbf{m} > \right] \tag{32}$$

where $\mathbf{x} = (x_1, \cdots, x_k)$.

In the context of observations, we consider that $y = y(x)$ is a one-dimensional continuous-time Gaussian process, i.e., $X_t \equiv y_i$, corresponding to the output observations.

$$y(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{33}$$

The mean value reflects the expected function value at input x,

$$m(x) = \mathbb{E}[y(x)].$$

The covariance function $k(x, x')$, the kernel of the Gaussian process, models the dependence between the function values at different input points x and x',

$$k(x, x') = \mathbb{E}[(y(x) - m(x))(y(x') - m(x'))].$$

Covariance is a measure of the relationship between two data values, $y(x)$ and $y(x')$, usually expressed as a function of the distance between x and x'. When data are uncorrelated, then their covariance approximates zero.

3.2.2 Linear regression

Assume a training data set comprising n_s observations $\mathbf{x}_n$, where $n = 1, 2, \cdots, n_s$ with the corresponding target values $\{\tilde{t}_n\}$, our aim is to predict the value of y, following the observation equation

$$\tilde{t} = y(\mathbf{x}) + \epsilon \tag{34}$$

for a new value of $\mathbf{x}$. Here ϵ is the observation error, usually following a zero mean Gaussian. We choose a suitable loss function to minimize and to model the conditional distribution $p(\tilde{t}|\mathbf{x})$, which expresses the uncertainty about the value of $\tilde{t}$ for each value of $\mathbf{x} = (x_1, \cdots, x_{n_s})^{tr}$ Ref. [6]. The most common model for regression is the linear model with the form

$$y(\mathbf{x}) = \mathbf{a}^{tr}\phi(\mathbf{x}), \tag{35}$$

where $\mathbf{a} = (a_1, \cdots, a_{n_s})^{tr}$ is the n_s-dimensional weight vector and $\phi(\mathbf{x})$ is a vector of n_s fixed nonlinear basis function with input vector $\mathbf{x}$.

In the Bayesian linear regression, we consider a prior distribution over $\mathbf{a}$ given by an isotropic Gaussian of the form Ref. [6]

$$p(\mathbf{a}) = N(\mathbf{0}, \beta^{-1}\mathbf{I}), \tag{36}$$

where β is the hyper-parameter (precision of the distribution). The probability distribution over $\mathbf{a}$ induces a probability distribution over functions $y(\mathbf{x})$ corresponding to eq.35. This equation also can be written as

$$\mathbf{y} = \mathbf{\Phi}\mathbf{a}, \tag{37}$$

where $\mathbf{\Phi}$ is the design matrix with elements $\Phi_{ij} = \phi_j(\mathbf{x}_i)$. As we notice, the joint distribution of the function $\mathbf{y} = (y(\mathbf{x}_1), \cdots, y(\mathbf{x}_{n_s}))^{tr}$, which is a linear combination of Gaussian distribution values, $\mathbf{a}$, follows the Gaussian distribution too. Therefore, the mean and the covariance of the probability distribution of $\mathbf{y}$ have the forms:

$$\mathbb{E}[\mathbf{y}] = \mathbf{\Phi}\mathbb{E}[\mathbf{a}] = 0, \quad \text{Cov}\,[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^{tr}] = \mathbf{\Phi}\mathbb{E}[\mathbf{a}\mathbf{a}^{tr}]\mathbf{\Phi}^{tr} = \frac{1}{\beta}\mathbf{\Phi}\mathbf{\Phi}^{tr} = \mathbf{K} \tag{38}$$

where $\mathbf{K}$ is the Gram matrix with elements

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\beta}\phi(\mathbf{x}_i)^{tr}\phi(\mathbf{x}_j) \tag{39}$$

and $k(\mathbf{x}, \mathbf{x}')$ is the kernel function. In most applications, it does not exist any prior knowledge about the mean of the Gaussian stochastic process $y(\mathbf{x})$, hence, by symmetry, it can be equal to zero. This is the same as choosing the mean of the prior over the weight values $p(\mathbf{a})$ to be zero in the basis function viewpoint.

3.2.3 Modelling functions: the weight space view

According to a Bayesian viewpoint, we model functions with linear regression. We assume that the output of a linear function of the inputs also includes the noise of the observed target values: Ref. [6, 19]

$$\tilde{t}_n = y(\mathbf{x}_n) + \epsilon_n \tag{40}$$

where $y(\mathbf{x}_n) = \Phi(\mathbf{x}_n)\mathbf{a} = \mathbf{x}_n^{tr}\mathbf{a}$, $\Phi(\mathbf{x}_n)$ the linear form, and ϵ is a random noise n-dimensional vector with independent variables. Assume that the noise process has a Gaussian distribution

$$\epsilon_n \sim N(0, \lambda) \text{ and } \tilde{t}_n | y_n \sim N(y_n, \lambda), \quad \text{where } \lambda \text{ the precision of the noise.} \tag{41}$$

Hence, the joint distribution conditioned on $\mathbf{y}$ is given by

$$p(\tilde{\mathbf{t}}|\mathbf{y}) = p(\tilde{\mathbf{t}}|\mathbf{X}, \mathbf{a}) = N(\mathbf{y}, \lambda \mathbf{I}_{n_s}) = N(\mathbf{X}^{tr}\mathbf{a}, \lambda \mathbf{I}_{n_s}), \tag{42}$$

then this posterior distribution is Ref. [19]

$$p(\mathbf{a}|\tilde{\mathbf{t}}, \mathbf{X}) \propto p(\tilde{\mathbf{t}}|\mathbf{X}, \mathbf{a}) p(\mathbf{a}) = N(\lambda^{-1}\mathbf{A}^{-1}\mathbf{X}\tilde{\mathbf{t}}, \mathbf{A}^{-1}) \tag{43}$$

where $\mathbf{A} = \beta \mathbf{I} + \lambda^{-1}\mathbf{X}\mathbf{X}^{tr}$. As inference is performed over the weights, this is referred to as "the weight space view of regression". To predict a new output $\tilde{t}_*$ of a new point $\mathbf{x}_*$, we can average out the noise term and then concentrate on the function $y_* = \tilde{t}_* - \epsilon_*$. The marginal distribution for $\mathbf{y}$

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathbf{X}, \tilde{\mathbf{t}}) &= \int p(y_*|\mathbf{x}_*, \mathbf{a}) p(\mathbf{a}|\tilde{\mathbf{t}}, \mathbf{X}) d\mathbf{a} \\ &= N(\lambda^{-1}\mathbf{x}_*^{tr}\mathbf{A}^{-1}\mathbf{X}\tilde{\mathbf{t}}, \mathbf{x}_*^{tr}\mathbf{A}^{-1}\mathbf{x}_*). \end{aligned} \tag{44}$$

Thus, we generate this posterior predictive distribution over y_* by first sampling weights from the posterior distribution over weights and then using these sampled weights to generate predictions for the new input points.

Example of Modelling functions: the weight space view

Let y denote an unknown function which maps inputs $\mathbf{X}$ to outputs $\tilde{\mathbf{t}}$: $y : \mathbf{X} \to \mathbf{Y}$, according to Ref. [19].

Inputs and corresponding outputs

n	x_n	$\tilde{t}_n$
1	0.9	0.1
2	3.8	1.2
3	5.2	2.1
4	6.1	1.1
5	7.5	1.5
6	9.6	1.2

Our goal is to predict the output of a new input point, eg. $x* = 3$. We construct the linear regression form,

$$\begin{aligned} \tilde{t}_n &= y(x_n) + \epsilon_n \\ &= a_0 + a_1 x_n + \epsilon_n \quad \text{or} \\ \tilde{t}_n &= \mathbf{x}_n^{tr}\mathbf{a} + \epsilon_n \end{aligned}$$

where the vectors $\mathbf{x}_n, \mathbf{a}$ are defined as

$$\mathbf{x}_n = \begin{bmatrix} 1 \\ x_n \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

To estimate the output of the new input x_*, we estimate the weights from the previous observations

$$\mathbf{X} = \begin{bmatrix} 1 & 0.9 \\ 1 & 3.8 \\ \vdots & \\ 1 & 9.6 \end{bmatrix} \quad \tilde{\mathbf{t}}_n = \begin{bmatrix} 0.1 \\ 1.2 \\ \vdots \\ 1.2 \end{bmatrix}$$

$$\mathbf{A}_n^{-1} = \begin{bmatrix} 0.08 & -0.01 \\ -0.01 & 0.002 \end{bmatrix}$$

Assume that the weight vector takes the value of the mean. So, if $\mathbf{x}_* = [1.0, 3.0]$ and $y(x^*) = \mathbf{x}_*^{tr}\mathbf{a}$, the value of $y(x^*)$ is equal to $y(x^*) = 0.9037$. If we take an uninformative prior, i.e. large variance Σ, then the solution of this method is equivalent to the solution of the least squares problem. For this problem, according to the matrix $\mathbf{A}_n = \mathbf{X}\mathbf{X}^{tr}$ and $\mathbf{a} = \mathbf{A}_n^{-1}\mathbf{X}\tilde{\mathbf{t}}_n$, we have the value $y(x^*) = 0.9039$. The code of these examples can be found in Appendix.A.

3.2.4 Modelling functions: the function space view

In the weight space view of the previous section, we focus our interest on distributions over weights. As each set of weights implies a particular function, a distribution over weights implies a distribution over functions. In Gaussian process regression, we focus directly on such distributions over functions.

A Gaussian process defines a distribution over functions such that, if we take any two pairs of input-output points in a function, then the values of the outputs follow a joint Gaussian distribution. In this case, we also have the eq.40, where the noise term is a random variable that follows a particular distribution and reflects the inherent randomness in the observations. Thus, we have the function $y(\mathbf{x})$ as a Gaussian process:

$$y(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{45}$$

where the mean function $m(\mathbf{x})$ models the expected function value at input, $m(\mathbf{x}) = \mathbb{E}[y(\mathbf{x})]$. The prior mean function is often set to $m(\mathbf{x}) = 0$ to avoid expensive posterior computations. On the other hand, the covariance function reflects the dependence between the function values at different input points $\mathbf{x}$ and $\mathbf{x}'$, $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(y(\mathbf{x}) - m(\mathbf{x}))(y(\mathbf{x}') - m(\mathbf{x}'))]$, known as the kernel function. If a kernel function is chosen, then it is easy to draw a priori function values using the Gaussian process, and posterior function values conditional upon previous observations.

First, we start by selecting a data set of input points to construct the kernel matrix for the Gaussian process. Let $\mathbf{X}$ be a matrix denoting the data set. We set the mean function

to zero and calculate the covariances between all inputs $k(\mathbf{x}_i, \mathbf{x}_j), i, j = 1 \cdots n_s$. Thus, the marginal distribution for $\mathbf{y}$ according to (38)

$$\mathbf{y} \sim N(0, K(\mathbf{X}, \mathbf{X})) \sim N(\mathbf{0}, \mathbf{K}), \tag{46}$$

where $\mathbf{y} = [y(\mathbf{x}_1), \cdots, y(\mathbf{x}_{n_s})]$.

The kernel function that determines $\mathbf{K}$ is typically chosen to express the property that, for points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are similar, the corresponding values $y(\mathbf{x}_i)$ and $y(\mathbf{x}_j)$ will be more strongly correlated than for dissimilar points. To calculate the marginal distribution $p(\tilde{\mathbf{t}})$, considering the input vector $(\mathbf{x}_1, \cdots, \mathbf{x}_{n_s})$, we integrate over $\mathbf{y}$.

$$p(\tilde{\mathbf{t}}) = \int p(\tilde{\mathbf{t}}|\mathbf{y}) p(\mathbf{y}) d\mathbf{y} = \mathbb{E}_{\mathbf{y}}[p(\tilde{\mathbf{t}}|\mathbf{y})] = N(\mathbf{0}, \mathbf{C}) \tag{47}$$

where the covariance matrix $\mathbf{C}$ has elements

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \lambda \delta_{nm}. \tag{48}$$

Analytical calculation of $p(\tilde{\mathbf{t}})$

According to the following equations:

$$p(\tilde{\mathbf{t}}|\mathbf{y}) = \frac{1}{(2\pi)^{\frac{n_s}{2}}} (|\lambda^{-1}\mathbf{I}_{n_s}|)^{\frac{1}{2}} e^{-\frac{1}{2}(\tilde{\mathbf{t}}-\mathbf{y})^{tr}(\lambda^{-1}\mathbf{I}_{n_s})(\tilde{\mathbf{t}}-\mathbf{y})} \tag{49}$$

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{\frac{n_s}{2}}} \frac{1}{(|\mathbf{K}|)^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{y}^{tr}\mathbf{K}^{-1}\mathbf{y}} \tag{50}$$

$$
\begin{aligned}
p(\tilde{\mathbf{t}}) &= \int \frac{1}{(2\pi)^{\frac{n_s}{2}}}|\lambda^{-1}\mathbf{I}_{n_s}|^{\frac{1}{2}}e^{-\frac{1}{2}(\tilde{\mathbf{t}}-\mathbf{y})^{tr}(\lambda^{-1}\mathbf{I}_{n_s})^{-1}(\tilde{\mathbf{t}}-\mathbf{y})}\frac{1}{(2\pi)^{\frac{n_s}{2}}}\frac{1}{|\mathbf{K}|^{\frac{1}{2}}}e^{-\frac{1}{2}\mathbf{y}^{tr}\mathbf{K}^{-1}\mathbf{y}}dy \\
&= \frac{1}{(2\pi)^{n_s}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}|}\right]^{\frac{1}{2}}\int e^{-\frac{1}{2}[(\tilde{\mathbf{t}}-\mathbf{y})^{tr}\mathbf{\Sigma}^{-1}(\tilde{\mathbf{t}}-\mathbf{y})+\mathbf{y}^{tr}\mathbf{K}^{-1}\mathbf{y}]}dy \\
&= \frac{1}{(2\pi)^{n_s}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}|}\right]^{\frac{1}{2}}\int e^{-\frac{1}{2}[\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}-\mathbf{y}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}-\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\mathbf{y}+\mathbf{y}^{tr}(\mathbf{\Sigma}^{-1}+\mathbf{K}^{-1})\mathbf{y}]}dy \\
&\overset{\mathbf{A}=(\mathbf{\Sigma}^{-1}+\mathbf{K}^{-1})}{=} \frac{1}{(2\pi)^{n_s}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}|}\right]^{\frac{1}{2}}\int e^{-\frac{1}{2}[\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}-\mathbf{y}^{tr}\mathbf{A}\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}-\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\mathbf{A}^{-1}\mathbf{A}\mathbf{y}+\mathbf{y}^{tr}\mathbf{A}\mathbf{y}]}dy \\
&\overset{(MIL)}{=} \frac{1}{(2\pi)^{n_s}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}|}\right]^{\frac{1}{2}}\int e^{-\frac{1}{2}[\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}+(\mathbf{y}-\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}})^{tr}\mathbf{A}(\mathbf{y}-\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}})+\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}}]}dy \\
&= \frac{1}{(2\pi)^{n_s}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}|}\right]^{\frac{1}{2}}\int e^{-\frac{1}{2}[(\mathbf{y}-\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}})^{tr}\mathbf{A}(\mathbf{y}-\mathbf{A}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}})+\tilde{\mathbf{t}}^{tr}(\mathbf{\Sigma}+\mathbf{K})^{-1}\tilde{\mathbf{t}}]}dy \\
&= \frac{1}{(2\pi)^{n_s/2}}\left[\frac{1}{|\mathbf{\Sigma}||\mathbf{K}||\mathbf{\Sigma}^{-1}+\mathbf{K}^{-1}|}\right]^{\frac{1}{2}}e^{\tilde{\mathbf{t}}^{tr}(\mathbf{\Sigma}+\mathbf{K})^{-1}\tilde{\mathbf{t}}} \\
&\overset{(\lambda\mathbf{I}_{n_s}=\mathbf{\Sigma})}{=} \frac{1}{(2\pi)^{n_s/2}}\left[\frac{1}{|\lambda\mathbf{I}_{n_s}||\mathbf{K}||\lambda^{-1}\mathbf{I}_{n_s}+\mathbf{K}^{-1}|}\right]^{\frac{1}{2}}e^{\tilde{\mathbf{t}}^{tr}(\lambda\mathbf{I}_{n_s}+\mathbf{K})^{-1}\tilde{\mathbf{t}}} \\
&= \frac{1}{(2\pi)^{n_s/2}}\left[\frac{1}{|\lambda^{-1}\mathbf{I}_{n_s}\mathbf{K}\lambda\mathbf{I}_{n_s}+\mathbf{K}\mathbf{K}^{-1}\lambda\mathbf{I}_{n_s}|}\right]^{\frac{1}{2}}e^{\tilde{\mathbf{t}}^{tr}(\lambda\mathbf{I}_{n_s}+\mathbf{K})^{-1}\tilde{\mathbf{t}}} \\
&= \frac{1}{(2\pi)^{n_s/2}}\left[\frac{1}{|\mathbf{K}+\lambda\mathbf{I}_{n_s}|}\right]^{\frac{1}{2}}e^{\tilde{\mathbf{t}}^{tr}(\lambda\mathbf{I}_{n_s}+\mathbf{K})^{-1}\tilde{\mathbf{t}}} \qquad (51)
\end{aligned}
$$

(52)

where we applied the Matrix Inversion Lemma (MIL) $\mathbf{B}^{-1} = \mathbf{\Sigma} - \mathbf{\Sigma}(\mathbf{\Sigma}+\mathbf{K})^{-1}\mathbf{\Sigma}$ (eq.3.45-Bishop)

$$
\begin{aligned}
\tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} - \tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\mathbf{B}^{-1}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} &= \tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} - \tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}(\mathbf{\Sigma}-\mathbf{\Sigma}(\mathbf{\Sigma}+\mathbf{K})^{-1}\mathbf{\Sigma})\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} \qquad (53)\\
&= \tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} - \tilde{\mathbf{t}}^{tr}\mathbf{\Sigma}^{-1}\tilde{\mathbf{t}} + \tilde{\mathbf{t}}^{tr}(\mathbf{\Sigma}+\mathbf{K})^{-1}\tilde{\mathbf{t}} \\
&= \tilde{\mathbf{t}}^{tr}(\mathbf{\Sigma}+\mathbf{K})^{-1}\tilde{\mathbf{t}}
\end{aligned}
$$

As we mentioned, the goal of the regression is to predict the target variable $\tilde{t}_{n_s+1}$ for a new input vector $\mathbf{x}_{n_s+1}$, thus the evaluation of the predictive distribution $p(\tilde{t}_{n_s+1}|\tilde{\mathbf{t}}_{n_s})$ is needed, which is conditioned on the variables $\mathbf{x}_1, \cdots, \mathbf{x}_{n_s}$ and $\mathbf{x}_{n_s+1}$. Thus, from eq. 47 we have

$$
p(\tilde{\mathbf{t}}_{n_s+1}) = N(\mathbf{0}, \mathbf{C}_{n_s+1}) \qquad (54)
$$

where $\tilde{\mathbf{t}}_{n_s+1} = (\tilde{\mathbf{t}}_{n_s}, \tilde{t}_{n_s+1})$ and we partition the covariance matrix $\mathbf{C}_{n_s+1}$ as follows

$$
\mathbf{C}_{n_s+1} = \begin{pmatrix} \mathbf{C}_{n_s} & \mathbf{k} \\ \mathbf{k}^{tr} & c \end{pmatrix}, \quad c = k(\mathbf{x}_{n_s+1}, \mathbf{x}_{n_s+1}) + \lambda^{-1} \text{ (scalar)} . \qquad (55)
$$

The second-order statistics, namely the mean and covariance, is given by
$\tilde{t}_{n_s+1}|\tilde{\mathbf{t}}_{n_s} \sim N(m(\mathbf{x}_{n_s+1}), \sigma^2(\mathbf{x}_{n_s+1}))$

$$m(\mathbf{x}_{n_s+1}) = \mathbf{k}^{tr}\mathbf{C}_{n_s}^{-1}\tilde{\mathbf{t}} \tag{56}$$

where $\mathbf{k}$ is the vector with elements $k(x_i, x_{n_s+1}), i = 1, \cdots, n_s$.

$$\sigma^2(\mathbf{x}_{n_s+1}) = c - \mathbf{k}^{tr}\mathbf{C}_{n_s}^{-1}\mathbf{k}. \tag{57}$$

In order to calculate $p(\tilde{t}_{n_s+1}|\tilde{\mathbf{t}}_{n_s})$, the Bayes' theorem is applied:

$$p(\tilde{t}_{n_s+1}|\tilde{\mathbf{t}}_{n_s}) = \frac{p(\tilde{\mathbf{t}}_{n_s}, \tilde{t}_{n_s+1})}{p(\tilde{\mathbf{t}}_{n_s})}, \text{ where } p(\tilde{\mathbf{t}}_{n_s}, \tilde{t}_{n_s+1}) = p(\tilde{\mathbf{t}}_{n_s+1}), \tag{58}$$

according to Ref. [6]

$$p(\tilde{t}_{n_s+1}|\tilde{\mathbf{t}}_{n_s}) = \frac{(\det(\mathbf{C}_{n_s+1}))^{-\frac{1}{2}} e^{-\frac{1}{2}\tilde{\mathbf{t}}_{n_s+1}^{tr}(\mathbf{C}_{n_s+1})^{-1}\tilde{\mathbf{t}}_{n_s+1}}}{(\det(\mathbf{C}_{n_s}))^{-\frac{1}{2}} e^{-\frac{1}{2}\tilde{\mathbf{t}}_{n_s}^{tr}(\mathbf{C}_{n_s})^{-1}\tilde{\mathbf{t}}_{n_s}}}. \tag{59}$$

The second-order statistics, i.e. the mean and the covariance, are determinants for the Gaussian process regression definition, as it seems that the regression depends on the mean and covariance of $\mathbf{x}_{n_s+1}$, due to function $\mathbf{k} \in \mathbb{R}^{(n_s+1)}$.
An expansion in radial basis function of eq.56 gives the following form:

$$m(\mathbf{x}_{n_s+1}) = \sum_{n=1}^{n_s} a_n k(\mathbf{x}_n, \mathbf{x}_{n_s+1}) \tag{60}$$

where a_n is the n^{th} of $\mathbf{C}_{n_s}^{-1}\tilde{\mathbf{t}}$.

Example of Modelling functions: the function space view

As in the previous example, the data of the model are the same according to Ref. [19]. The vectors $\mathbf{x}_n, \tilde{\mathbf{t}}_n$ are defined as

n	x_n	$\tilde{t}_n$
1	0.9	0.1
2	3.8	1.2
3	5.2	2.1
4	6.1	1.1
5	7.5	1.5
6	9.6	1.2

In this method, we can calculate the weights $\mathbf{a} = [\mathbf{K} + \lambda\mathbf{I}]^{-1}\tilde{\mathbf{t}}_n$. Also, we have the parameters $\lambda = 0.1, \delta = 1$, $\theta = 1.1$. and the new input point $x_* = 3$. Thus,

n	x_n	$\tilde{t}_n$	a_n	$k(x_n, x_*)$	$a_n k(x_n, x_*)$
1	0.9	0.1	0.08	1.6e-01	1.3e-02
2	3.8	1.2	0.20	7.6e-01	1.5e-02
3	5.2	2.1	2.47	1.3e-01	3.3e-01
4	6.1	1.1	-1.23	1.8e-03	-2.3e-02
5	7.5	1.5	1.48	2.3e-04	3.4e-04
6	9.6	1.2	0.87	1.5e-08	1.3e-08
				$y(x^*) = \sum_{n=1}^{6} a_n k(x_n, x_*)$:	0.48

If the hyperparameters take different values, $y(x^*)$ also changes, e.g. for $\theta = 2.2 \rightarrow y(x^*) = 0.92$. By changing the set of hyperparameters, the $y(x^*)$ value changes, and by applying the optimal set we can better approximate the value of the least square problem (0.9039). The code of this example can be found in Appendix.A

3.2.5 Switching back to the weight view

We want to approximate $y(\mathbf{x})$ by $f(\mathbf{x})$ expressed as a linear combination of n_s basis functions. As we mentioned before we can rewrite the eq.56.

$$f(\mathbf{x}) = m(\mathbf{x}_n) = \sum_{i=1}^{n_s} a_i k(\mathbf{x}, \mathbf{x}_i). \tag{61}$$

One very popular choice of a kernel is the radial basis function kernel, which is defined as

$$k(\mathbf{x}, \mathbf{x}_i) = \delta^2 \exp\left(-\frac{|\mathbf{x} - \mathbf{x}_i|^2}{2\theta^2}\right) \tag{62}$$

where each $\mathbf{x}_i$ is a previously observed input value in $\mathbf{X}$ and the weights are collected in the vector

$$\mathbf{a} = [\mathbf{K} + \lambda \mathbf{I}]^{-1} \tilde{\mathbf{t}}_n. \tag{63}$$

where λ is the regularization parameter and the matrix $\mathbf{K}+\lambda\mathbf{I}$ corresponds to the matrix $\mathbf{C}_{n_s}$ of the eq.56 and eq.55. This equation shows that Gaussian process regression is equivalent to a linear regression model using basis functions k to project the inputs into a feature space.

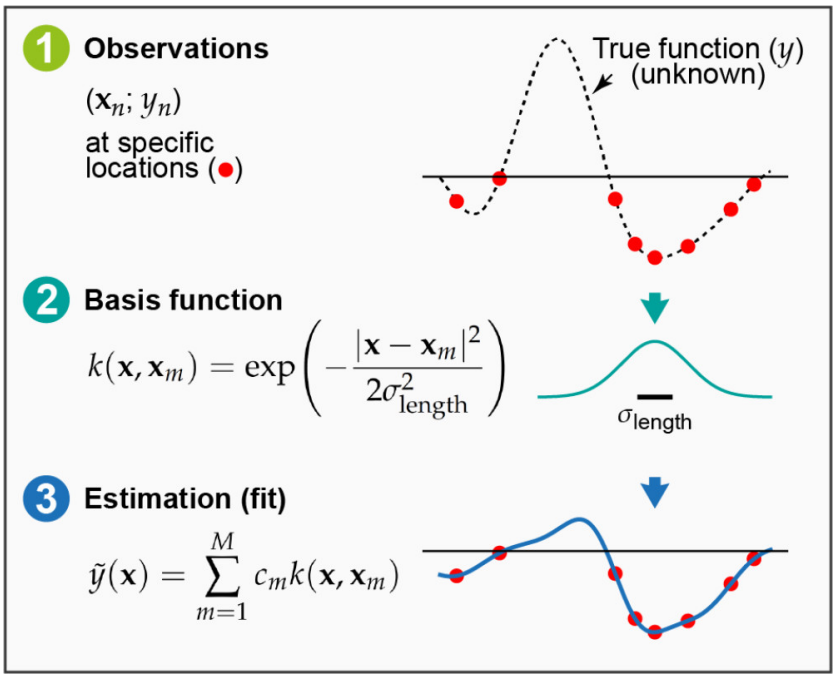


Figure 7: The prediction of a GPR model.Ref. [7] (1) observations of an unknown function at a number of locations, (2) basis functions centered at the data locations, (3) an estimation, $\tilde{y}$, defined by the set of coefficients and the corresponding basis functions; this is the prediction of the GPR model.

The basis function, k (symmetric and positive semidefinite), is placed at arbitrary n_s locations in the input space, $\mathbf{x}_i$, and a_i are the weights. The $\mathbf{K}$ matrix is positive semidefinite, if for an arbitrary set of inputs $\mathbf{x}_n$, the function $\mathbf{K}_{nn} = k(\mathbf{x}_n, \mathbf{x}_n)$ is positive semidefinite. The length scale hyperparameter, θ, affects how quickly the model converges to the real data.

Hyperparameter is a parameter whose value controls the learning process and the behavior of the fit. Hyperparameters are estimated from experience or iteratively optimized using data. A detailed discussion will be given in the section 3.4.

3.3 Relation to Kernel Ridge Regression

The fitting of the Gaussian process regression model can also be written as a (kernel) ridge regression problem. That is, to find a by minimizing the loss (or cost) function

$$L = \sum_{n=1}^{n_s} \frac{[y_n - f(\mathbf{x}_n)]^2}{\sigma_i^2} + \lambda \mathbf{R}, \tag{64}$$

where $\mathbf{R}$ is the regularization parameter. The relative importance of individual data points is controlled by the parameters σ_i. In the sequel, we choose $\sigma_i = 1$ reflecting the equal importance of the input data.

The regularization terms that we work with in this thesis are:

$$\mathbf{R} = \sum_{n,n'}^{n_s} a_n k(x_n, x_{n'}) a_{n'} = \mathbf{a}^{tr} \mathbf{K}_{n_s n_s} \mathbf{a} \tag{65}$$

and

$$\mathbf{R} = \sum_{n,n'}^{n_s} a_n a_{n'}. \tag{66}$$

The key difference between kernel ridge regression and Gaussian process regression is the interpretation of regularization. In KRR, there is an empirical regularization parameter λ that needs to be optimized. In the Bayesian framework of Gaussian process regression, the regularization parameter can be identified with the variance of noisy observations, $\lambda = \sigma^2$, i.e. it is a physical quantity that is an inherent property of the data.

In the loss function, the goal of the first term is to achieve a close fit to the data points, and the second term makes sure that the coefficients remain small to avoid overfitting. The parameters $\{\sigma\}_i^{n_s}$ (with the hyperparameters θ) adjust the balance between accurately reproducing the fitting data points and the overall smoothness of the estimator. The loss function:

$$L = (\mathbf{y} - \mathbf{K}_{n_s n_s}\mathbf{a})^{tr} \mathbf{\Sigma}^{-1} (\mathbf{y} - \mathbf{K}_{n_s n_s}\mathbf{a}) + \lambda \mathbf{a}^{tr} \mathbf{K}_{n_s n_s} \mathbf{a} \tag{67}$$

where Σ is a diagonal matrix of size n_s, collecting all the σ_i values, with $\Sigma_{ii} = \sigma_i^2$. We differentiate the last equation, to minimize L. $\nabla_{\mathbf{a}^{tr}} L = 0$.

$$-\mathbf{K}_{n_s n_s} \mathbf{\Sigma}^{-1} \mathbf{y} + \mathbf{K}_{n_s n_s} \mathbf{\Sigma}^{-1} \mathbf{K}_{n_s n_s}^{tr} \mathbf{a} + \lambda \mathbf{K}_{n_s n_s} \mathbf{a} = 0. \tag{68}$$

3.4 Hyperparameters

The covariance of $y(\mathbf{x})$ given any two values of $\mathbf{x}$ is evaluated by the kernel function

$$\mathbb{E}[y(\mathbf{x}_n), y(\mathbf{x}_m)] = k(\mathbf{x}_n, \mathbf{x}_m). \tag{69}$$

In the Gaussian process model, the forecast depends on the choice of the covariance function. In practice, it is preferred to use a parametric function, where the fitting values of the hyperparameters, such as the length scale of the correlations and the accuracy of the noise, are chosen according to the data. In principle, in big data models, if they were not computationally expensive, these parameters could be optimized from the data itself Ref. [8].

The regularization term λ from eq.(105) solves the overfitting problem by forcing the coefficients to remain small Ref. [7]. Also, it handles the noisy data and adjusts the balance between the exact reproduction of the fitting data points and the overall smoothness of the estimator.

According to the Bayesian theorem

$$\text{Bayes' Rule: } \mathit{Posterior} = \frac{\mathit{Likelihood} \times \mathit{Prior}}{\mathit{Evidence}} \quad p(\Theta|\mathbf{q}) = \frac{p(\mathbf{q}|\Theta) \times p(\Theta)}{p(\mathbf{q})} \tag{70}$$

which uses the conditional probabilities

$$p(\Theta|\mathbf{q}) = \frac{p(\mathbf{q},\Theta)}{p(\mathbf{q})} \quad p(\mathbf{q}|\Theta) = \frac{p(\Theta,\mathbf{q})}{p(\Theta)} \tag{71}$$

where Θ includes the knowledge of the data's hyperparameters, we observe that for a larger amount of training data, the influence of the prior decreases.

4 Coarse-grained molecular models with Gaussian approximation methods

In this section, we analyze the framework of our method, involving many-body interactions from training data of all CG particles. Every minimization problem begins with the definition of the minimization function, known as the cost function. Therefore, we constructed this function according to prior knowledge of previous sections. Our priority is to approximate the many-body PMF using local multi-body terms, based on 'descriptors'. We define the total force as a sum of local terms. We mention two ways to parameterize our model. The first approach refers to a system where we apply only pair interactions of CG particles, while in the second, we have interactions between two and three CG particles.

4.1 Minimization Problem

As we mentioned, there are numerical parameterization methods where the corresponding CG model is constructed and they can approximate the properties of the microscopic model based on principles of statistical mechanics. These methods examine the optimization of the proposed parametric model corresponding to different minimization principles. The optimization problem that occurs involves a cost function L of pre-selected observable Φ and the minimization over a parameter set $\mathbf{a}$

$$\min_{\mathbf{a}} L(\Phi; \mathbf{a}) \tag{72}$$

In the Force Matching (FM) approach the analytical form of the optimization problem is

$$\min_{\mathbf{a}} \mathbb{E}_{\mu}\Big[||(\mathbf{F}o\Pi)(\cdot) - \mathbf{F}_{CG}(\cdot; \mathbf{a})||_{3M}^2\Big] \tag{73}$$

where $(\mathbf{F}o\Pi)$ is known from samples and the second term $\mathbf{F}_{CG}$ contains the approximated values.

Here, we have observations/samples $\{\mathbf{q}_i, \mathbf{F}_i\}_{i=1}^{n_s}$, from the all atom simulations. From there we evaluate, according to the CG map the $\mathcal{Q} = \Pi\mathbf{q}$,

$$\min_{\mathbf{a}} \frac{1}{n_s} \sum_{l=1}^{n_s} \sum_{I=1}^{M} ||(\mathbf{F}o\Pi_I)(\mathbf{q}^{(l)}) - \mathbf{F}_{CG,I}(\Pi\mathbf{q}^{(l)}; \mathbf{a})||_3^2 \tag{74}$$

where the force is calculated by

$$\mathbf{F}_{CG,I}(\mathcal{Q}; \mathbf{a}) = -\nabla_{\mathbf{Q}_I} U_{CG}(\mathcal{Q}; \mathbf{a}), \quad \mathbf{F}_{CG,I}(\mathcal{Q}; \mathbf{a}) \in \mathbb{R}^3. \tag{75}$$

The minimization problem eq.74 is in analogy to the one with the loss function eq.64, where y_n corresponds to the term $(\mathbf{F}o\Pi_I)(\mathbf{q}^{(l)})$ and $f(\mathbf{x}_n)$ with the term $\mathbf{F}_{CG,I}(\Pi\mathbf{q}^{(l)}; \mathbf{a})$. The term $\lambda\mathbf{I}$ in eq.64 corresponds to the regularization approach, which we do not include here in eq.74.

The parametrized CG potential that describes the free energy surface U_{CG} is in principle a many-body potential of the 3M CG coordinates ($U_{CG}(\mathcal{Q}; \mathbf{a}) \in \mathbb{R}$) and $\mathbf{a} = (a_1, a_2, \cdots, a_M)$,

the optimization parameters.

$$\begin{aligned} U_{CG}(\mathbf{Q}_1, \mathbf{Q}_2, ..., \mathbf{Q}_M; \mathbf{a}) = & \sum_j W_{mono}(D_{mono}(\mathbf{Q}_{j,\alpha}); \acute{\mathbf{a}}) \\ & + \sum_{j<k} W_{dimer}(D_{dimer}(\mathbf{Q}_{j,\{\alpha\}}, \mathbf{Q}_{k,\{\beta\}}); \hat{\mathbf{a}}) \\ & + \sum_{j,k,l} W_{trimer}(D_{trimer}(\mathbf{Q}_{j,\{\alpha\}}, \mathbf{Q}_{k,\{\beta\}}, \mathbf{Q}_{l,\{\gamma\}}); \mathbf{a}^*) \end{aligned} \tag{76}$$

where letters i, j, k, l iterate over molecules, α, β, γ iterate over CG sites within each molecule and parameter $\mathbf{a} = (\acute{\mathbf{a}}, \hat{\mathbf{a}}, \mathbf{a}^*)$ and $W_{mono}, W_{dimer}, W_{trimer}$ denote the corresponding potential function according to the type of interaction. The monomer, dimer, and trimer descriptors D are constructed from all pairwise distances between the involved CG sites. The monomer, for example, is defined by the set of all intramolecular distances:

$$D_{mono}(\mathcal{Q}_{j,\{\alpha\}}; \acute{\mathbf{a}}) = \{D^{jj}_{\alpha\beta} | \alpha = 1, ..., m; \beta = \alpha + 1, .., m\} \tag{77}$$

where $D^{jk}_{\alpha\beta} = |\mathbf{Q}_{j,\alpha} - \mathbf{Q}_{k,\beta}|$ is the distance between site α in molecule j and site β in molecule k and m is the number of CG sites within a monomer.

In our case, i.e. the methane system, we have only one type of CG sites within a molecule, so we don't need W_{mono},

$$\begin{aligned} U_{CG}(\mathcal{Q}; \mathbf{a}) &= \sum_{j,k} W_{dimer}(D_{dimer}(\mathbf{Q}_j, \mathbf{Q}_k); \hat{\mathbf{a}}) + \sum_{j,k,l} W_{trimer}(D_{trimer}(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l); \mathbf{a}^*) \\ &= \sum_{j=1}^{M} \sum_{k=j+1}^{M} W_{dimer}(D_{dimer}(\mathbf{Q}_j, \mathbf{Q}_k); \hat{\mathbf{a}}) + \sum_{j=1}^{M} \sum_{k=j+1}^{M} \sum_{l=k+1}^{M} W_{trimer}(D_{trimer}(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l); \mathbf{a}^*), \end{aligned} \tag{78}$$

where $D_{dimer}(\mathbf{Q}_j, \mathbf{Q}_k) = D(\mathbf{Q}_j, \mathbf{Q}_k) \in \mathbb{R}, \quad j, k = 1, 2, \cdots, M$ is the Euclidean distance between two CG particles, and

$$D(\mathbf{Q}_j, \mathbf{Q}_k) = r_{jk} = \sqrt{|Q_{j1} - Q_{k1}|^2 + |Q_{j2} - Q_{k2}|^2 + |Q_{j3} - Q_{k3}|^2}. \tag{79}$$

The trimer descriptor $D_{trimer}(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)$ can have different forms, and it's designed to preserve the physical symmetries of the coarse system. The trimer descriptor, which we applied, is given in section 4.3.

4.2 Application of Gaussian Process Regression - Only Dimer Descriptor

In this work, we constructed two models; one includes only the dimer descriptors and the second consists of the dimer and trimer descriptors. This section describes the total force, including only pair contributions.

According to eq. 62, the quantity $f(x)$, where x is a scalar quantity, represents the value of the pair force. Using the eq.62 and eq.78, the pair force is equal to the derivative of the W function.

$$\hat{f}(x) = \frac{dW_{dimer}(x)}{dx} \text{ where } x = D(\mathbf{Q}_j, \mathbf{Q}_k) = |\mathbf{Q}_j - \mathbf{Q}_k|, \quad j, k = 1, 2, \ldots, M \tag{80}$$

The goal in an active learning setting is to learn an unknown function as accurately and quickly as possible. According to 3.2.4 theory, making predictions involves only a finite sum over all past observations, i.e., all particle pairs and system configurations. Such an approach is computationally expensive. Thus, we choose a sparser model with far fewer kernel basis functions than the input data points where the locations of these basis functions (which we call the representative set) need not coincide with the input data locations. Thus,

$$\hat{f}(|\mathbf{Q}_j - \mathbf{Q}_k|) \overset{eq(62)}{=} \sum_{i}^{n_p} \hat{\alpha}_i \hat{k}(x_i, |\mathbf{Q}_j - \mathbf{Q}_k|) \tag{81}$$

where n_p is the number of basis points and x_i takes values of a grid, specifically from the minimum distance between two CG particles $r_{min} - \epsilon$ to r_{cut}, where $\epsilon > 0$. The kernel function of including only the dimer descriptor has the following form:

$$\hat{k}(x_i, x) = \hat{\delta}^2 \exp\left(-\frac{|x - x_i|^2}{2\hat{\theta}^2}\right). \tag{82}$$

The utility function that is used to optimize a_i parameters is based on the forces involving the dimer descriptors D representing the pair interactions, $\mathbf{F}_{CG,i}(\mathcal{Q}; \mathbf{a}) \approx \hat{\mathbf{F}}_i(\mathcal{Q}; \hat{\mathbf{a}})$ according to eq. 74:

$$\begin{aligned}
\hat{\mathbf{F}}_i(\mathcal{Q}; \hat{\mathbf{a}}) = -\nabla_{\mathbf{Q}_i} U_{CG}(\mathcal{Q}; \hat{\mathbf{a}}) &= -\nabla_{\mathbf{Q}_i}\Big(\sum_{j=1}^{M} \sum_{k=j+1}^{M} W_{dimer}(D(\mathbf{Q}_j, \mathbf{Q}_k))\Big) \\
&= -\sum_{j=1}^{M} \sum_{k=j+1}^{M} \frac{dW_{dimer}(D(\mathbf{Q}_j, \mathbf{Q}_k))}{dD(\mathbf{Q}_j, \mathbf{Q}_k)} \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) \\
&= -\sum_{j=1}^{M} \sum_{k=j+1}^{M} \hat{f}(|\mathbf{Q}_j - \mathbf{Q}_k|) \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) \\
&= -\sum_{j=1}^{M} \sum_{k=j+1}^{M} \sum_{l=1}^{n_p} \hat{a}_l \hat{k}(x_l, |\mathbf{Q}_j - \mathbf{Q}_k|) \nabla_{\mathbf{Q}_i}(|\mathbf{Q}_j - \mathbf{Q}_k|)
\end{aligned} \tag{83}$$

where $\hat{\mathbf{F}}_i$ is the total force exerted on a particle and the gradient of each descriptor $D(\mathbf{Q}_j, \mathbf{Q}_k) = |\mathbf{Q}_j - \mathbf{Q}_k|$ is calculated by

- $i = k \rightarrow \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) = \nabla_{\mathbf{Q}_i}(|\mathbf{Q}_j - \mathbf{Q}_i|) = -\frac{\mathbf{Q}_j - \mathbf{Q}_i}{|\mathbf{Q}_j - \mathbf{Q}_i|}$
- $i = j \rightarrow \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) = \nabla_{\mathbf{Q}_j}(|\mathbf{Q}_j - \mathbf{Q}_i|) = \frac{\mathbf{Q}_j - \mathbf{Q}_i}{|\mathbf{Q}_j - \mathbf{Q}_i|}$
- $i \neq k, j \rightarrow \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) = (0, 0, 0)$

Thus, the final expression of the force exerted on a particle for one sample is

$$\mathbf{F}_{CG,I}(\mathcal{Q}; \mathbf{a}) \approx \hat{\mathbf{F}}_{CG,I}(\mathcal{Q}; \hat{\mathbf{a}}) = -\sum_{j<k}^{M} \sum_{i=1}^{n_p} \hat{a}_i \hat{k}(x_i, r_{jk}) \nabla_{\mathbf{Q}_I} r_{jk}. \tag{84}$$

For the minimization problem, we need to solve a linear equation. Specifically, the analytical form of the minimization problem is given by

$$\min_{\hat{\mathbf{a}}} \frac{1}{n_s} \sum_{l=1}^{n_s} \sum_{I=1}^{M} ||\mathbf{F}(\mathbf{Q}_I^l) - \sum_{j<k}^{M} \sum_{i=1}^{n_p} \hat{a}_i k(x_i, |\mathbf{Q}_j^l - \mathbf{Q}_k^l|)(-\nabla_{\mathbf{Q}_I} D(\mathbf{Q}_j^l, \mathbf{Q}_k^l))||_3^2 \tag{85}$$

where $\mathbf{Q}_I^l = (\Pi \mathbf{q}^l)_I$ and the term $\sum_{j<k}^{M} \sum_{i=1}^{n_p} a_i \hat{k}(x_i, r_{jk})(-\nabla_{\mathbf{Q}_I} r_{jk})$ corresponds to the linear term $\hat{\mathbf{K}}\hat{\mathbf{a}}$.

According to the expression

$$\mathbf{y} = \hat{\mathbf{K}}\hat{\mathbf{a}}, \tag{86}$$

the $\mathbf{y} \in \mathbb{R}^{3M \cdot n_s}$ denotes the target forces of each CG particle in all three dimensions and for all the configurations of the model. The second term $\hat{\mathbf{K}}\hat{\mathbf{a}}$ approximates the $\mathbf{y}$ dataset corresponding to the dimer Kernel matrix and the optimization parameters.

$$\mathbf{y} = \begin{bmatrix} (Fo\Pi)_{1,x}(\mathbf{q}^1) \\ (Fo\Pi)_{1,y}(\mathbf{q}^1) \\ (Fo\Pi)_{1,z}(\mathbf{q}^1) \\ \vdots \\ (Fo\Pi)_{M,z}(\mathbf{q}^1) \\ (Fo\Pi)_{1,x}(\mathbf{q}^2) \\ \vdots \\ \vdots \\ (Fo\Pi)_{M,z}(\mathbf{q}^{n_s}) \end{bmatrix} = \begin{bmatrix} F_{1,x}(\mathbf{Q}^1) \\ F_{1,y}(\mathbf{Q}^1) \\ F_{1,z}(\mathbf{Q}^1) \\ \vdots \\ F_{M,z}(\mathbf{Q}^1) \\ F_{1,x}(\mathbf{Q}^2) \\ \vdots \\ \vdots \\ F_{M,z}(\mathbf{Q}^{n_s}) \end{bmatrix} \tag{87}$$

The parameter $\hat{\mathbf{a}}$ is the vector with length n_p, $\hat{\mathbf{a}} \in \mathbb{R}^{n_p}$, and the $\hat{\mathbf{K}}$ matrix, including the values of the distance gradient has size $(3M \cdot n_s) \times n_p$, $\quad \hat{\mathbf{K}} \in \mathbb{R}^{(3M \cdot n_s) \times n_p}$.

$$\hat{\mathbf{a}} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_{n_p} \end{bmatrix} \quad \nabla_{\mathbf{Q}_i} r_{jk} = \begin{pmatrix} \frac{dr_{jk}}{dQ_{i,x}} \\ \frac{dr_{jk}}{dQ_{i,y}} \\ \frac{dr_{jk}}{dQ_{i,z}} \end{pmatrix} \quad r_{jk} = |\mathbf{Q}_j - \mathbf{Q}_k| \tag{88}$$

And

$$\hat{\mathbf{K}} = \begin{bmatrix} \sum_{j<k}^{M} \hat{k}(x_1, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,x}^{(1)}}) & \sum_{j<k}^{M} \hat{k}(x_2, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,x}^{(l)}}) & \cdots & \sum_{j<k}^{M} \hat{k}(x_{n_p}, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,x}^{(l)}}) \\ \sum_{j<k}^{M} \hat{k}(x_1, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,y}^{(1)}}) & \sum_{j<k}^{M} \hat{k}(x_2, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,y}^{(l)}}) & \cdots & \sum_{j<k}^{M} \hat{k}(x_{n_p}, r_{jk}^{(1)})(-\frac{dr_{jk}^{(1)}}{d\mathbf{Q}_{1,y}^{(l)}}) \\ \vdots & \ddots & \ddots & \vdots \\ \sum_{j<k}^{M} \hat{k}(x_1, r_{jk}^{(l)})(-\frac{dr_{jk}^{(l)}}{d\mathbf{Q}_{I,\beta}^{(l)}}) & \ddots & \ddots & \sum_{j<k}^{M} \hat{k}(x_{n_p}, r_{jk}^{(l)})(-\frac{dr_{jk}^{(l)}}{d\mathbf{Q}_{I,\beta}^{(l)}}) \\ \vdots & \ddots & \ddots & \vdots \\ \sum_{j<k}^{M} \hat{k}(x_1, r_{jk}^{(n_s)})(-\frac{dr_{jk}^{(n_s)}}{d\mathbf{Q}_{M,z}^{(n_s)}}) & \sum_{j<k}^{M} \hat{k}(x_2, r_{jk}^{(n_s)})(-\frac{dr_{jk}^{(n_s)}}{d\mathbf{Q}_{M,z}^{(n_s)}}) & \cdots & \sum_{j<k}^{M} \hat{k}(x_{n_p}, r_{jk}^{(n_s)})(-\frac{dr_{jk}^{(n_s)}}{d\mathbf{Q}_{M,z}^{(n_s)}}) \end{bmatrix} \tag{89}$$

4.3 Application of Gaussian Process Regression - Dimer and Trimer Descriptors including

Here, we present a model where dimer and trimer descriptors are included. There are two forms for the quantity $f(x)$ (eq. 62), one function refers to the approximation of the total force only with the description of the dimer and has the well-known form as $\hat{k}(x, x_i) = \hat{\delta}^2 \exp\Big(-\frac{|x-x_i|^2}{2\hat{\theta}^2}\Big)$, with scalar x, x_i, and the other one refers to the trimer descriptor Ref.[20].

$$k^*(\mathbf{x}, \mathbf{x}') = \delta_*^2 \exp\big[\sum_{i=1}^{3}(-\frac{|x_i - x_i'|^2}{2(\theta_*^i)^2})\big] \tag{90}$$

where $\mathbf{x}, \mathbf{x}', \theta_*$ are vectors, $\mathbf{x}$ is the three-body descriptor and $\mathbf{x}'$ denote the basis points vector. Also, the $\theta_* = [\theta_*^1, \theta_*^2, \theta_*^3]$ is a vector and δ_* a scalar quantity. The elements of the symmetrized three-body descriptor $\mathbf{x} = [d_1, d_2, d_3]$ are defined as

$$d_1 = r_{ij} + r_{ik}, \quad d_2 = (r_{ij} - r_{ik})^2 \text{ and } d_3 = r_{jk}, \tag{91}$$

which ensures invariance to rigid rotations and translations as well as swapping the indices of j and k, while providing a bijective mapping between atomic coordinates and the descriptor elements.

The basis point $\mathbf{x}' = [x_1', x_2', x_3']$ is a 3d vector, and each element takes values of a different grid. Specifically, x_1' takes values from the $2r_{min} - \epsilon$ distance to $2r_{cut}$, where r_{min} is the minimum distance between two CG particles and $\epsilon > 0$. The second element x_2' takes values from 0 to $(r_{cut} - r_{min})^2$, and the third element x_3' from $r_{min} - \epsilon$ to r_{cut}. All grids have the same size.

Using the eq.62, eq.90 and eq.78, the total force has two terms, the dimer and the trimer contributions, and the derivative of the functions W_{dimer} and W_{trimer} function have the following forms

$$\hat{f}(x) = \frac{dW_{dimer}(x)}{dx} \text{ where } x = D(\mathbf{Q}_j, \mathbf{Q}_k) = |\mathbf{Q}_j - \mathbf{Q}_k|, \text{ but} \tag{92}$$

$$f^*(\mathbf{x}) = W_{trimer}(\mathbf{x}) \text{ where } \mathbf{x} = D(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l) = (r_{jk}+r_{jl}, (r_{jk}-r_{jl})^2, r_{kl}) \text{ and } r_{jk} = D(\mathbf{Q}_j, \mathbf{Q}_k) \tag{93}$$

Also here, we will choose a sparser model with far fewer kernel basis functions. Thus,

$$f^*(D(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)) = \sum_{I}^{\mathbf{n_p}^*} \alpha_I^* k^*(\mathbf{x}_I, D(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)) \tag{94}$$

where $\mathbf{n_p}^* = (\cdot, \cdot, \cdot)$ is a vector with the number of basis points in each dimension and $\mathbf{x}_I$ is a vector and each element takes values of a different grid. I is a multi-index $I = (i, j, k)$, $i, j, k = 1, \ldots, n_p^*$ and takes $n_p^* = \#\mathbf{n_p}^*$ different values in total, e.g. if we choose 3 basis points for each element, then the multi-index I takes $n_p^* = 27$ values: $\mathbf{n_p}^* = (1,1,1), (1,1,2), \cdots, (1,3,3), (2,1,1), \cdots, (3,3,3)$.

The utility function that is used to optimize $\hat{a}_i, a_I^*$ parameters can be based on the forces based on the dimer and the trimer contributions. The dimer term is calculated by the eq.83. The corresponding trimer term can be approximated by

$$\begin{aligned}
\mathbf{F}_i^*(\mathcal{Q};\mathbf{a}^*) &= -\nabla_{\mathbf{Q}_i} U_{CG}(\mathcal{Q};\mathbf{a}^*) \\
&= -\nabla_{\mathbf{Q}_i}(\sum_{j=1}^{M}\sum_{k=j+1}^{M}\sum_{l=k+1}^{M} W_{trimer}(D(\mathbf{Q}_j,\mathbf{Q}_k,\mathbf{Q}_l))) \\
&\overset{eq.91}{=} -\nabla_{\mathbf{Q}_i}\Big[\sum_{j<k<l}^{M} W_{trimer}(d_1(\mathbf{Q}_j,\mathbf{Q}_k,\mathbf{Q}_l), d_2(\mathbf{Q}_j,\mathbf{Q}_k,\mathbf{Q}_l), d_3(\mathbf{Q}_j,\mathbf{Q}_k,\mathbf{Q}_l))\Big] \\
&= -\sum_{j<k<l}^{M}\Big[\frac{\partial W_{tr}(d_1,d_2,d_3)}{\partial d_1}\nabla_{\mathbf{Q}_i} d_1(\cdot) + \frac{\partial W_{tr}(d_1,d_2,d_3)}{\partial d_2}\nabla_{\mathbf{Q}_i} d_2(\cdot) + \frac{\partial W_{tr}(d_1,d_2,d_3)}{\partial d_3}\nabla_{\mathbf{Q}_i} d_3(\cdot)\Big] \\
&\overset{eq.93}{=} -\sum_{j<k<l}^{M}\sum_{I}^{\mathbf{n_p}^*}\alpha_I^*\Big[\frac{\partial k^*(\mathbf{x}_I,(d_1,d_2,d_3))}{\partial d_1}\nabla_{\mathbf{Q}_i} d_1(\cdot) + \frac{\partial k^*(\mathbf{x}_I,(d_1,d_2,d_3))}{\partial d_2}\nabla_{\mathbf{Q}_i} d_2(\cdot) \\
&\quad + \frac{\partial k^*(\mathbf{x}_I,(d_1,d_2,d_3))}{\partial d_3}\nabla_{\mathbf{Q}_i} d_3(\cdot)\Big] \\
&= -\sum_{j<k<l}^{M}\sum_{I}^{\mathbf{n_p}^*}\alpha_I^*\sum_{z}^{3}\frac{\partial k^*(\mathbf{x}_I,(d_1,d_2,d_3))}{\partial d_z}\nabla_{\mathbf{Q}_i} d_z(\cdot)
\end{aligned} \tag{95}$$

where $\mathbf{F}_i(\mathcal{Q};\hat{\mathbf{a}},\mathbf{a}^*) = \hat{\mathbf{F}}_i(\mathcal{Q};\hat{\mathbf{a}}) + \mathbf{F}_i^*(\mathcal{Q};\mathbf{a}^*)$ is the total force exerted on a particle. Analytical calculation of the above equations can be found in Appendix B.

Thus, the final expression of the force exerted on a particle for one sample is

$$\begin{aligned}
\mathbf{F}_{CG,I}(\mathcal{Q};\hat{\mathbf{a}},\mathbf{a}^*) &= \hat{\mathbf{F}}_i(\mathcal{Q};\hat{\mathbf{a}}) + \mathbf{F}_i^*(\mathcal{Q};\mathbf{a}^*) \\
&= -\sum_{j<k}^{M}\sum_{i=1}^{n_p}\hat{a}_i\hat{k}(x_i, D(\mathbf{Q}_j,\mathbf{Q}_k))\nabla_{\mathbf{Q}_I} D(\mathbf{Q}_j,\mathbf{Q}_k) \\
&\quad -\sum_{j<k<l}^{M}\sum_{I}^{\mathbf{n_p}^*}\alpha_I^*\sum_{z}^{3}\frac{\partial k^*(\mathbf{x}_I,(d_1(\cdot),d_2(\cdot),d_3(\cdot)))}{\partial d_z(\cdot)}\nabla_{\mathbf{Q}_i} d_z(\mathbf{Q}_j,\mathbf{Q}_k,\mathbf{Q}_l)
\end{aligned} \tag{96}$$

In this case, the minimization problem has the analytical form

$$\begin{aligned}
\min_{\hat{\mathbf{a}},\mathbf{a}^*}\frac{1}{n_s}\sum_{s=1}^{n_s}\sum_{I=1}^{M}||(\mathbf{F}(\mathbf{Q}_I^s) &- \sum_{j<k}^{M}\sum_{i=1}^{n_p}\hat{a}_i k(x_i, r_{jk}^s)(-\nabla_{\mathbf{Q}_I} r_{jk}^s) \\
&- \sum_{j<k<l}^{M}\sum_{\mathbf{i}}^{\mathbf{n_p}^*}\alpha_{\mathbf{i}}^*\sum_{z}^{3}\frac{\partial k^*(\mathbf{x_i},(d_1(\cdot),d_2(\cdot),d_3(\cdot)))}{\partial d_z(\cdot)}(-\nabla_{\mathbf{Q}_I} d_z(\mathbf{Q}_j^s,\mathbf{Q}_k^s,\mathbf{Q}_l^s))||_3^2
\end{aligned} \tag{97}$$

where $\mathbf{Q}_I^s = (\Pi \mathbf{q}^s)_I$, the second term is equal to the linear form $\hat{\mathbf{K}}\hat{\mathbf{a}}$, and the third term corresponds to the linear form $\mathbf{K}^*\mathbf{a}^*$.

The equation

$$\mathbf{y} = \mathbf{K}\mathbf{a}, \quad \mathbf{K} = [\hat{\mathbf{K}} \quad \mathbf{K}^*] \text{ and } \mathbf{a} = \begin{bmatrix} \hat{\mathbf{a}} \\ \mathbf{a}^* \end{bmatrix} \tag{98}$$

denotes the forces of each CG particle in all three dimensions and for all the configurations of the model $\mathbf{y} \in \mathbb{R}^{3M \cdot n_s}$. The vector $\mathbf{y}$ has the form as in eq.87.

The parameter $\mathbf{a}$ is the vector with length $n_p + {n_p}^*$, $\mathbf{a} \in \mathbb{R}^{n_p + {n_p}^*}$, and the $\mathbf{K}$ matrix has size $(3M \cdot n_s) \times (n_p + {n_p}^*)$, $\mathbf{K} \in \mathbb{R}^{(3M \cdot n_s) \times (n_p + {n_p}^*)}$.

$$\mathbf{a} = \begin{bmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_{n_p} \\ a_1^* \\ \vdots \\ a_{n_p^*}^* \end{bmatrix} \quad \text{and} \tag{99}$$

$$\nabla_{\mathbf{Q}_i} r_{jk} = \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j, \mathbf{Q}_k) = \begin{pmatrix} \frac{\partial D(\mathbf{Q}_j, \mathbf{Q}_k)}{\partial Q_{i,x}} \\ \frac{\partial D(\mathbf{Q}_j, \mathbf{Q}_k)}{\partial Q_{i,y}} \\ \frac{\partial D(\mathbf{Q}_j, \mathbf{Q}_k)}{\partial Q_{i,z}} \end{pmatrix}, \quad \nabla_{\mathbf{Q}_I} d_z(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l) = \begin{pmatrix} \frac{\partial d_z(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)}{\partial Q_{i,x}} \\ \frac{\partial d_z(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)}{\partial Q_{i,y}} \\ \frac{\partial d_z(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)}{\partial Q_{i,z}} \end{pmatrix} \tag{100}$$

where $\frac{\partial D(\mathbf{Q}_j, \mathbf{Q}_k)}{\partial Q_{i,x}} \in \mathbb{R}$, $\frac{\partial d_z(\mathbf{Q}_j, \mathbf{Q}_k, \mathbf{Q}_l)}{\partial Q_{i,x}} \in \mathbb{R}^3$. For the kernel matrix $\mathbf{K}$, the dimer kernel matrix $\hat{\mathbf{K}}$ has been calculated in eq.89,thus the trimer kernel matrix $\mathbf{K}^*$ has the following form:

$$\mathbf{K}^* = - \begin{bmatrix} \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_1, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^1_z} \nabla_{\mathbf{Q}^1_{1,x}} d^1_z(\cdot) & \cdots & \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_{n_p^*}, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^1_z} \nabla_{\mathbf{Q}^1_{1,x}} d^1_z(\cdot) \\ \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_1, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^1_z} \nabla_{\mathbf{Q}^1_{1,y}} d^1_z(\cdot) & \cdots & \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_{n_p^*}, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^1_z} \nabla_{\mathbf{Q}^1_{1,y}} d^1_z(\cdot) \\ \vdots & \ddots & \vdots \\ \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_1, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^s_z} \nabla_{\mathbf{Q}^I_{1,z}} d^1_z(\cdot) & \ddots & \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_{n_p^*}, (d^1_{1,m}, d^1_{2,m}, d^1_{3,m}))}{\partial d^s_z} \nabla_{\mathbf{Q}^I_{1,z}} d^1_z(\cdot) \\ \vdots & \ddots & \vdots \\ \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_1, (d^s_1, d^s_{2,m}, d^s_{3,m}))}{\partial d^s_z} \nabla_{\mathbf{Q}^1_{I,x}} d^I_z(\cdot) & \ddots & \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_{n_p^*}, (d^s_1, d^s_{2,m}, d^s_{3,m}))}{\partial d^s_z} \nabla_{\mathbf{Q}^1_{I,x}} d^I_z(\cdot) \\ \vdots & \ddots & \vdots \\ \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_1, (d^{n_s}_{1,m}, d^{n_s}_{2,m}, d^{n_s}_{3,m}))}{\partial d^{n_s}_z} \nabla_{\mathbf{Q}^{n_s}_{M,z}} d^{n_s}_z(\cdot) & \cdots & \sum_{j<k<l}^{M} \sum_z^3 \frac{\partial k^*(\mathbf{x}_{n_p^*}, (d^{n_s}_{1,m}, d^{n_s}_{2,m}, d^{n_s}_{3,m}))}{\partial d^{n_s}_z} \nabla_{\mathbf{Q}^{n_s}_{M,z}} d^{n_s}_z(\cdot) \end{bmatrix} \tag{101}$$

4.4 The normal equations

According to eq. 85 and eq. 97, a linear problem arises. The quantity $\mathbf{F}_{CG,I}(\mathcal{Q}; \mathbf{a}) = \mathbf{F}$ from eq.75 is described as

$$\mathbf{F} = \mathbf{K} \cdot \mathbf{a} \tag{102}$$

Use normal equations $\mathbf{A}^{tr}\mathbf{A}\mathbf{x} = A^{tr}\mathbf{b}$ to solve eq.86, where $\mathbf{b} = \mathbf{F}$, $\mathbf{A} = \mathbf{K}$ and $\mathbf{x} = \mathbf{a}$.

In general, we want to minimize $g(\mathbf{x}) = ||\mathbf{b} - \mathbf{A}\mathbf{x}||_2^2$, which corresponds to the eq. 73, where b is equivalent to the term $(\mathbf{F}o\Pi)(\cdot)$ and $\mathbf{A}\mathbf{x}$ to $\mathbf{F}_{CG}(\cdot; \mathbf{a})$.

$$g(\mathbf{x}) = ||\mathbf{b} - \mathbf{A}\mathbf{x}||_2^2 = (\mathbf{b} - \mathbf{A}\mathbf{x})^{tr}(\mathbf{b} - \mathbf{A}\mathbf{x}) = \mathbf{b}^{tr}\mathbf{b} - \mathbf{x}^{tr}\mathbf{A}^{tr}\mathbf{b} - \mathbf{b}^{tr}\mathbf{A}\mathbf{x} + \mathbf{x}^{tr}\mathbf{A}^{tr}\mathbf{A}\mathbf{x} \quad (103)$$

If $\mathbf{x}$ is a global minimum of g, then its gradient $\boldsymbol{\nabla} g(\mathbf{x})$ is the zero vector. Let's take the gradient of g remembering that

$$\boldsymbol{\nabla} g(\mathbf{x}) = \begin{pmatrix} \frac{\partial g}{\partial x_1} \\ \frac{\partial g}{\partial x_2} \\ \vdots \\ \frac{\partial g}{\partial x_{n_p}} \end{pmatrix}, \quad \boldsymbol{\nabla}\big(\mathbf{x}^{tr}\mathbf{A}^{tr}\mathbf{b}\big) = \mathbf{A}^{tr}\mathbf{b} \quad \boldsymbol{\nabla}\big(\mathbf{b}^{tr}\mathbf{A}\mathbf{x}\big) = \mathbf{A}^{tr}\mathbf{b} \quad \boldsymbol{\nabla}\big(\mathbf{x}^{tr}\mathbf{A}^{tr}\big)\mathbf{A}\mathbf{x} = 2\mathbf{A}^{tr}\mathbf{A}\mathbf{x}$$

Thus,

$$\begin{aligned} &\boldsymbol{\nabla} g(\mathbf{x}) = 2\mathbf{A}^{tr}\mathbf{A}\mathbf{x} - 2\mathbf{A}^{tr}\mathbf{b} = 0 \\ &\mathbf{A}^{tr}\mathbf{A}\mathbf{x} = \mathbf{A}^{tr}\mathbf{b} \end{aligned} \quad (104)$$

At this point, we present two techniques to solve the linear problem to avoid overfitting. See also the discussion in section 3.3, where we relate the Gaussian process regression to the KRR applied here.

The regularization parameter

According to the eq.64, we add the regularization parameter that can be equal to a scalar parameter λ with the identity matrix, i.e.

$$(\mathbf{A}^{tr}\mathbf{A} + \lambda\mathbf{I})\mathbf{x} = \mathbf{A}^{tr}\mathbf{b}$$

For our system, we solve the normal equation

$$(\mathbf{K}^{tr} \cdot \mathbf{K} + \lambda\mathbf{I}) \cdot \mathbf{a} = \mathbf{K}^{tr} \cdot \mathbf{F} \quad (105)$$

The regularization matrix

In this case, for the regularization parameter, we choose a different matrix,

$$(\mathbf{A}^{tr}\mathbf{A} + \lambda\mathbf{R})\mathbf{x} = \mathbf{A}^{tr}\mathbf{b}$$

where $\mathbf{R} = \sum_{n,n'}^{n_p} a_n k(x_n, x_{n'}) a_{n'}$ and $x_n, x_{n'}$ take value of the basis points. Thus, for our system, we solve the normal equation

$$(\mathbf{K}^{tr} \cdot \mathbf{K} + \lambda\mathbf{R}) \cdot \mathbf{a} = \mathbf{K}^{tr} \cdot \mathbf{F} \quad (106)$$

4.5 Hyperparameters tuning

The optimal forecast of a Gaussian process model depends on the selection of the hyperparameters on the kernel function. There are various techniques in order to learn hyperparameters. There are three approaches to learning the hyperparameters: (a) a grid search, (b) maximizing the likelihood, and (c) with a Bayesian approach. For our project, we applied the grid search. Although, we will present the theoretical part of the likelihood technique.

The most straightforward technique suggests maximizing the log-likelihood function $p(\mathbf{t}|\theta)$, where θ denotes the set of hyperparameters. An efficient way to maximize the log-likelihood function with respect to θ, includes the implementation of gradient-based optimization algorithms such as conjugate gradients.

The log-likelihood function for a Gaussian process regression model is given by

$$\ln p(\mathbf{t}|\theta) = -\frac{1}{2} ln|\mathbf{C}_{n_s}| - \frac{1}{2}\mathbf{t}^{tr}\mathbf{C}_{n_s}^{-1}\mathbf{t} - \frac{n_s}{2}\ln(2\pi).$$

For nonlinear optimization, we also use the gradient of the log-likelihood function with respect to θ

$$\frac{\partial}{\partial\theta_i}\ln p(\mathbf{t}|\theta) = -\frac{1}{2}\,\mathrm{Tr}\left(\mathbf{C}_{n_s}^{-1}\frac{\partial\mathbf{C}_{n_s}}{\partial\theta_i}\right) + \frac{1}{2}\mathbf{t}^{tr}\mathbf{C}_{n_s}^{-1}\frac{\partial\mathbf{C}_{n_s}}{\partial\theta_i}\mathbf{t},$$

where $\ln p(\mathbf{t}|\theta)$ is a nonconvex function, and thus there can be multiple maxima.

The grid search method is not the optimal choice because it may be computationally more costly than the other choices. Although, it is straightforward and easy to implement. Grid search builds a model for every combination of hyperparameters specified and evaluates each model. Because the value of the hyperparameter has to be set before the learning process begins, we choose the following parameters for our model:

For the model with only the dimer descriptor:

- Number of basis points: 15, 20, 48.
- Parameter δ : 1.0, 2.0, 6.0.
- Parameter θ : 2.5, 3.5, 5.0.
- Parameter λ : 0.0, 0.1, 0.01, 0.001.

4.6 Other representations of the pair potential

Lennard-Jones Model

The Lennard-Jones potential models soft repulsive and attractive interactions. This is a straightforward, parametric way to approximate the total force of the system.

$$u_{LJ}(x) = 4\epsilon\Big[\left(\frac{\sigma}{x}\right)^{12} - \left(\frac{\sigma}{x}\right)^{6}\Big] = 4\Big[\left(\frac{A}{x}\right)^{12} - \left(\frac{B}{x}\right)^{6}\Big]. \tag{107}$$

The total force for i-th particle is computed by

$$\begin{aligned}\mathbf{F}_i &= -\nabla_{\mathbf{Q}_i} U_{CG}(\mathcal{Q};\mathbf{a}) \\ &= -\sum_{j=1}^{M}\sum_{k=j+1}^{M} \nabla_{\mathbf{Q}_i} u_{LJ}(D(\mathbf{Q}_j,\mathbf{Q}_k);\mathbf{a}) \\ &= -\sum_{j=1}^{M}\sum_{k=j+1}^{M} \frac{\mathrm{d}u_{LJ}(D(\mathbf{Q}_j,\mathbf{Q}_k);\mathbf{a})}{\mathrm{d}D} \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j,\mathbf{Q}_k)\end{aligned} \tag{108}$$

where here the parameters $\mathbf{a}$ are the LJ parameters A, B, i.e. $\mathbf{a} = (A, B)$ and $\mathbf{F}_i \in \mathbb{R}^3$.

We calculate the term $\frac{\mathrm{d}u_{LJ}(D(\mathbf{Q}_j,\mathbf{Q}_k);\mathbf{a})}{\mathrm{d}D}$

$$\begin{aligned}\frac{\mathrm{d}u_{LJ}(D(\mathbf{Q}_j,\mathbf{Q}_k);\mathbf{a})}{\mathrm{d}D} &= 4\frac{\mathrm{d}\Big[(\frac{A}{D})^{12} - (\frac{B}{D})^{6}\Big]}{\mathrm{d}D} \\ &= \begin{cases} 4\Big[-\frac{12A^{12}}{D^{13}} + \frac{6B^6}{D^7}\Big] & \text{if } D \le r_{cut} \\ 0 & \text{if } D > r_{cut} \end{cases}.\end{aligned} \tag{109}$$

and also, the term $\nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j,\mathbf{Q}_k) = (\frac{\partial D(\mathbf{Q}_j,\mathbf{Q}_k)}{\partial Q_{i,x}}, \frac{\partial D(\mathbf{Q}_j,\mathbf{Q}_k)}{\partial Q_{i,y}}, \frac{\partial D(\mathbf{Q}_j,\mathbf{Q}_k)}{\partial Q_{i,z}})$.

Linear splines Basis Model

A simple linear model for regression leads to spline functions ref. [31]. A spline model is referred to as a representation of the pair potential $u(x)$ with piecewise linear functions,

$$g_i(x) = \begin{cases} \frac{x - x_{i-1}}{\Delta x} & x_{i-1} < x \le x_i \\ \frac{x_{i+1} - x}{\Delta x} & x_i < x \le x_{i+1} \end{cases} \tag{110}$$

where x_i is a basis point, $\Delta x = x_{i+1} - x_i$ is the distance between two consecutive basis points and x is the distance of two CG particles. That is

$$u(x) = \sum_{l}^{n_p} \alpha_l g_l(x).$$

The total force for i-th particle is computed by

$$\begin{aligned}\mathbf{F}_i &= -\nabla_{\mathbf{Q}_i} V(\mathcal{Q};\mathbf{a}) \\ &= -\sum_{j=1}^{M}\sum_{k=j+1}^{M} \nabla_{\mathbf{Q}_i} u(D(\mathbf{Q}_j,\mathbf{Q}_k);\mathbf{a}) \\ &= -\sum_{j=1}^{M}\sum_{k=j+1}^{M}\sum_{l}^{n_p} \alpha_l g_l(|\mathbf{Q}_j - \mathbf{Q}_k|) \nabla_{\mathbf{Q}_i} D(\mathbf{Q}_j,\mathbf{Q}_k)\end{aligned} \tag{111}$$

where here the parameters $\mathbf{a} = (a_1, a_2, \cdots, a_{n_p})$ are the minimization parameters.

4.7 Quality of the model

When we estimate a regression model, the differences between the actual and "predicted" values for the dependent variable can be measured by different measures. In this work, we report the chi-square error and a probability metric. There are a lot of metrics available to quantify the difference between two probability measures. In our case, we applied the Wasserstein metric, as it takes into account the metric space. In both cases, the metric's and Chi-Square distance meanings can interpret hard numbers, and both of them raise questions about our model's quality Ref. [18].

Wasserstein metric

Regarding the convergence of measures, a challenge is the choice of probability metric. In our case, it is very important to quantify that convergence in terms of some probability metric.

Let Ω denote a measurable space with σ-algebra $\mathcal{B}$. Let $\mathcal{M}$ be the set of all probability measures on $(\Omega, \mathcal{B})$. Let μ, ν denote two probability measures on Ω. Let h and g denote their corresponding density functions assuming they are defined. At Wasserstein metric, where $\Omega = \mathbb{R}$, let $\mathcal{H}, \mathcal{G}$ denote their corresponding distribution functions. Also, $\mathcal{X}, \mathcal{Y}$ will denote random variables on Ω such that $Law(\mathcal{X}) = \mu$ and $Law(\mathcal{Y}) = \nu$.

Definition of Wasserstein (or Kantorovich) metric

For $\Omega = \mathbb{R}$, if $\mathcal{H}, \mathcal{G}$ are the distribution functions of μ, ν, respectively, the Kantorovich metric is defined by

$$\begin{aligned} d_W(\mu, \nu) &:= \int_{-\infty}^{\infty} |\mathcal{H}(x) - \mathcal{G}(x)| dx \\ &= \int_0^1 |\mathcal{H}^{-1}(t) - \mathcal{G}^{-1}(t)| dt \end{aligned} \tag{112}$$

where $\mathcal{H}^{-1}, \mathcal{G}^{-1}(t)$ are the inverse functions of the distribution functions. If the metric space is separable, then the eq.112 is equivalent to

$$d_W(\mu, \nu) := \sup\{| \int z d\mu - \int z d\nu| : ||z||_L \leq 1\}, \tag{113}$$

the supremum being taken over all z satisfying the Lipschitz condition $|z(x) - z(y)| \leq d(x, y)$ where d is the metric on $\mathbb{R}$. The Wasserstein metric assumes values in $[0, diam(\Omega)]$, where $diam(\Omega)$ is the diameter of the metric space (Ω, d), and also measures weak convergence on spaces of bounded diameter.

Chi-squared error

The form of the χ^2-error is adjusted for the number of coordinates, samples, and particles. It is defined as:

$$\begin{aligned}\chi^2 &= \frac{1}{3 * M * n_s}||(\mathbf{F}o\Pi)(\cdot) - \mathbf{F}_{CG}(\cdot;\mathbf{a})||^2 \\ &= \frac{1}{3 * M * n_s}((\mathbf{F}o\Pi)(\cdot) - \mathbf{F}_{CG}(\cdot;\mathbf{a}))^{tr}((\mathbf{F}o\Pi)(\cdot) - \mathbf{F}_{CG}(\cdot;\mathbf{a}))\end{aligned} \tag{114}$$

where $(\mathbf{F}o\Pi)(\cdot)$ known from samples and the second term $\mathbf{F}_{CG}(\cdot;\mathbf{a})$ are the approximated values.

5 Simulation of methane system

In this section, we describe the bulk methane system, for which we learn the CG model with the GPR method. Specifically, we implement the GPR as the equivalent KRR problem. Next, we present the results for different values of the chosen kernel parameters. The results presented below explain how effective the KRR method is according to the pair forces' behavior.

5.1 Bulk methane system

To check how accurately the model can approximate the pair force of CG particles, we examine a more realistic system, of a bulk methane liquid. Methane is the simplest alkane and its chemical formula is CH_4.

Our system has 512 CH_4 methane molecules at the NVT ensemble at 100K temperature and its density was calculated after equilibrating the system in the NPT ensemble for $5ns$ ($\rho = 0.38g/cm^3$). The time step was $0.5fs$ and a cut-off distance of 10 Angstroms was used. The calculations have been performed using the all-atom Dreiding force field. For the coarse-grained representation of CH_4, we have used a one-site representation with a pair potential. We performed the various CG procedures using these trajectories. In total, 7400 frames (observations) were collected.

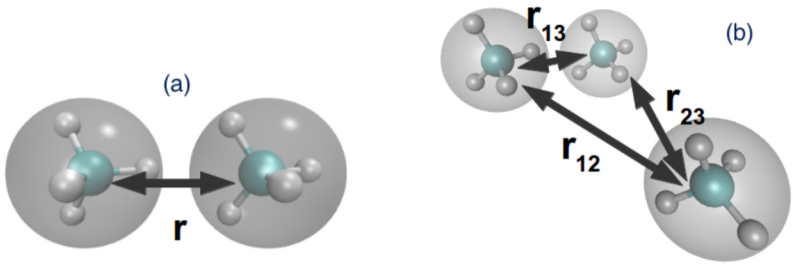


Figure 8: Example of two (a) and three (b) methane molecules in atomistic and coarse-grained description

5.2 Results with Dimer Descriptor only

In this section, we present some figures corresponding to the different simulations that we have executed. In fig.9 and fig.10, we present the models with the optimal combinations of the model parameters i.e. the number of grid points, the kernel parameters eq. 62, and the regularization parameter λ. The best models were derived by finding the minimum value of the chi-square error and the Wasserstein distance for the different forms of the regularization matrix. We also present the corresponding measurements of the LJ model and the linear-splines model for comparison.

The training of the CG methane model was performed for: three different choices for the number of basis points, three values for each parameter; δ, θ, and four values for the λ parameter. In total, there are 108 different trained models. Moreover, as we mentioned in section 4.4, we applied two regularization matrices: the first one consisted of the parameter λ and the identity matrix $\mathbf{I}$, and the second one of the parameter λ and the kernel matrix $\mathbf{R}$ of the basis points. Thus, the total number of different models has been summarized to 189: the models for $\lambda = 0$ in both techniques give us the same minimization function.

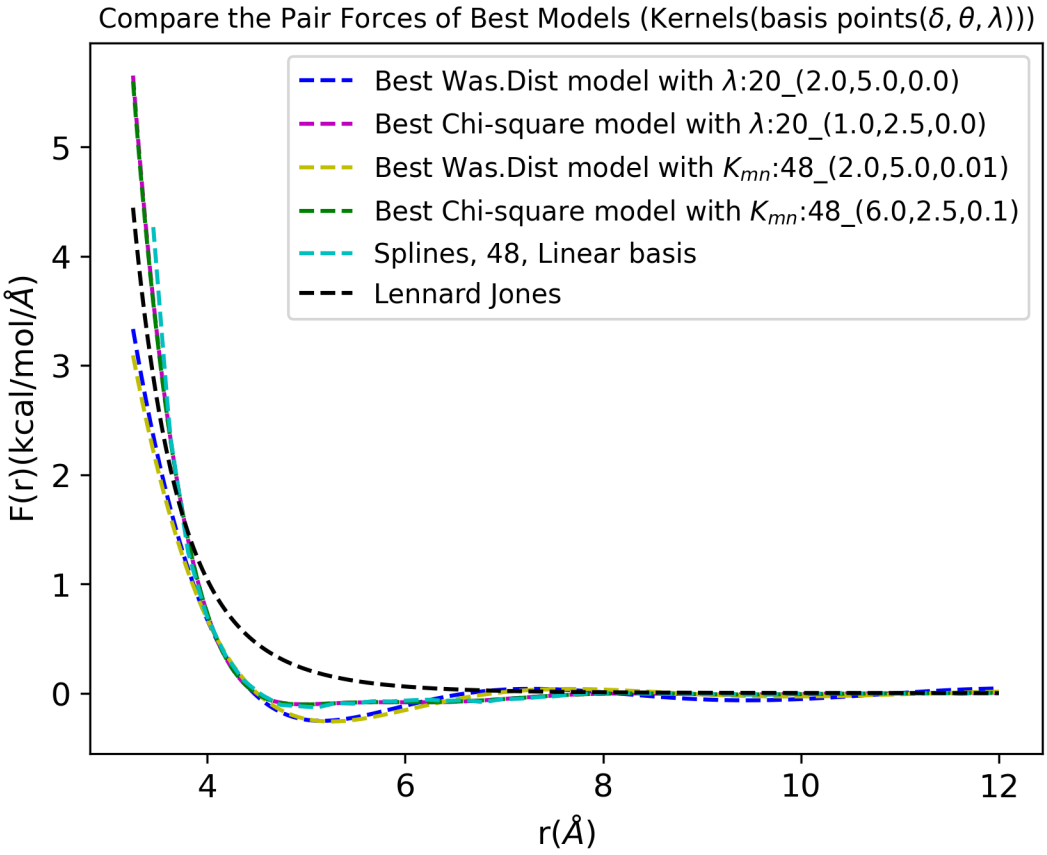


Figure 9: Pair Forces plot for the methane system (100 samples)

In fig. 9, the values of the pair forces of the model with splines and the two KRR models with the optimal Chi-square value give the same results. This is not surprising, if we consider that cost function is the same as the Chi-square measurement and the linear basis method has been proved to have a good fitting Ref. [2]. Only the results using the LJ basis slightly deviate.

Moreover, the above conclusions can be observed in fig. 10. In this figure, the derived potential, through numerical integration of the forces, is shown.

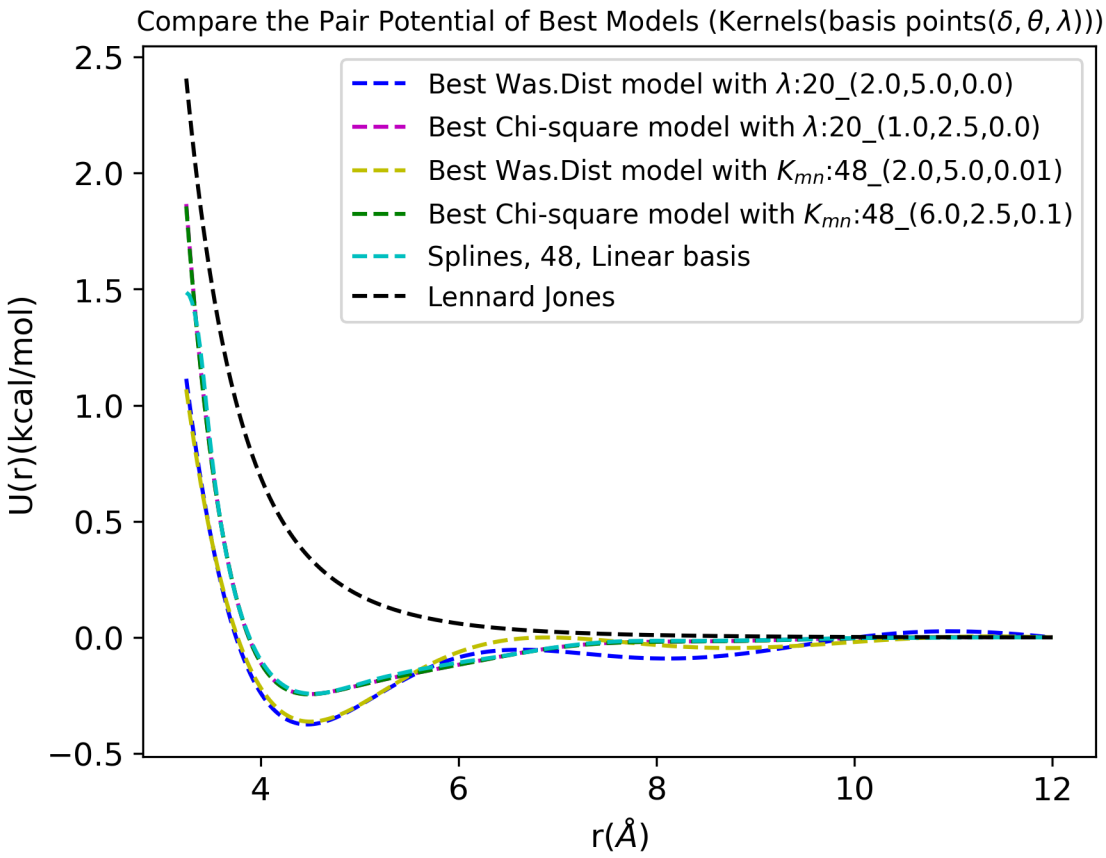


Figure 10: Potential plot according to the Pair Forces values for the methane system (100 samples)

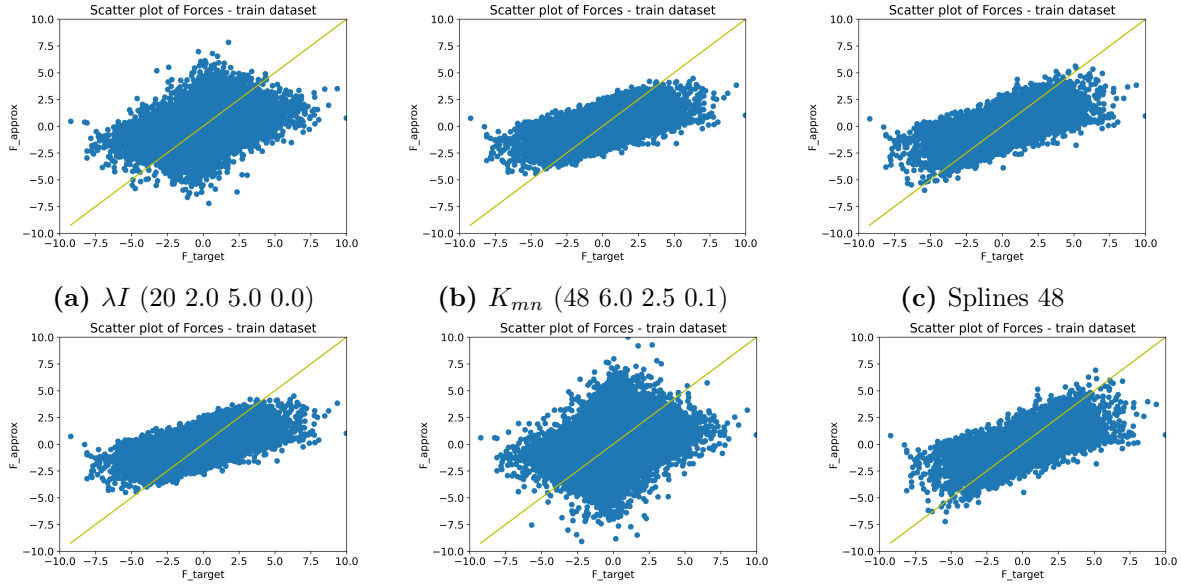
Note that for all models we considered the same data set. Initially, the first 100 samples were used, but 90% of this amount is referred to the training dataset and the remaining 10% is the test dataset. In figures 9 and 10, we notice by observing the values for small distances r of the pair force and the pair potential, that the values differ. This happens because in this region we have little data. What matters, in this case, is the minimum distance between two CG particles which is around 3.27.

Table 1 Models with minimum values of Chi-Square error and Wasserstein distance for 100 samples

(basis points, δ, θ, λ)	Wasserstein train	Wasserstein test	Chi-Square train	Chi-Square test
λI (20 2.0 5.0 0.0)	0.09	0.10	1.33	1.36
λI (20 1.0 2.5 0.0)	0.23	0.24	0.77	0.80
K_{mn} (48 2.0 5.0 0.01)	0.07	0.07	1.73	1.73
K_{mn} (48 6.0 2.5 0.1)	0.23	0.24	0.77	0.80
Splines 48	0.23	0.25	0.76	0.80
LJ	0.25	0.27	0.78	0.81

In the above table 1, we have the exact values of the two measurements for each of the models. We notice that among the best models, parameters and measure values do not

repeat. For the models with the minimum Wasserstein value (the first and third models), the Chi-square values seem to increase quite a bit, while the opposite does not seem to be the case. Of course, it can be explained as the training is based on the Chi-square error, not the Wasserstein metric. Also, the number of samples does not suffice to draw a valid conclusion.

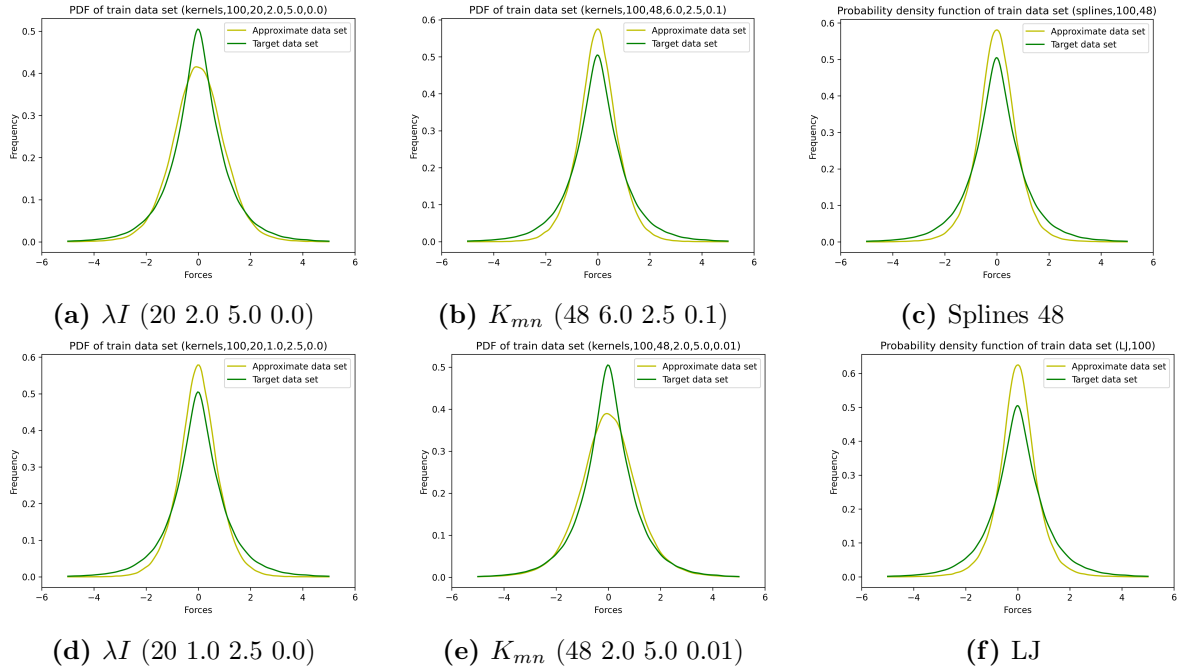


(a) λI (20 2.0 5.0 0.0) **(b)** K_{mn} (48 6.0 2.5 0.1) **(c)** Splines 48

(d) λI (20 1.0 2.5 0.0) **(e)** K_{mn} (48 2.0 5.0 0.01) **(f)** LJ

Figure 11: Scatter plots for 100 samples. The x-axis denotes the F_{target} and the y-axis the F_{approx}.

In fig.11, we present a scatter plot of the training dataset's forces. A scatter plot uses dots to present values for two different numeric variables: the target forces F_{target}, and the approximated forces according to the KRR method, the LJ method, and the linear basis method. The dots refer to all three dimensions F_x, F_y, F_z. Scatter plots are used to observe relationships between variables. In our case, the goal is for the dots to be closer to the trend line. This indicates a strong correlation relationship between the variables and a good fitting. According to the figures, the two models which F_{approx} seem to fail to approximate well, are the models that have the minimum value for the Wasserstein metric (λI (20 2.0 5.0 0.0), K_{mn} (48 2.0 5.0 0.01) - fig.11a and fig.11e).



(a) λI (20 2.0 5.0 0.0) **(b)** K_{mn} (48 6.0 2.5 0.1) **(c)** Splines 48

(d) λI (20 1.0 2.5 0.0) **(e)** K_{mn} (48 2.0 5.0 0.01) **(f)** LJ

Figure 12: Probability density function

According to the Wasserstein values at table 1, the models with parameters λI (20 2.0 5.0 0.0) and

K_{mn} (48 2.0 5.0 0.01) seem to have better results in terms of the probability density function. Although, this is not evident in the corresponding figures, 12a and 12e, because they do not differ from the rest probability density function (pdf) figures. All models' pdf of the approximate data is very close to the pdf of the target data but with larger differences around the mean value.

Even though 100 samples is a small number to draw valid conclusions, we will try to understand if there is any pattern to the parameters of the best models according to the two measurements. In the tables 2 and 3, we present the first five best models that were tested for two different regularization matrices. Specifically, table 2 depicts the models that give better results with respect to the Wasserstein metric (both the training set W_{tr} and the testing set W_{te}). These have λ equal to zero and the value of δ equal to two (red color) or equal to one (green color). Respectively for

Table 2 Models with a Regularization parameter for 100 samples

	#basis point	δ	θ	λ	W_{tr}	W_{te}	χ^2_{tr}	χ^2_{te}
W_{tr}	20	2.0	5.0	0.0	0.09	0.10	1.33	1.36
	48	2.0	5.0	0.0	0.14	0.13	2.02	2.01
	20	2.0	3.5	0.0	0.15	0.16	0.92	0.95
	20	1.0	5.0	0.0	0.15	0.16	1.18	1.20
	48	1.0	2.5	0.0	0.16	0.17	0.89	0.92
W_{te}	20	2.0	5.0	0.0	0.09	0.10	1.33	1.36
	48	2.0	5.0	0.0	0.14	0.13	2.02	2.01
	20	2.0	3.5	0.0	0.15	0.16	0.92	0.95
	20	1.0	5.0	0.0	0.15	0.16	1.18	1.20
	48	1.0	2.5	0.0	0.16	0.17	0.89	0.92
χ^2_{tr}	20	1.0	2.5	0.0	0.2318	0.2439	0.7756	0.8078
	48	6.0	2.5	0.0	0.2319	0.2440	0.7758	0.8083
	20	2.0	2.5	0.0	0.2318	0.2437	0.7759	0.8079
	15	6.0	2.5	0.0	0.2311	0.2432	0.7759	0.8081
	15	1.0	2.5	0.0	0.2305	0.2425	0.7762	0.8089
χ^2_{te}	20	1.0	2.5	0.0	0.2318	0.2439	0.7756	0.8078
	20	2.0	2.5	0.0	0.2318	0.2437	0.7759	0.8079
	15	6.0	2.5	0.0	0.2311	0.2432	0.7759	0.8081
	48	6.0	2.5	0.0	0.2319	0.2440	0.7758	0.8083
	15	1.0	2.5	0.0	0.2305	0.2425	0.7762	0.8089

the Chi-square error, the parameters that appear frequently are the θ equal to 2.5 and λ equal to 0. For both metrics, the number of basis points varies.

In table 3, the minimum values of the Wasserstein measure of the datasets (training set and test set), are found in the models with the initially preferable number of basis points 48 and secondary 20. For the rest of the parameters, we do not observe any pattern. The same conclusion for the number of base units applies to models with a minimum Chi-square value. In addition, in this case, the optimal choice for the parameter θ is 2.5, and for the parameter δ, first, it is the value 6.0, and second, 1.0. Although, in the case of the regularization matrix, we do not observe any pattern for the parameter λ.

The performance of different values for each parameter can be viewed in Appendices C and D. There is no further discussion because each time every model has different behavior depending on the variable parameter. We must also mention that there is no specific behavioral pattern.

Table 3 Models with a Regularization matrix for 100 samples

	# basis point	δ	θ	λ	W_{tr}	W_{te}	χ^2_{tr}	χ^2_{te}
W_{tr}	48	2.0	5.0	0.01	0.07	0.07	1.73	1.73
	48	1.0	3.5	0.001	0.09	0.09	1.20	1.24
	20	2.0	5.0	0.0	0.09	0.10	1.33	1.36
	48	6.0	3.5	0.1	0.11	0.12	1.08	1.09
	48	2.0	2.5	0.001	0.12	0.12	1.06	1.10
W_{te}	48	2.0	5.0	0.01	0.07	0.07	1.73	1.73
	48	1.0	3.5	0.001	0.09	0.09	1.20	1.24
	20	2.0	5.0	0.0	0.09	0.10	1.33	1.36
	48	2.0	2.5	0.001	0.12	0.12	1.06	1.10
	48	6.0	3.5	0.1	0.11	0.12	1.08	1.09
χ^2_{tr}	48	6.0	2.5	0.1	0.2318	0.2440	0.7752	0.8077
	48	6.0	2.5	0.001	0.2319	0.2444	0.7755	0.8073
	20	1.0	2.5	0.0	0.2318	0.2439	0.7756	0.8078
	48	6.0	2.5	0.0	0.2319	0.2440	0.7758	0.8083
	20	1.0	2.5	0.001	0.2317	0.2437	0.7758	0.8078
χ^2_{te}	48	6.0	2.5	0.001	0.2319	0.2444	0.7755	0.8073
	48	6.0	2.5	0.1	0.2378	0.2440	0.7752	0.8077
	20	1.0	2.5	0.0	0.2318	0.2439	0.7756	0.80783
	20	1.0	2.5	0.001	0.2317	0.2437	0.7758	0.80784
	20	1.0	2.5	0.01	0.2318	0.2436	0.7759	0.80786

5.2.1 Results for more samples

In this section, we tested the optimal models for more samples. We chose to use 4000 samples of the 7400 in total due to the computationally cost. In table 4 we observe that almost all the models present similar results. Also, in this case, the first model with parameters λI (20 2.0 5.0 0.0), the third one with parameters K_{mn} (48 2.0 5.0 0.01), and the LJ model have the maximum value for the Wasserstein and Chi-square metrics. The rest three models present better results. Between the Gaussian models, the minimum value of Chi-square gives the model λI (20 1.0 2.5 0.0) (except the spline model) and the minimum Wasserstein value the model K_{mn} (48 2.0 5.0 0.01). The observations of the figures 13 and 14 (of 4000 samples) present similar quality with the corresponding figures 9 and 10 of 100 samples.

Table 4 Models with minimum values of Chi-Square and Wasserstein distance for 4000 samples

Method & Samples	Wasserstein train	Wasserstein test	Chi-Square train	Chi-Square test
$\lambda\mathbf{I}$ (20 2.0 5.0 0.0)	0.27	0.27	0.97	0.97
$\lambda\mathbf{I}$ (20 1.0 2.5 0.0)	0.23	0.23	0.78	0.78
K_{mn} (48 2.0 5.0 0.01)	0.23	0.22	0.89	0.89
K_{mn} (48 6.0 2.5 0.1)	0.23	0.23	0.78	0.78
Splines 48	0.24	0.23	0.77	0.77
LJ	0.26	0.26	0.79	0.78

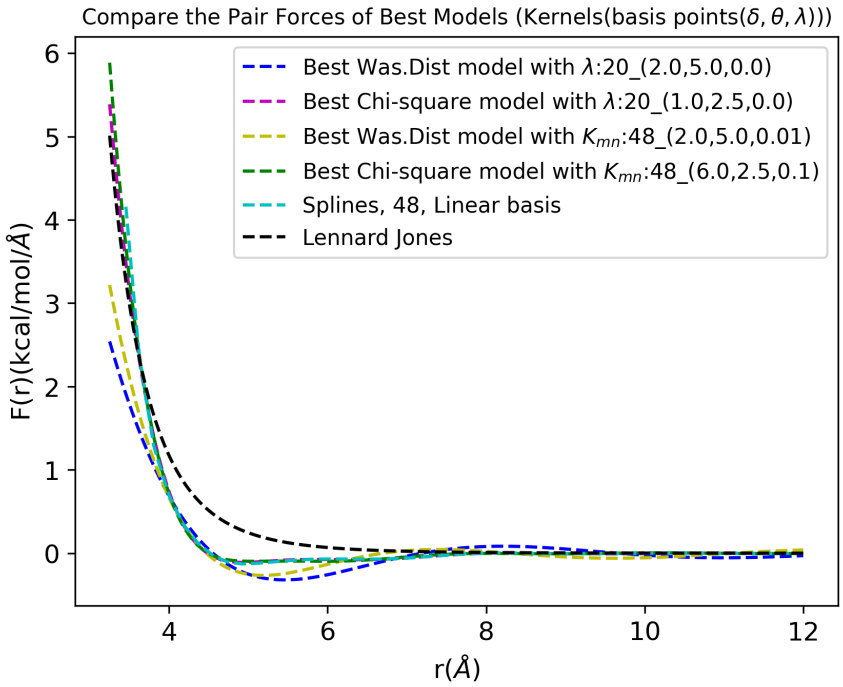


Figure 13: Pair Forces plot for the methane system (4000 samples)

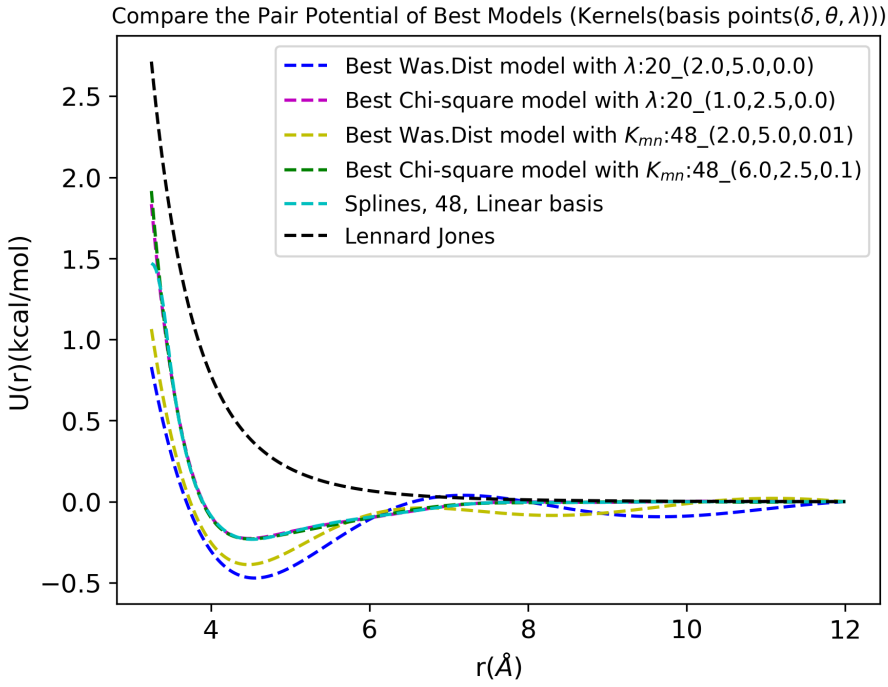


Figure 14: Potential plot according to the Pair Forces values for the methane system (4000 samples)

5.3 Implementation of the model

The implementation of the two models was accomplished in Python. The codes of both the dimer and the trimer Gaussian models, constructed from scratch and are linked in https://github.com/VassiaKyr/Master_Thesis.git. The results we present refer to the model with only dimer descriptors because the method with trimer descriptors has a computational cost that did not allow us to test it for many samples.

5.4 Comments on the computational cost of the methods

All methods except the Lennard-Jones method, which were studied in this work, are computationally demanding. As a result, parallel calculations are mandatory for the FM problem. The heavy workload makes it difficult to experiment with different parameters.

As it is observed, we present the results of the FM problem with only the dimer descriptor. The code of the FM problem with the trimer descriptor has been programmed, but the computational cost prevents us from the training, even though we perform a parallel code. To understand the time required, for a single sample with as few as possible, five basis points in each dimension (a total of $5^3 = 125$ basis points), the kernel matrix preparation process can take half a day. At this point, we should clarify that in this code we only calculate $F_{trimer} = \mathbf{F}^*$ applying in the training part not F_{target} but the difference between this quantity and the values of the forces F_{dimer} of the best FM dimer model, i.e. $F_{trimer} - F_{dimer}$.

The FM problem with only the dimer descriptor faces the same challenge on a different scale. The needed time to complete a training depends on the value of the number of basis points. Specifically, the code for 100 samples may take from a half hour to three hours for one combination of parameters corresponding to a specific value of the basis point. In the case of a model with 4000 samples, it took two days approximately for 48 basis points to reach and complete the training part.

5.5 Conclusion and Future Work

In molecular systems, the goal is to study atoms' movement on a very small order of magnitude. The main challenge here is the expensive calculations of the dynamic quantities due to the wide range of length and time scales and their as-accurate-as-possible approximation. The CG method facilitates the computational cost challenge by decreasing the number of degrees of freedom and the number of variables.

We studied the Gaussian process regression methods and applied them in the equivalent form, the Kernel Ridge Regression method to address the Force Matching (FM) problem. The FM is the minimization problem of the average distance between the forces calculated from atomistic simulations and the corresponding approximate CG forces. To understand the results of the method we also applied the Lennard-Jones method and the Linear B-splines representation. For each case, the measurements of Chi-square error and Wasserstein metric facilitate the interpretation of the results.

We observe that the Gaussian kernel models produce results of similar quality to the splines method. In a simple molecular system like the methane one, both Gaussian and Splines models require an almost equal computational cost. In the Gaussian method, the set

of hyperparameters is a determining factor in the construction of a fitting model, in contrast to the Splines method, which only contains the value of the basis points as a hyperparameter. The metrics that we evaluate each time, provide a measure to test the models. Both of them are significant, although, for a small dataset, the Chi-square error seems to have better results.

Nevertheless, the construction of the KRR model can be improved. Firstly, instead of the grid search which is applied so that we learn the hyperparameters, another option is the maximization of the likelihood function, so it will be more likely to yield the optimal set of hyperparameters. Secondly, except for the Wasserstein metric we can test the results for another probability density distance.

In molecular systems, the more samples used, the more optimal the models are. Therefore, we need to test our models for more samples and also for more complex systems in order to see how they behave.

Last but not least, the trimer term for the representation of the total force must be included. In this case, we can apply a cut-off function to reduce the complexity of the algorithm as suggested by ref.[7]. The same technique can be also applied to the dimer kernel matrix.

Appendices

A Example Algorithm

Here there are the codes of the Gaussian model examples.

```
import numpy as np

#------------data--------------------------
X = np.array([[1,0.9], [1,3.8], [1,5.2], [1,6.1], [1,7.5], [1,9.6]])
t = np.array([0.1,1.2,2.1,1.1,1.5,1.2])
S = [[0.1, 1.1], [1.1, 0.1]]
x_new = np.array([1.0,3.0])
beta_inv = 0.1

#----------model of weight space view-------
print('Model of weight space view')
A = np.linalg.inv(S) + (beta_inv**(-1))*np.dot(X.T,X)
a = (beta_inv**(-1)*(np.dot(np.dot(np.linalg.inv(A),X.T),t.T) ))
print('mean = ', a)
print('A^(-1) = ',np.linalg.inv(A))
y_new = np.dot(x_new.T,a)
print("y_new", y_new)
print()

#----------least squares model model-------
print('Least squares model model')
A = np.dot(X.T,X)
a = np.dot(np.dot(np.linalg.inv(A),X.T),t.T)
y_new = np.dot(x_new.T,a)
print("y_new", y_new)
print()

#----------model of function space view-------
print('Model of function space view')
x = np.array([0.9, 3.8, 5.2, 6.1, 7.5, 9.6])
delta = 1.0
theta = 2.2
x_n = 3.0
K = np.zeros((6,6))
for i in range(6):
    for j in range(6):
        k = (delta**2)*np.exp(-(np.abs(x[i]-x[j])**2)/(2*theta**2))
        K[i][j] = k
a = np.dot(np.linalg.inv(K+beta_inv*np.identity(6)), t.T)
print('a = ', a)

kn = np.array([ (delta**2)*np.exp(-(np.abs(x_n-x[i])**2)/(2*theta**2)) for
    i in range(6)])
print('kn = ', kn)
akn = np.array([a[i]*kn[i] for i in range(6)])
print('akn = ', akn)
print("y_new",sum(akn))
```

B Analytical Calculation of Trimer Terms

- Analytical calculation of $\frac{\partial k^*(\mathbf{x}_I,(d_i,d_j,d_k))}{\partial(d_j)}$

$$\begin{aligned}
\frac{\partial k^*(\mathbf{x}_I,(d_1,d_2,d_3))}{\partial(d_j)} &= \frac{\partial\big[\delta_*^2 \exp\sum_{i=1}^{3}(-\frac{|x_{I,i}-d_i|^2}{2(\theta_*^i)^2})\big]}{\partial(d_j)} \\
&= \delta_*^2\frac{\partial\big[\exp\Big(-\frac{|x_{I,1}-d_1|^2}{2(\theta_*^1)^2}-\frac{|x_{I,2}-d_2|^2}{2(\theta_*^2)^2}-\frac{|x_{I,3}-d_3|^2}{2(\theta_*^3)^2}\Big)\big]}{\partial(d_j)} \\
&= \delta_*^2\frac{\partial\big[\exp\Big(-\frac{|x_{I,1}-d_1|^2}{2(\theta_*^1)^2}-\frac{|x_{I,2}-d_2|^2}{2(\theta_*^2)^2}-\frac{|x_{I,3}-d_3|^2}{2(\theta_*^3)^2}\Big)\big]}{\partial(d_j)} \\
&= k^*(\mathbf{x}_I,(d_1,d_2,d_3))\frac{\partial(-\frac{|x_{I,1}-d_1|^2}{2(\theta_*^1)^2}-\frac{|x_{I,2}-d_2|^2}{2(\theta_*^2)^2}-\frac{|x_{I,3}-d_3|^2}{2(\theta_*^3)^2})}{\partial(d_j)} \\
&= -\frac{1}{(\theta_*^j)^2}k^*(\mathbf{x}_I,(d_1,d_2,d_3))|x_{I,j}-d_j|\frac{\partial(|x_{I,j}-d_j|)}{\partial(d_j)} \\
&= -\frac{1}{(\theta_*^j)^2}k^*(\mathbf{x}_I,(d_1,d_2,d_3))|x_{I,j}-d_j|(-\frac{x_{I,j}-d_j}{|x_{I,j}-d_j|}) \\
&= \frac{x_{I,j}-d_j}{(\theta_*^j)^2}k^*(\mathbf{x}_I,(d_1,d_2,d_3))
\end{aligned} \tag{115}$$

- Analytical calculation of $\nabla_{\mathbf{Q}_i}d_j(\cdot)$ according to 4.2

1. If j=1:

$$\begin{aligned}
\nabla_{\mathbf{Q}_i}d_1(\cdot) &= \nabla_{\mathbf{Q}_i}(r_{jk}+r_{jl}) \\
&= -\frac{\mathbf{Q}_j-\mathbf{Q}_k}{|\mathbf{Q}_j-\mathbf{Q}_k|}\delta_{ik}-\frac{\mathbf{Q}_j-\mathbf{Q}_l}{|\mathbf{Q}_j-\mathbf{Q}_l|}\delta_{il}+(\frac{\mathbf{Q}_j-\mathbf{Q}_k}{|\mathbf{Q}_j-\mathbf{Q}_k|}+\frac{\mathbf{Q}_j-\mathbf{Q}_l}{|\mathbf{Q}_j-\mathbf{Q}_l|})\delta_{ij}
\end{aligned} \tag{116}$$

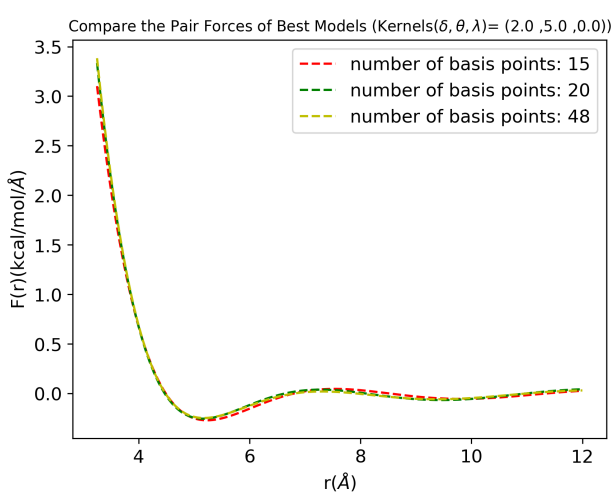
2. If j=2:

$$\begin{aligned}
\nabla_{\mathbf{Q}_i}d_2(\cdot) &= \nabla_{\mathbf{Q}_i}((r_{jk}-r_{jl})^2) \\
&= 2(r_{jk}-r_{jl})\nabla_{\mathbf{Q}_i}(r_{jk}-r_{jl}) \\
&= 2(r_{jk}-r_{jl})(-\frac{\mathbf{Q}_j-\mathbf{Q}_k}{|\mathbf{Q}_j-\mathbf{Q}_k|}\delta_{ik}+\frac{\mathbf{Q}_j-\mathbf{Q}_l}{|\mathbf{Q}_j-\mathbf{Q}_l|}\delta_{il}+(\frac{\mathbf{Q}_j-\mathbf{Q}_k}{|\mathbf{Q}_j-\mathbf{Q}_k|}-\frac{\mathbf{Q}_j-\mathbf{Q}_l}{|\mathbf{Q}_j-\mathbf{Q}_l|})\delta_{ij})
\end{aligned} \tag{117}$$

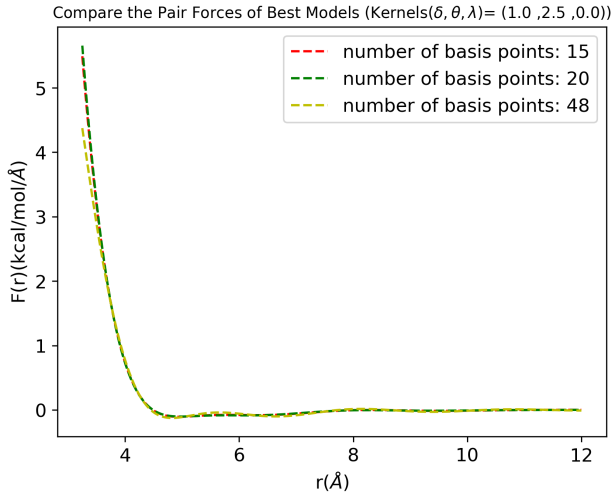
3. If j=3:

$$\nabla_{\mathbf{Q}_i} d_3(\cdot) = \nabla_{\mathbf{Q}_i} r_{kl} = \frac{\mathbf{Q}_k - \mathbf{Q}_l}{|\mathbf{Q}_k - \mathbf{Q}_l|}\delta_{ik} - \frac{\mathbf{Q}_k - \mathbf{Q}_l}{|\mathbf{Q}_k - \mathbf{Q}_l|}\delta_{il} \tag{118}$$

C Plots of the Dimer model - Regularization parameter

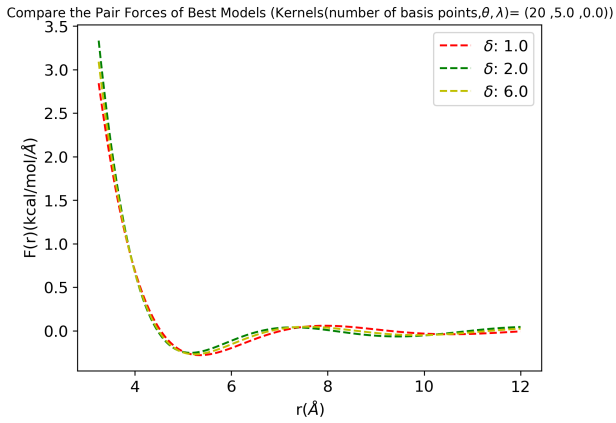


(a) $\#P_1$: $W = 0.24, \chi^2 = 0.89$,
$\#P_2$: $W = 0.09, \chi^2 = 1.33$,
$\#P_3$: $W = 0.14, \chi^2 = 2.02$

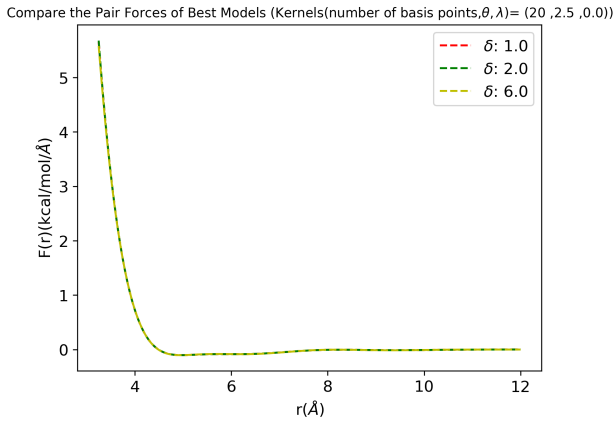


(b) $\#P_1$: $W = 0.23, \chi^2 = 0.77$,
$\#P_2$: $W = 0.23, \chi^2 = 0.77$,
$\#P_3$: $W = 0.16, \chi^2 = 0.89$

Figure 15: Different values of the number of basis points

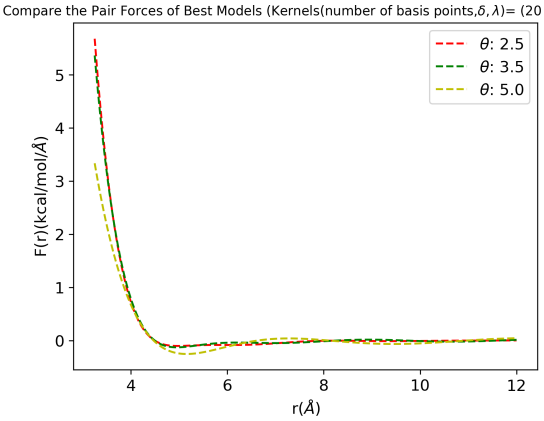


(a) δ_1: $W = 0.15, \chi^2 = 1.18$,
δ_2: $W = 0.09, \chi^2 = 1.33$
δ_3: $W = 0.24, \chi^2 = 0.88$

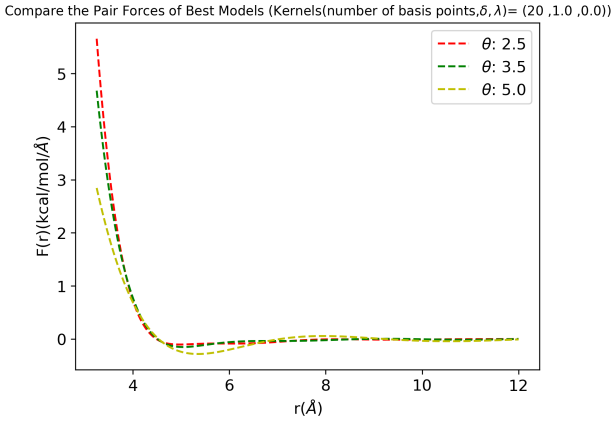


(b) δ_1: $W = 0.23, \chi^2 = 0.77$,
δ_2: $W = 0.23, \chi^2 = 0.77$
δ_3: $W = 0.23, \chi^2 = 0.77$

Figure 16: Different values of δ

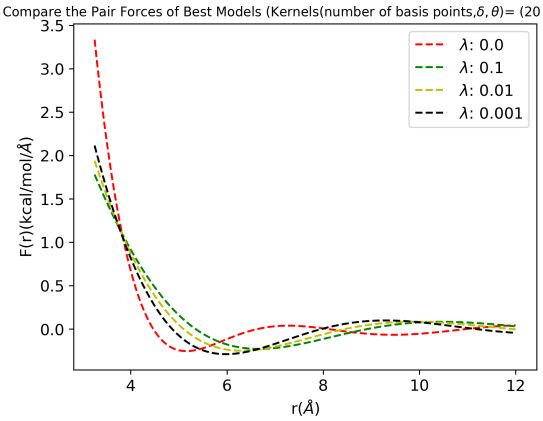


(a) θ_1: $W = 0.23, \chi^2 = 0.77$,
θ_2: $W = 0.15, \chi^2 = 0.92$
θ_3: $W = 0.09, \chi^2 = 1.33$

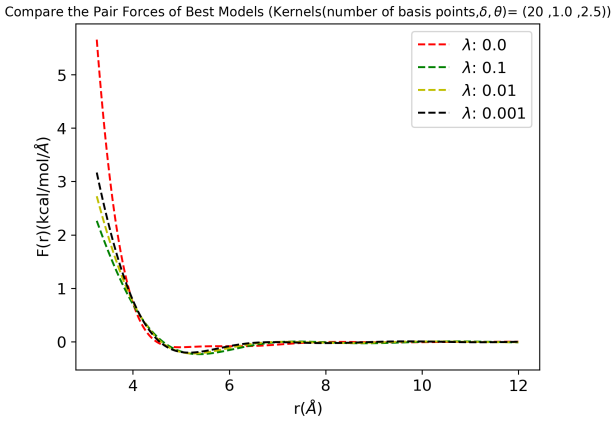


(b) θ_1: $W = 0.23, \chi^2 = 0.77$,
θ_2: $W = 0.20, \chi^2 = 0.82$
θ_3: $W = 0.15, \chi^2 = 1.18$

Figure 17: Different values of θ



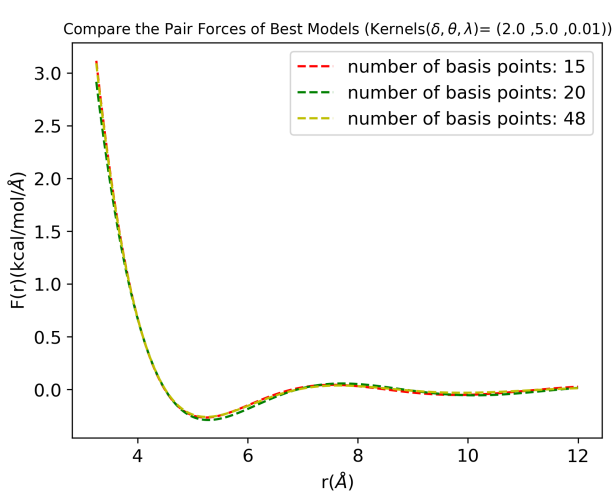
(a) λ_1: $W = 0.09, \chi^2 = 1.33$,
λ_2: $W = 0.42, \chi^2 = 1.11$
λ_3: $W = 0.40, \chi^2 = 1.06$
λ_4: $W = 0.36, \chi^2 = 1.02$



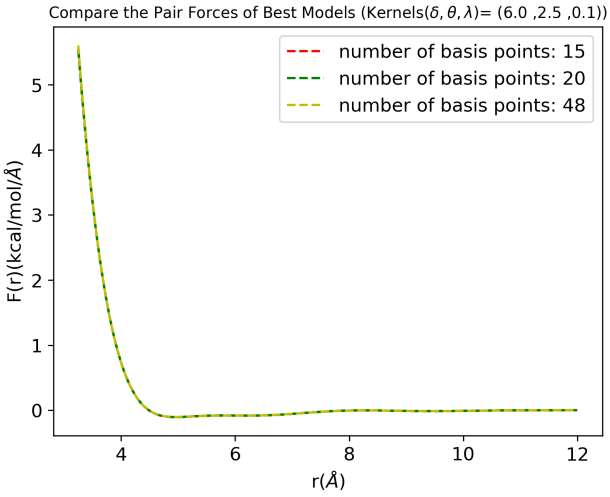
(b) λ_1: $W = 0.23, \chi^2 = 0.77$,
λ_2: $W = 0.30, \chi^2 = 0.91$
λ_3: $W = 0.27, \chi^2 = 0.86$
λ_4: $W = 0.25, \chi^2 = 0.83$

Figure 18: Different values of λ

D Plots of the Dimer model - Regularization matrix

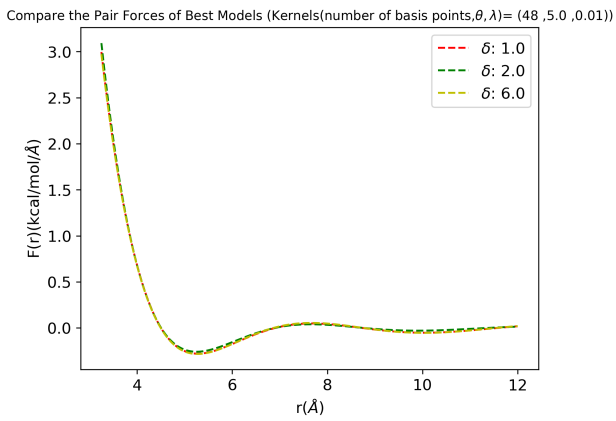


(a) $\#P_1$: $W = 0.22, \chi^2 = 0.91$,
$\#P_2$: $W = 0.25, \chi^2 = 0.90$,
$\#P_3$: $W = 0.07, \chi^2 = 1.73$,

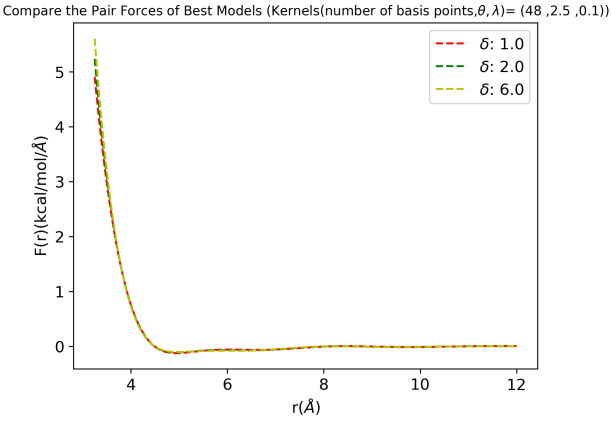


(b) $\#P_1$: $W = 0.23, \chi^2 = 0.77$,
$\#P_2$: $W = 0.23, \chi^2 = 0.77$,
$\#P_3$: $W = 0.23, \chi^2 = 0.77$

Figure 19: Different values of the number of basis

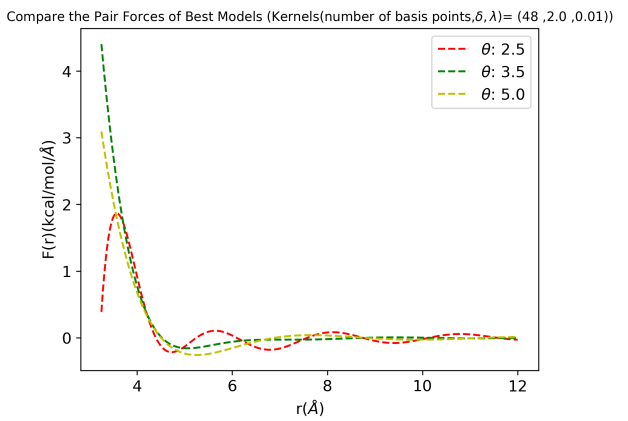


(a) δ_1: $W = 0.18, \chi^2 = 1.03$,
δ_2: $W = 0.07, \chi^2 = 1.73$
δ_3: $W = 0.25, \chi^2 = 0.88$

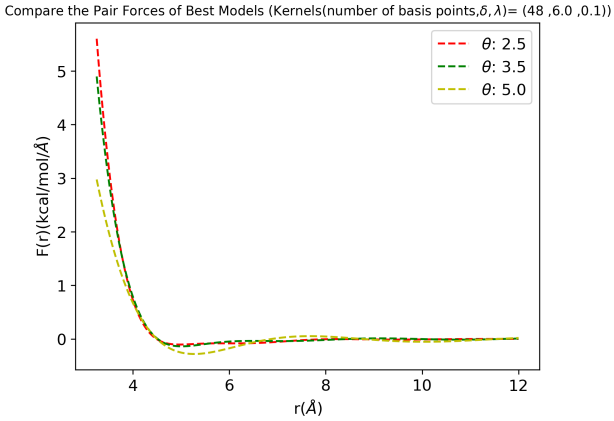


(b) δ_1: $W = 0.22, \chi^2 = 0.78$,
δ_2: $W = 0.22, \chi^2 = 0.78$
δ_3: $W = 0.23, \chi^2 = 0.77$

Figure 20: Different values of δ



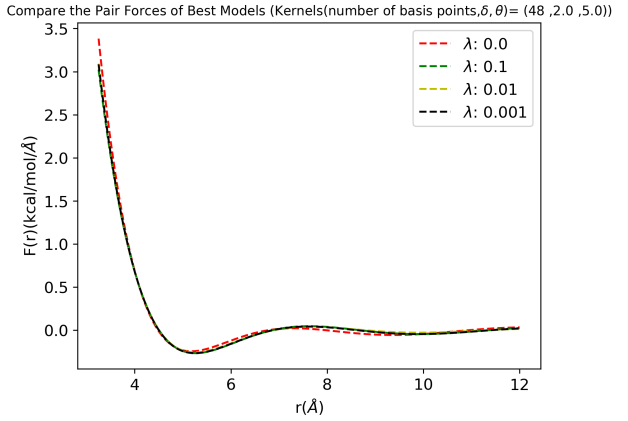
(a) θ_1: $W = 0.29, \chi^2 = 2.40$,
θ_2: $W = 0.21, \chi^2 = 0.82$
θ_3: $W = 0.07, \chi^2 = 1.73$



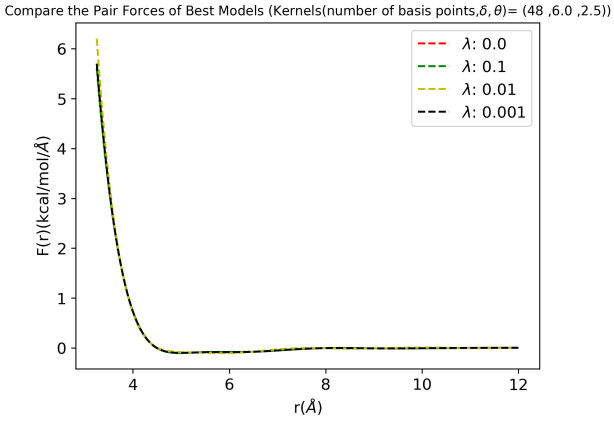
(b) θ_1: $W = 0.23, \chi^2 = 0.77$,
θ_2: $W = 0.11, \chi^2 = 1.08$
θ_3: $W = 0.25, \chi^2 = 0.88$

Figure 21: Different values of θ

Different values of λ



(a) λ_1: $W = 0.14, \chi^2 = 2.02$,
λ_2: $W = 0.21, \chi^2 = 0.96$
λ_3: $W = 0.07, \chi^2 = 1.73$
λ_4: $W = 0.23, \chi^2 = 0.91$



(b) λ_1: $W = 0.23, \chi^2 = 0.77$,
λ_2: $W = 0.23, \chi^2 = 0.77$
λ_3: $W = 0.19, \chi^2 = 2.00$
λ_4: $W = 0.23, \chi^2 = 0.77$

Figure 22: Different values of λ

References

[1] Anastasios Tsourtis, Vagelis Harmandaris, and Dimitrios Tsagkarogiannis. Parameterization of coarse-grained molecular interactions through potential of mean force calculations and cluster expansion techniques. *Entropy*, 19(8):395, 2017.

[2] E. Kalligiannaki, A. Chazirakis, A. Tsourtis, M.A. Katsoulakis, P. Plech, and V. Harmandaris. Parametrizing coarse-grained models for molecular systems at equilibrium. *The European Physical Journal Special Topics*, 225(8):1347–1372, 2016.

[3] E. Kalligiannaki, V. Harmandaris, M.A. Katsoulakis, and P. Plech. The geometry of generalized force matching and related information metrics in coarse-graining of molecular systems. *The Journal of Chemical Physics*, 143(8), 2015.

[4] Akira Satoh. *Introduction to practice of molecular simulation: molecular dynamics, Monte Carlo, Brownian dynamics, Lattice Boltzmann and dissipative particle dynamics*. Elsevier, 2010.

[5] Grigorios A Pavliotis. Stochastic processes and applications. *Informe técnico*, 2015.

[6] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[7] Volker L Deringer, Albert P Bartók, Noam Bernstein, David M Wilkins, Michele Ceriotti, and Gábor Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021.

[8] ST John and Gábor Csányi. Many-body coarse-grained interactions using gaussian approximation potentials. *The Journal of Physical Chemistry B*, 121(48):10934–10949, 2017.

[9] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*, volume 1. Elsevier, 2001.

[10] Vasiliki Kyrieri. Machine learning methods for parameterizing molecular models. University of Crete, Department of Mathematics and Applied Mathematics, 2020.

[11] Vlasis G. Mavrantzas Vagelis A. Harmandaris. *Molecular Dynamics Simulations of Polymers", Chapter in Book "Simulation Methods for Polymers*. M.J. Kotelyanskii and D.N. Theodorou, Marcel Dekker, New York, 2004.

[12] Carsten Hartmann. Molecular dynamics. with deterministic and stochastic numerical methods, 2016.

[13] Hasitha Muthumala Waidyasooriya, Masanori Hariyama, and Kota Kasahara. An fpga accelerator for molecular dynamics simulation using opencl. *Int. J. Networked Distributed Comput.*, 5(1):52–61, 2017.

[14] Jiapu Zhang. Canonical dual theory applied to a lennard-jones potential minimization problem. *arXiv: Optimization and Control*, 2011.

[15] Jacob Chapman. *Improving the Functional Control of Ferroelectrics using Insights from Atomistic modelling*. PhD thesis, 09 2018.

[16] Sergei Simdyankin and M. Dzugutov. Case study: Computational physics – the molecular dynamics method. 06 2022.

[17] WG Noid, Pu Liu, Yanting Wang, Jhih-Wei Chu, Gary S Ayton, Sergei Izvekov, Hans C Andersen, and Gregory A Voth. The multiscale coarse-graining method. ii. numerical implementation for coarse-grained molecular models. *The Journal of chemical physics*, 128(24):244115, 2008.

[18] Alison L Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002.

[19] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[20] Albert P Bartók, James Kermode, et al. Improved uncertainty quantification for gaussian process regression based interatomic potentials. *arXiv preprint arXiv:2206.08744*, 2022.

[21] Florian Müller-Plathe. Coarse-graining in polymer simulation: from the atomistic to the mesoscopic scale and back. *ChemPhysChem*, 3(9):754–769, 2002.

[22] VA Harmandaris, NP Adhikari, Nico FA van der Vegt, and Kurt Kremer. Hierarchical modeling of polystyrene: From atomistic to coarse-grained simulations. *Macromolecules*, 39(19):6708–6719, 2006.

[23] Sergei Izvekov and Gregory A Voth. Multiscale coarse-graining of liquid-state systems. *The Journal of chemical physics*, 123(13):134105, 2005.

[24] Joseph F Rudzinski and WG Noid. Coarse-graining entropy, forces, and structures. *The Journal of chemical physics*, 135(21):214101, 2011.

[25] Sergei Izvekov and Gregory A Voth. Effective force field for liquid hydrogen fluoride from ab initio molecular dynamics simulation using the force-matching method. *The Journal of Physical Chemistry B*, 109(14):6573–6586, 2005.

[26] M Scott Shell. The relative entropy is fundamental to multiscale and inverse thermodynamic problems. *The Journal of chemical physics*, 129(14):144108, 2008.

[27] Aviel Chaimovich and M Scott Shell. Anomalous waterlike behavior in spherically-symmetric water models optimized with the relative entropy. *Physical Chemistry Chemical Physics*, 11(12):1901–1915, 2009.

[28] Markos A Katsoulakis and Petr Plechac. Information-theoretic tools for parametrized coarse-graining of non-equilibrium extended systems. *The Journal of chemical physics*, 139(7):074115, 2013.

[29] A. P. Lyubartsev and A. Laaksonen. On the Reduction of Molecular Degrees of Freedom in Computer Simulations. In M. Karttunen, A. Lukkarinen, and I. Vattulainen, editors, *Novel Methods in Soft Matter Simulations*, volume 640 of *Lecture Notes in Physics, Berlin Springer Verlag*, pages 219–244, 2004.

[30] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[31] E. Süli and D.F. Mayers. *An Introduction to Numerical Analysis*. An Introduction to Numerical Analysis. Cambridge University Press, 2003.

[32] Anthony Chazirakis, Vassia Kirieri, Ilias-Marios Sarris, Evangelia Kalligiannaki, and Vagelis Harmandaris. Neural network potential surfaces: A comparison of two approaches. *Procedia Computer Science*, 178:345–354, 2020.

[33] Jörg Behler. Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations. *Physical Chemistry Chemical Physics*, 13(40):17930–17955, 2011.

[34] Berend Smit Daan Frenkel. *Understanding Molecular Simulation, From Algorithms to Applications*. Academic Press, San Diego, California, 2002.

[35] V. Harmandaris, E. Kalligiannaki, M.A. Katsoulakis, and P. Plechac. Path-space variational inference for non-equilibrium coarse-grained systems. *Journal of Computational Physics*, 314:355–383, 2016.

[36] Evangelia Kalligiannaki, Markos Katsoulakis, Petr Plechac, and Vagelis Harmandaris. Path space force matching and relative entropy methods for coarse-graining molecular systems at transient regimes. *Procedia Computer Science*, 136:331 – 340, 2018.